Product Review Error Detection using LLM and Knowledge Graph

1. 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.

---

2. 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

Core Steps:

1) Small-Scale Testing

- **Create a synthetic dataset using Azure OpenAI**
    - Simulate real customer reviews to enrich dataset diversity.
    - Introduce 20% erroneous reviews (semantic & specialized errors)
- **Build Knowledge Graph from PDF product manuals:**
    - PDF Parsing & Extraction:
        - **Used PyMuPDF (fitz)** to extract text from Bosch product manuals in PDF format.
        - **Regular expression** patterns were applied to strengthen extraction of tool names, numeric units, and range values.
        - The extracted content was sent to Azure OpenAI with custom prompts to identify product entities and relationships.
        - Using pre-defined node labels (e.g., 'Tool', 'Specification', etc.), transform LLM outputs into GraphDocument objects via LangChain's LLMGraphTransformer.
    - **Neo4j Graph Insertion**
        - The structured knowledge graph was written to Neo4j using the py2neo and neo4j packages, ensuring unique nodes via the MERGE clause.
- **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**:
        - **Semantic Error** → Detected if ambiguous language is used without clear context.
        - **Specialized Error** → Detected if the user claim contradicts known specifications.
- **Error Dataset Generation**
    - Convert results into a structured **file format**
    - Add an **error flag** and provide a **detailed explanation**
    - Ensure a standardized output format for internal usage
- **Evaluate Accuracy & Improve Prompts**

    - Use accuracy metrics as key indicators to compare the performance of the **LLM-only** approach and the combined **LLM + Knowledge Graph** method.

2) 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:** Ensure prompts adapt to larger knowledge graph expansions
- Develop automation tools to:
    - Process review datasets automatically
    - Detect and analyze errors at scale

---

3. Limitations & Challenges

- Knowledge Graph Expansion Challenges: Building a large-scale knowledge graph is not easy. Even for a single product, it may not cover all possible variables and error types.
- Integration with Existing Systems: Ensuring that LLM-generated data seamlessly integrates with Bosch's internal data pipelines and governance frameworks.
- Data Updates & Maintenance: AI models and knowledge graphs require regular updates to maintain data quality and prevent issues caused by outdated or obsolete information.
- Computational Resources & Latency: Processing large-scale data demands high computational power, which may impact system real-time performance and operational costs.