Product Review Error Detection using LLM and Knowledge Graph

1. Project Overview

Customer reviews are a cornerstone of brand perception and purchasing decisions in the digital era. For Bosch USA, a global leader in professional power tools, ensuring the accuracy of these reviews is vital to maintaining customer trust and satisfaction. However, reviews often contain inaccuracies, such as misunderstandings of product functions or incorrect technical claims, which can mislead consumers and harm Bosch's reputation.

This solution story outlines an innovative AI-driven approach to tackle these challenges by integrating **Azure OpenAI's Large Language Model (LLM)** with a **Neo4j-based Knowledge Graph (KG)**. Developed in collaboration with Bosch USA, this solution automates the detection and classification of review errors, offering a scalable and reliable system to enhance review accuracy. By combining cutting-edge AI with structured domain knowledge, this project not only addresses immediate needs but also lays the groundwork for advanced customer feedback management.

2. Problem Statement

Bosch receives a massive volume of customer reviews annually, many of which contain inaccurate information. These errors generally fall into two key categories:

- **Semantic Misalignment:** Semantic misalignment occurs when a review contains logically or contextually incorrect statements that reflect a misunderstanding of the product's function, usage, or terminology. These reviews may appear grammatically correct and natural, but their meaning does not align with how the product is actually designed or used. This can interfere with product tracking, analytics, and insight extraction by introducing misleading narratives.

    - **Common cases include:**

      a. Misunderstanding product functions or intended usage: Customers may describe using tools in ways that are inconsistent with their actual purpose.
      - *Example:* "I used this impact driver to drill into concrete, but it barely made a dent." → Impact drivers are not intended for drilling into concrete, unlike hammer drills.

      b. Misinterpreting product terminology: Users may confuse technical terms or misunderstand common tool-related labels.
      - *Example:* "This cordless drill doesn't need a battery, which is convenient." → "Cordless" means battery-powered without a cord. It doesn't mean battery-free.

      c. Confusing tool categories or types: Some reviews incorrectly assume two different tools are functionally equivalent.
      - *Example:* "This grinder works just like a drill if you push hard enough." → Grinders are not designed for drilling; this usage reflects a misunderstanding.

- **Specialized Errors:** Specialized errors involve factually incorrect statements about a product's technical specifications or usage limits. These include incorrect voltage, torque, or battery specifications, as well as unsafe or unsupported usage scenarios. Unlike surface-level typos, these mistakes typically require product knowledge or reference to official documentation to detect.
    - **Common cases include:**

      a. Using the product outside of recommended operating conditions

      - *Example:* "I charged my tool at 50°C and it overheated!" → This exceeds the recommended charging temperature and may damage the battery.

b. Recommending unsafe or unofficial usage methods

- *Example:* "I use a 12V battery on my 18V driver with tape—it works!" → This is an unsupported and potentially hazardous setup.

c. Misleading claims about product performance

- *Example:* "This drill can torque to 80 Nm on any screw!" → This overstates the tool's actual capability and can lead to material damage or user injury.

While traditional rule-based NLP systems can detect surface-level patterns, they often lack the domain knowledge required to resolve such deeper ambiguities or technically incorrect statements. Manual review is also time-consuming and not scalable given the volume of content.

To address these challenges, we focused on these complex error types—semantic and specialized—within textual customer reviews. We extracted Bosch's official specifications through PDF parsing and web scraping, and structured this information into a Neo4j-based knowledge graph.

This graph serves as a trusted reference, providing verified ranges and canonical product terms. It enables Azure OpenAI to perform more accurate and context-aware analysis by grounding its reasoning within a bounded, domain-specific space.

3. Synthetic Data Generation

We created a synthetic dataset using Azure OpenAI to simulate diverse customer reviews.

- 20% of the reviews were intentionally injected with semantic and specialized errors to test detection performance.

4. Solution: AI-Driven Review Analysis with LLM (Azure OpenAI) + Knowledge Graph (Neo4j)

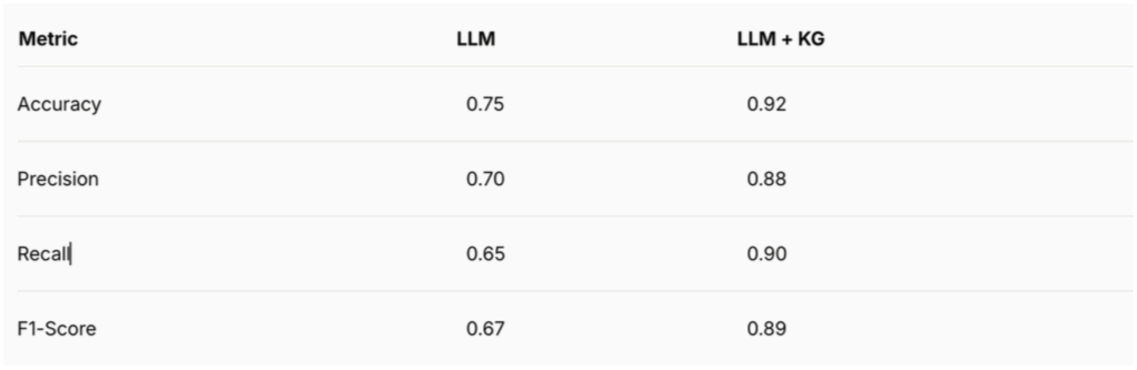
Designing an automated pipeline that integrates Azure OpenAI with a Neo4j-based Knowledge Graph to semantically extract, validate, and correct domain-specific errors in customer reviews

- **Solution Components**
    - **Large Language Model (LLM)**:
        - Azure OpenAI's LLM excels at understanding and generating human-like text, making it ideal for initial review analysis and error detection.
        - It processes review text to identify potential inaccuracies based on general language patterns.
    - **Knowledge Graph (KG)**:
        - A Neo4j-based KG is constructed from official Bosch product specifications, extracted from PDF manuals using PyMuPDF and Bosch's website via BeautifulSoup.
        - The KG serves as a trusted, structured repository of accurate product data, such as torque, RPM, and intended uses.
- **Process Flow**
    - **Build Knowledge Graph from PDF product manuals:**
        - Used PyMuPDF (fitz) to extract text from Bosch product manuals in PDF format.
        - Regular expression patterns were applied to better extract tool names, specifications, accessories, limitations, etc.
        - Generated custom Cypher queries based on the standardized data to create the knowledge graph structure.
        - Inserted nodes and relationships into Neo4j using the neo4j Python driver.
    - **Detect Errors via LLM and Knowledge Graph Comparison**

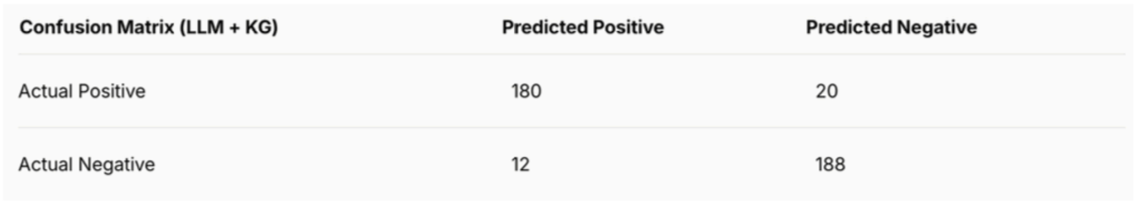
        - Each review was processed by extracting the associated product ID.
        - **Azure OpenAI + Neo4j AuraDB Cross-Checking**: Used Cypher queries to retrieve standard specifications and constraints from Neo4j and LLM prompt to evaluate whether the customer reviews aligned with these constraints. This included semantic matching (e.g., synonyms) and numerical value range checking (e.g., torque, temperature).
        - **Error Classification:**
            - **i.** Semantic Error → Detected if ambiguous language is used without clear context.
            - **ii.** Specialized Error → Detected if the user claim contradicts known specifications.
    - **Output Generation**
        - Convert results into a standardized and structured file format for internal usage.
        - Add an **error flag** and provide a **detailed explanation.**
        - Ensure a standardized output format for internal usage.

5. Performance Evaluation

- The overall performance was evaluated using standard metrics: **accuracy** (how often the system correctly identified whether a review had an error), **precision** (how many of the flagged errors were actually errors), **recall** (how many of the actual errors were flagged), and **F1-score** (a balance of precision and recall). The table below compares the two approaches:

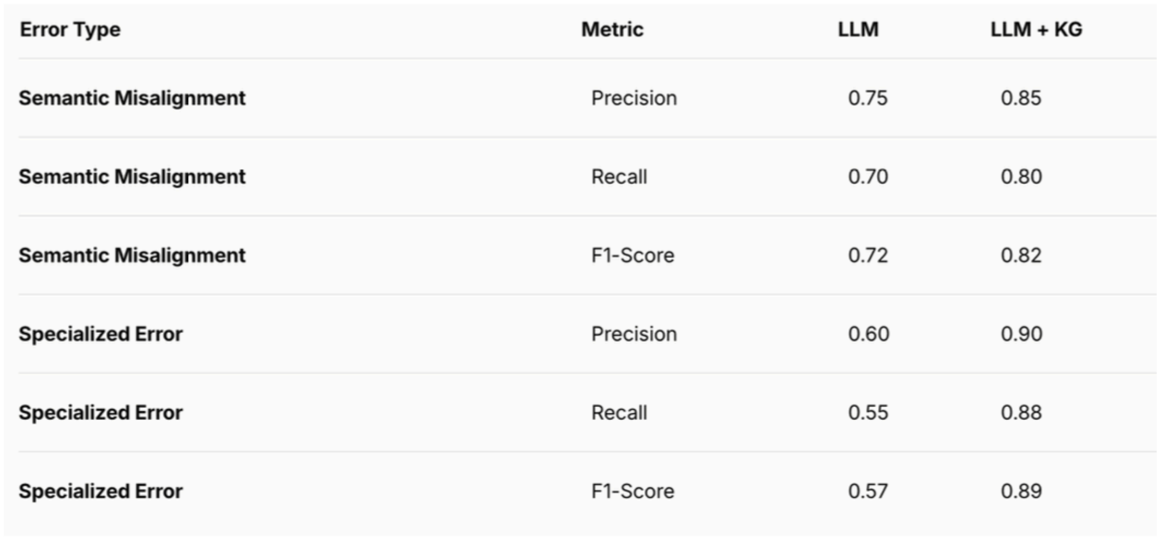


Metric	LLM	LLM + KG
Accuracy	0.75	0.92
Precision	0.70	0.88
Recall	0.65	0.90
F1-Score	0.67	0.89



Confusion Matrix (LLM + KG)	Predicted Positive	Predicted Negative
Actual Positive	180	20
Actual Negative	12	188

- **Separate Comparison of Semantic Misalignment and Specialized Errors:**
  To provide a more granular analysis, we evaluated the performance of each approach for detecting semantic misalignments and specialized errors separately. This is critical because semantic misalignments rely on language understanding, where the LLM alone may perform adequately, while specialized errors require factual accuracy, where the Knowledge Graph is essential:



Error Type	Metric	LLM	LLM + KG
Semantic Misalignment	Precision	0.75	0.85
Semantic Misalignment	Recall	0.70	0.80
Semantic Misalignment	F1-Score	0.72	0.82
Specialized Error	Precision	0.60	0.90
Specialized Error	Recall	0.55	0.88
Specialized Error	F1-Score	0.57	0.89

*Insights from the classification report will be analyzed, and conclusions will be drawn.

**Values in the screenshots are hypothetical and will be replaced with actual values.*

6. Lesson Learned

The project yielded valuable insights:

- **Domain Knowledge Value**
    - Combining LLMs with KGs significantly boosts accuracy in specialized tasks.
- **KG Construction Challenges**
    - Building a robust KG requires meticulous planning and maintenance.
- **Automation vs. Accuracy**
    - Balancing automation with precision may require human oversight for edge cases.

7. Limitations & Challenges

The solution, while effective, faces several challenges:

- **Knowledge Graph Expansion**
    - Scaling the KG to cover all Bosch products requires ongoing data extraction and updates, a resource-intensive process.
- **System Integration**
    - Integrating with Bosch's existing review platforms demands significant development to ensure compatibility.
- **Data Quality**
    - KG accuracy is critical; errors in source data could lead to incorrect validations.
- **Computational Resources**:
    - LLMs and KGs require substantial computing power, posing latency challenges for real-time analysis.
- **Scalability**:

    - Expanding to all product categories necessitates optimized prompts and possibly distributed computing.

8. Future Development – Scaling up

- Replicate the successful workflow
- After testing on small-scale testing, expand to additional product categories
- **Web Scraping:** Utilize **BeautifulSoup** to extract product specifications from Bosch’s official website. **JSON-LD metadata** and HTML tables are parsed to retrieve key attributes, which are then transformed into structured key–value pairs.
- **Enhance the Knowledge Graph** to cover new products and technical specification
- **Prompt optimization** ensures that prompts adapt to larger knowledge graph expansions.