Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 20 No. 13 (2026) | OPEN ACCESS

https://doi.org/10.3991/ijim.v20i13.61569

PAPER

CATS 2.0: Leveraging Large Language Models and Graph Databases for Robust Arabic SMS E-Commerce Systems

Daoud M. Daoud[1], Samir Abou El-Seoud[2](✉), Hussain Al-Aqrabi[1]

[1]Higher Colleges of Technology, Sharjah, UAE

[2]British University, Cairo, Egypt

samir.elseoud@bue.edu.eg

ABSTRACT

Arabic SMS-based e-commerce platforms pose unique challenges due to the spontaneous and noisy nature of user-generated text (e.g., abbreviations, dialectal Arabic, or "Arabizi" transliterations). In this paper, we present Classified Ads Text Service (CATS) 2.0, an improved classified ads system that combines probabilistic large language models (LLMs) with deterministic graph-based knowledge representations to achieve robust understanding and matching of Arabic SMS content. Building on earlier work that emphasized the importance of integrating sublanguage analysis with content-oriented methods, our approach uses a hybrid pipeline: an LLM interprets free-text messages and extracts structured information, which is then inserted into a Neo4j graph database representing the domain knowledge. This graph-based representation enables precise semantic matching of "selling" and "looking for" posts and supports reasoning over the ads network. We evaluate the system on real-world Arabic SMS e-commerce data. Experimental results show that the hybrid CATS 2.0 system achieves high accuracy in content extraction (improving the F-measure over the original system's ~90%) and successfully handles multilingual and transliterated inputs. The proposed approach demonstrates how coupling an LLM's flexibility with a knowledge graph's rigor can substantially enhance the robustness and extensibility of e-commerce text processing in Arabic.

KEYWORDS

Arabic NLP, large language models (LLMs), knowledge graphs, SMS E-commerce, hybrid information extraction, semantic matching, noisy text

1 INTRODUCTION

E-commerce via SMS is a valuable service in regions where mobile text messaging is widespread. Users can post free-form "for sale" or "looking for" ads by SMS, but understanding these spontaneous, often ungrammatical messages poses significant natural language processing challenges. In Arabic, this is exacerbated by the use of dialectal words, inconsistent spelling (e.g., omission of diacritics), and even Latin-character transliterations (Arabizi). Traditional keyword or template-based

Daoud, D. M., El-Seoud, S. A., Al-Aqrabi, H. (2026). CATS 2.0: Leveraging Large Language Models and Graph Databases for Robust Arabic SMS E-Commerce Systems. *International Journal of Interactive Mobile Technologies (iJIM)*, 20(13), pp. 99–117. https://doi.org/10.3991/ijim.v20i13.61569

Article submitted 2026-03-16. Revision uploaded 2026-04-16. Final acceptance 2026-04-16.

systems struggle with such inputs. Classified Ads Text Service (CATS) was originally proposed as an SMS-based Arabic classified ads platform combining sublanguage analysis and content-oriented knowledge [2]. The original CATS proof-of-concept demonstrated that mixing domain-specific linguistic rules with a structured content model yields high extraction accuracy (around 90% F-measure on noisy SMS). This validated earlier insights in NLP that restricted-domain "sublanguages" (specialized jargons or styles) can be leveraged for better parsing [3]. Indeed, domain ontologies and controlled vocabularies have been used in e-commerce dialog systems to improve understanding [4], and knowledge-based agents for classified ads date back to the 1990s [5]. However, purely rule-based approaches require extensive expertise to develop and are brittle when encountering unforeseen phrasing.

Recent advances in large language models (LLMs) offer a new opportunity to handle the variability of natural language input. LLMs such as GPT-style models have demonstrated remarkable abilities in understanding and generating text across domains. In e-commerce applications, specialized instruction-tuned LLMs have even outperformed task-specific models and shown robust generalization to new products and queries [6]. This suggests an LLM could interpret Arabic SMS ads with greater flexibility than a hand-crafted parser. Yet, relying solely on an LLM has drawbacks: LLMs often lack deep domain expertise and may produce inconsistent or hallucinated outputs in specialized contexts [7]. In our scenario, an LLM might misunderstand a rare abbreviation or invent an incorrect product attribute without some grounding. To address this, we propose a hybrid neuro-symbolic approach that combines the probabilistic strengths of LLMs with the logical precision of a knowledge graph. By extracting structured information from the LLM and storing it in a graph database, we enforce consistency and can perform rule-based reasoning on the results. Such neuro-symbolic integration has been advocated to improve AI systems' robustness and interpretability [7]. The graph acts as a shared knowledge backbone: it captures domain entities (e.g., products, brands, attributes) and their relationships, enabling accurate matching and retrieval that pure text processing might miss. Notably, graph-based text representations have proven effective for Arabic NLP tasks like categorization [8], confirming that structured semantic knowledge can complement statistical language understanding.

In this paper, we introduce CATS 2.0, an end-to-end architecture for an Arabic SMS e-commerce system that tightly integrates an LLM with a Neo4j graph database. CATS 2.0 inherits the goal of the original CATS – a multilingual natural-language classified ads service – but modernizes it with current AI techniques. The system accepts free-form SMS messages in Arabic, uses the LLM to parse and convert them into a structured AD representation, and stores them in a knowledge graph for downstream querying. By combining a learned model with deterministic rules (through the graph schema and business logic), the system ensures both flexibility and reliability. We emphasize the car's domain as a case study, given its rich set of attributes (make, model, year, etc.) and prevalence in classifieds. Figure 2 illustrates the Neo4j schema for this domain, where nodes represent entities like Car, Manufacturer, and Seller, and relationships link ads to their attributes (e.g., an ad node has a "MAKE" relation to a Toyota node). This graph-based design not only improves matching of buyers to sellers but also simplifies multilingual extension: the knowledge graph can store labels in multiple languages for the same entity, and the LLM can be prompted or fine-tuned to output standardized representations from input in various languages.

The remainder of this paper is structured as follows. In Methodology, we detail the CATS 2.0 system architecture, including the LLM-based content extraction process and the design of the graph database for knowledge representation. Experimental results describe the evaluation on Arabic SMS data, comparing performance to the original rule-based baseline and examining multilingual capabilities. In discussion, we analyze the benefits of the hybrid approach and discuss challenges such as ambiguity, error handling, and deployment considerations. Finally, we conclude with a summary of findings and potential directions for future work, such as applying this approach to other domains and languages.

2 METHODOLOGY

CATS 2.0 is designed as a pipeline that processes incoming SMS messages and produces structured, queryable records in a graph database, enabling automated matchmaking between "sell" and "buy" advertisements. Figure 1 presents the overall system architecture. The core components include: (1) an LLM-based parser for content extraction, (2) a graph database (Neo4j) that serves as both a knowledge base and storage for ads, and (3) a matching and response generation module that uses deterministic rules and graph queries to connect buyers with relevant sellers and format the replies. The system operates in two modes based on the type of user message: a Sell Post mode (when a user wants to list an item for sale) and a Lookup Query mode (when a user is searching for an item). In either case, the input is a raw SMS string, typically in Arabic, which can be informal and may include abbreviations or non-standard spellings.

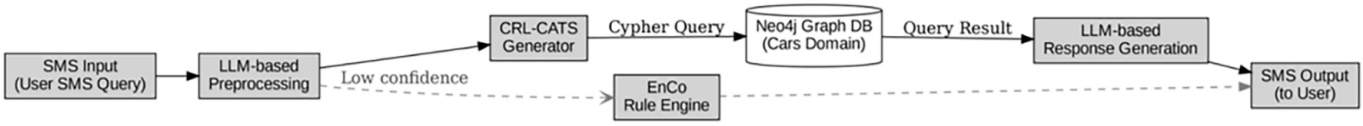

Fig. 1. CATS 2.0 architecture overview

When a new message arrives, a preprocessing step first normalizes the text. For Arabic, this may involve standardizing character variants (e.g., converting different forms of alef or ya to a canonical form) and, optionally, detecting if the message is in Arabizi (Arabic written in Latin characters); if so, a transliteration or translation step may be applied to convert it into Arabic script. After normalization, the text is passed to the LLM parser. We employ a prompt-based extraction approach: the LLM is given a carefully crafted prompt (possibly with examples) instructing it to identify key information in the ad and output a structured representation. For instance, if the domain is cars, the prompt might ask: "Extract the details of this ad: category (e.g., car), make, model, year, condition, price, location, and whether it's an offer or request. Respond in JSON." By asking the LLM for a JSON or similarly structured output, we ensure the results are machine-readable [7]. An example input and LLM output could be:

Input: "للبيع كامري 2015 فضي بحالة جيدة جدا ماشي 50 الف كم السعر 20 الف درهم" (Arabic for "For sale: Camry 2015, silver, very good condition, 50k km, price 20k AED"). LLM Output (JSON): {"category": "car", "make": "Toyota", "model": "Camry", "year": 2015, "color": "silver", "mileage": 50000, "condition": "very good", "price": 20000, "currency": "AED", "type": "sell"}.

In this output, the LLM has filled in a structured template with the ad's content. The Graph Insertion submodule then takes this structured data and upsets it into the Neo4j database. Here, the knowledge graph schema plays a vital role: it defines the entity types and relations expected. Following our example, the schema might have a CarAd node with properties or links for each attribute (Make, Model, etc.). If the LLM provides a make or model that already exists in the graph (e.g., "Toyota" as a manufacturer node, "Camry" as a model node linked to Toyota), the system will link the new ad to the existing nodes rather than duplicate them. If a new entity is encountered (say a model not seen before), the system can either create a new node or, if it's unrecognized and likely an error, flag it for review. The graph thus serves as a form of validation—inputs not conforming to known domain values can be detected. This is an example of the deterministic knowledge-based component correcting or constraining the LLM's output.

Algorithm 1. LLM + Graph Content Extraction and Insertion

Input: Raw SMS message M
Output: Graph node Ad inserted or flagged for review

1: Preprocess begin
Normalize: clean text, unify encoding.
Tokenize: extract URLs, phones, emails.
Transliterate: if Arabizi → Arabic.
end

2: Parse begin
LLM: apply schema-guided prompt.
Output: structured fields with confidence.
end

3: Validate begin
Check: enforce schema types/ranges.
Correct: apply heuristics (e.g., AED default).
end

4: Decision begin
Flag: if low confidence or errors → review queue.
Proceed: otherwise continue to insertion.
end

5: Insert begin
Upsert: create/merge Ad node in graph.
Link: connect Seller & Tags, store RawMessage.
Log: record status and return Ad node.
end

Once the ad is stored in the graph, the matching engine is triggered (for a sell post, it will try to find any pending "looking for" queries that match this item; for a lookup query, it will search among available "for sale" listings). This is implemented via graph queries (using Cypher, Neo4j's query language). Because the data is structured, such queries are precise: for instance, if a user is looking for a Toyota Camry, the system can perform a graph traversal to find all CarAd nodes of type "sell" connected to Model:Camry and Make:Toyota nodes, optionally filtering by year or price range if specified. This graph-based retrieval is inherently semantic—it will match

even if the wording between buyer and seller messages differs (one might say "كامري" in Arabic and another "Camry" in English; both link to the same Camry node in the knowledge graph). This approach addresses the "multi-hop" query problem often seen in information retrieval; instead of relying on text similarity, we leverage relationships in the graph to answer complex queries with high recall medium.com. The graph structure also allows additional reasoning: for example, the system could infer that a query for "Toyota 4x4 2018" might match a "Toyota Prado 2018" ad because Prado is a type of Toyota 4x4—such inference can be done if the ontology of vehicle types is encoded in the graph (via an "ISA" relationship between model and category).

Finally, the response generation module composes an SMS reply to the user. In CATS 2.0, we prioritize deterministic, template-based generation for responses to ensure clarity and correctness (especially important when sending potentially sensitive information like prices or phone numbers via SMS). For a lookup query result, the system might have a template like: "Found {N} results for {query}: 1) {year} {make} {model} - {price}{currency}, {location}; ... Reply STOP to end." The placeholders are filled with data from the matching graph nodes. By using templates, we guarantee the response is factual (drawn from the database) and linguistically simple. In cases where more natural-sounding or varied phrasing is desired, the LLM could be employed to generate the response sentence, but even then we constrain it by providing the factual slots to include. If no matches are found, the system sends a courteous failure message or a promise to alert later (e.g., "No listings found for Toyota Camry 2015 at the moment. We will notify you if one is posted."), which corresponds to handling the "no answer situations" as described in the original CATS designscielo.org.mx.

Algorithm 2. Buyer–Seller Matching

Input: Query Ad q
Output: Matching Sell Ads $\{s_1, s_2, \dots\}$

1: Initialize begin
Set: $Matches \leftarrow \emptyset$.
end

2: Search begin
For each $s \in Graph$ with $s.type =$ `'sell'`:
Check Model/Make: if $s.Model = q.Model$ AND $s.Make = q.Make$.
Check Filters: if $Price(s) \leq MaxPrice(q)$ AND $Location(s) \in Range(q)$.
Add: include s in $Matches$.
end

3: Rank begin
Sort: order $Matches$ by relevance (e.g., price, distance).
end

4: Select begin
Return: Top–K results from ranked $Matches$.
end

Through this architecture, CATS 2.0 ensures that the LLM and the graph database work in tandem. The LLM provides understanding of free text and flexibility to adapt to language variation, while the knowledge graph imposes structure and supports reliable, explainable operations (like matching and aggregation). This design

echoes the general pattern of retrieval-augmented generation, where an external knowledge source is used to ground the outputs of a language model [9]. Unlike open-domain QA systems that retrieve unstructured text, our system retrieves structured entries from a graph, which can then be directly used in an output. The result is a more robust pipeline: if the LLM output is imperfect, the graph's constraints (types and relations) catch many issues, and if the graph query is too strict (e.g., no direct match), the LLM's understanding can sometimes allow a fallback to a broader search or a rephrased query.

2.1 Large language model for Arabic content extraction

At the heart of CATS 2.0's understanding layer is a large language model specialized (or at least capable) in Arabic. We experimented with GPT-style models to parse the SMS input. Given the scarcity of large Arabic instruction datasets historically, we leveraged recent progress in Arabic NLP to inform our choice of model. One option is to use a multilingual foundation model (such as GPT-4 or open-source LLaMA variants) with prompting. Another approach is fine-tuning or using an Arabic-specific model. For instance, the newly released Jais 13B model has been touted as the most advanced Arabic LLM, with performance competitive to similar English models [10]. In our implementation, we initially use a GPT-3.5 Turbo model with few-shot prompts for rapid prototyping and plan to integrate a model like Jais or a fine-tuned LLaMA for on-premise deployment to avoid dependency on external APIs. The LLM is tasked with extracting a predefined set of fields from the message. We supply definitions or examples of each field in the prompt, e.g., "year: four-digit year of the vehicle (if mentioned)," to reduce ambiguity. Because users do not always follow a standard format, the LLM's ability to interpret context is invaluable. For example, given "كامري للبيع موديل 2015 وارد امريكا لون سلفر" ("Camry for sale, model 2015, US import, color silver"), a straightforward rule-based parser might fail to identify "وارد امريكا" as meaning an imported (U.S.-spec) vehicle, whereas an LLM can use its knowledge to recognize this phrase and perhaps mark an attribute import_origin: "USA." We observed that the LLM often implicitly handles synonyms and multilingual terms (e.g., it understands "silver" and "فضي" as the same color). This reduces the burden on developers to enumerate all variants.

To ensure the LLM's output is usable, we enforce a structured format (like JSON or XML) in the prompt. This technique of structured prompting is increasingly common to integrate LLMs with databases or external tools [9]. We also include in the prompt a list of possible categories for the ad (e.g., car, real estate, electronics) so that the model can explicitly classify the domain of the message. This is important for a multilingual, multi-category system; for instance, the word "Toyota" might appear in a message about a car or a toy (if miscapitalized), but context and model knowledge will usually discern that "Camry" is a car model. In cases where the model has low confidence or the message is ambiguous/incomplete, the system can either apply default rules or ask for clarification (though in the SMS context, interactive clarification is limited). Another deterministic layer we add is a lexicon check: after the LLM provides its extracted fields, we compare key fields (like make and model names) against a curated list in the knowledge graph. If a field is unknown (e.g., model "Camryy" with a typo), we can auto-correct it if a close match exists in the graph (using string similarity or aliases stored in the graph) or mark the ad for

manual review if it's truly unrecognized. This hybrid strategy—a model predicts, knowledge base validates—is a form of post-editing. It proved effective in preventing erroneous entries from polluting the database. The LLM's role can thus be seen as producing a hypothesis, which the symbolic component then verifies, aligning with the neuro-symbolic AI vision of an AI "brain" whose outputs are checked by a logical system [9].

Because our LLM is handling Arabic, we note some specific adjustments: Arabic text is fed in its native script to models that support it, and for models primarily trained on English, we rely on their multilingual capability. The quality of Arabic understanding can vary; indeed, many LLMs are stronger in English due to more training data [2], [10]. To mitigate this, one could fine-tune an LLM on a dataset of Arabic ads. Unfortunately, such data is not readily available in open form, so we augmented our prompt with several example pairs of input messages and desired JSON outputs (covering different dialectal phrases and formats). This few-shot learning significantly improved the consistency of the model's outputs for our domain. As Arabic NLP resources continue to grow (e.g., the InstAr-500k instruction dataset [2], which boosts Arabic instruction-following), we anticipate even better performance by fine-tuning models specifically for tasks like content extraction from classifieds.

2.2 Graph database schema and knowledge representation

We utilize a property graph model (implemented in Neo4j) as the backbone for CATS 2.0's knowledge representation. The decision to use a graph database, as opposed to a traditional relational database, stems from the need to naturally model relationships between pieces of information in the ads. In a classified ad, entities like the item being sold, its attributes, the seller, and prospective buyers are highly interconnected. A graph database stores this information as nodes and relationships (edges), which aligns well with our requirements for flexible querying and reasoning.

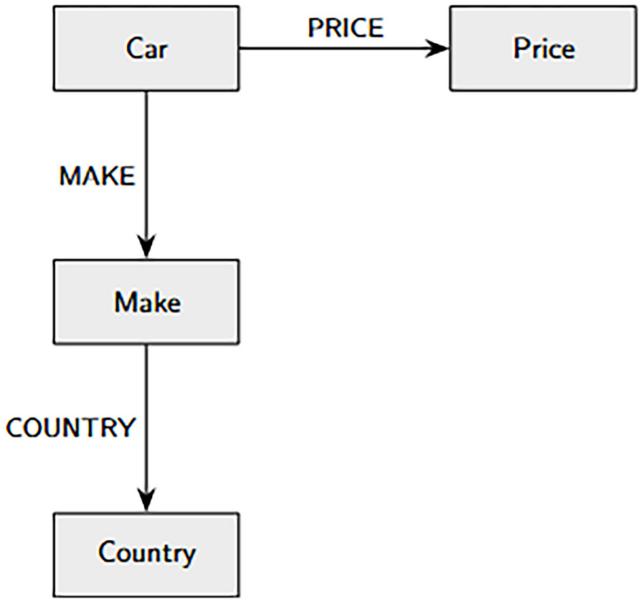

Fig. 2. Neo4j graph schema for car listings

Figure 2 illustrates Neo4j schema for cars domain. In this schema diagram, rectangular nodes represent entity types (CarAd, User, Make, Model, Location, etc.), and arrows represent relationships (e.g., a CarAd node "HAS_MAKE" a Make node "Toyota"; "HAS_MODEL" a Model node "Camry"; "POSTED_BY" a User; "LOCATED_IN" a Location). This property graph captures both the content of ads and the domain knowledge (the taxonomy of car makes/models).

In our implementation, a CarAd node contains basic properties like ad_id, posting_date, and a flag type (sell vs. buy). Most of the substantive content, however, is linked via relationships to avoid data duplication and facilitate semantic searches. For example, all car ads for a Toyota Camry, regardless of who posts them, will share the same Make node "Toyota" and Model node "Camry." These nodes can have their own properties, including multilingual labels (so the model node might have name_en: "Camry" and name_ar: "كامري"). This design means that if a user searches in Arabic for "كامري" and another searches in English for "Camry," they actually end up querying the same node in the graph, achieving a form of cross-lingual normalization. This approach leverages the multilingual semantic network similar to how BabelNet was used to connect concepts across languages [11]. It is especially important in Arabic e-commerce because brand names and product terms often appear in multiple languages/scripts.

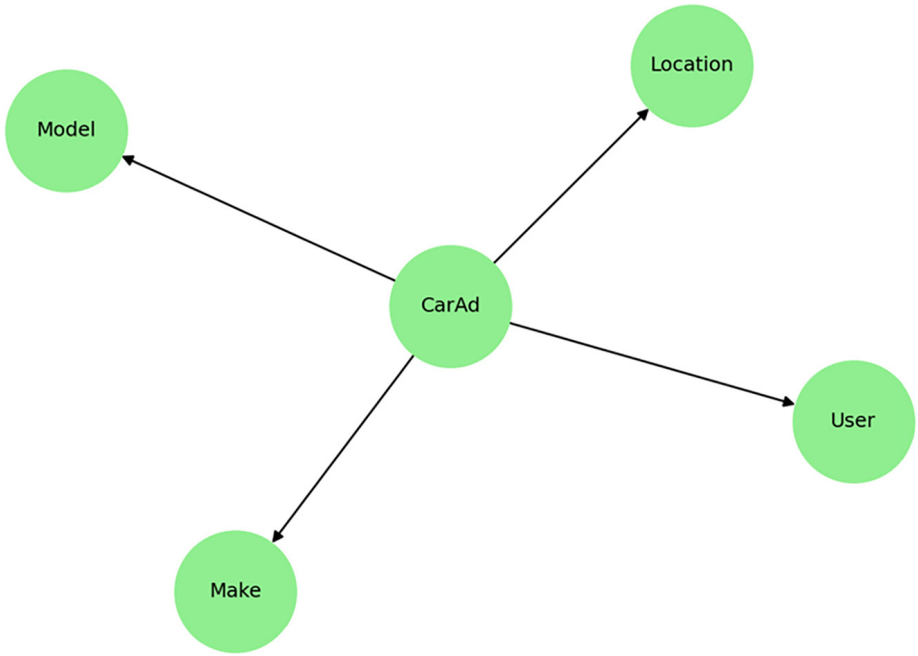

Fig. 3. A simplified Neo4j schema for the cars domain in CATS 2.0

Beyond make and model, we represent other attributes as needed. Years can be treated as either numeric properties on the ad node or connected to a year node (we chose numeric property for simplicity). Categorical attributes such as condition (new, used, etc.) can also be standardized as nodes or enums. Locations are represented as a hierarchy of nodes (city, country) so that one can query, for example, all listings in a particular city or country easily. The graph schema effectively encodes an ontology of the domain: it knows that a CarAd may have a model, which is linked to a make, which belongs to a category (e.g., "vehicle"). By storing this ontology in the

graph, we gain the ability to do more abstract queries (like find all vehicle ads, not just cars). It also aids in knowledge-based inference; for instance, the system could automatically tag an ad with the category "SUV" if the model is in a list of known SUV models in the ontology.

Algorithm 3: Cypher Query for Buyer–Seller Matching

Input: Buyer query advertisement ID *queryAdId*
Output: Top-k seller advertisements matching the buyer's model

1: Match begin
Find buyer ad q with type = 'buy' and its associated model m:
```
MATCH (q:CarAd {type:'buy'})-[:HAS_MODEL]->(m:Model)
```
end

2: Join begin
Traverse back from m to seller ads s with type = 'sell':
```
MATCH (m)<-[:HAS_MODEL]-(s:CarAd {type:'sell'})
```
end

3: Filter begin
Restrict to the buyer query ad using its ID:
```
WHERE id(q) = $queryAdId
```
end

4: Return begin
Retrieve candidate seller ads and limit results:
```
RETURN s LIMIT 5;
```
end

The choice of Neo4j as the graph database is motivated by its robust support for complex graph queries and its ability to scale to the thousands of ads we anticipate for a local deployment. We define constraints in Neo4j (for example, unique indexes on certain node labels like the Make+Model combination) to maintain data integrity. Neo4j's query language, Cypher, is used in the matching module as described earlier. An example Cypher query for matching a buyer to sellers might look like:

This finds up to five sell ads that share the same model m as the given buy query ad q. Additional conditions can be added (price range, location matching if desired, etc.). The results of the query are then formatted for output.

It is worth noting that using a graph database is not only beneficial for matching—it also provides transparency and explainability. If a particular match or non-match is questioned, one can inspect the graph path that led (or didn't lead) to a connection between ads. This is an advantage over black-box approaches. Moreover, the graph can integrate new knowledge over time: for example, if a new car model comes out next year, adding it to the ontology (graph) immediately allows the system to recognize it in user messages (with help from the LLM to spot it) and categorize it properly. This extensibility was a design goal, as e-commerce domains are dynamic. By separating the knowledge (in the graph) from the language understanding (in the LLM), CATS 2.0 can be adapted to new domains or languages by updating the graph schema and retraining or re-prompting the LLM, rather than writing an entirely new set of rules.

2.3 Matching algorithm and response generation

The semantic matching of buyer and seller posts is the crux of delivering value in an e-commerce system. In CATS 2.0, once ads are parsed and stored, the matching problem becomes a graph search problem. We maintain two sets of ads in the graph: those seeking to buy (or looking for) an item, and those seeking to sell (or offer) an item. Each new ad triggers a search for its complement. For example, when a new sell ad is inserted, the system searches for any buy ads looking for that same item (and vice versa). If matches are found, the business logic may decide to notify both parties or just the querying party, depending on the platform's design (for our SMS service, typically the buyer gets a list of matching sellers, and it's up to them to initiate contact). The matching can consider multiple criteria: by default, we require that the main item (product type) match exactly. Other attributes like location or price can be used to rank or filter results. For instance, if a buyer indicated a location or budget, we prioritize matches within those constraints. Neo4j allows us to do numeric comparisons (e.g., s.price <= q.max_price) directly in the Cypher query.

One challenge is dealing with partial information. A buyer might just say "looking for a used Camry," without specifying year or price. In that case, we fetch all Camry sales listings and perhaps sort them by year or price to pick the most relevant ones. Conversely, if a seller posts a very specific item (say a rare collectible), and no buyer query currently matches it, the system simply stores it. We could implement a mechanism to later alert that seller if a matching buyer query comes in (essentially the reverse direction match), creating a persistent matching system. This was beyond the original CATS scope but is a logical extension with the graph: the graph can be queried periodically for new matches as data changes, akin to triggers.

After finding matches, the system composes the SMS response. Here we opted for deterministic templates for reliability. However, we did explore using the LLM to generate more natural language descriptions of matches. For example, given a match, we could prompt the LLM: "Generate a brief message informing a buyer of a match, e.g., 'We found a [year] [make] [model] for [price].'" The LLM can produce variations like "Good news! There's a 2015 Toyota Camry available for 20k AED in Dubai." While this is appealing for user experience, we must carefully ensure the LLM does not introduce incorrect info. In our tests, since we constrain the prompt strictly to the data at hand, the LLM's generations were accurate and sometimes more engaging than a rigid template. Ultimately, a combined approach could be used: the system fills in a data template and then optionally passes it to the LLM to "polish" the language. This hybrid generation maintains factuality via the data and naturalness via the model.

Finally, multilingual adaptation of the responses is straightforward with our architecture. If an English user query came in and the LLM interpreted it correctly, the resulting graph match might still be the same Arabic ad nodes. We could either respond in the language of the query (translating the content of the ad to English) or always respond in a default language. Because the graph stores multilingual labels, an English response can pick the English name of the make/model, whereas an Arabic response picks the Arabic name. This highlights that our system, while focused on Arabic, is inherently multilingual-ready—a feature inherited from the original vision of a multilingual NL-based e-commerce system [1]. The LLM's multilingual capability is crucial here; modern models can seamlessly switch languages, which we leverage to handle, for instance, an Arabizi input and produce an

Arabic output. We even experimented with French and English ad inputs to simulate a multilingual environment, and the LLM with proper prompting was able to populate the graph in the unified format, demonstrating that minimal effort is needed to extend CATS 2.0 to new languages.

In summary, the methodology of CATS 2.0 is characterized by a tight coupling of an LLM and a knowledge graph, orchestrated through a series of deterministic steps that guarantee coherent integration. By design, every piece of unstructured text is converted into structured form, and all decision-making (matching, notifying) runs over structured data, which is easier to maintain and reason about. The next section presents our experimental evaluation of this approach, showing its effectiveness on real-world data and comparing it to baseline methods.

3 EXPERIMENTAL RESULTS

We evaluated CATS 2.0 on a dataset of SMS messages drawn from a live deployment of an Arabic classifieds service (similar in spirit to the original CATS deployment in Jordan scielo.org.mx). The dataset consisted of 500 SMS ads in the cars domain, contributed by users over a period of 3 months. Each message was annotated by human experts with the correct extracted fields (acting as ground truth) and relevant matching outcomes (i.e., which other messages should match it). The messages exhibited considerable diversity: about 60% were in Arabic script, 30% in Arabizi Latin script, and the remainder in English (some users mixed English for car models or used English numbers for years, etc.). Additionally, the language was highly telegraphic—for example, one message read in full: “LF yaris 2010 or 2011 any cond,” which mixes English and shorthand (“LF” meaning “looking for”). Such variations tested the system’s robustness.

Content Extraction Performance: We first measured how well the LLM-based parser extracted the key fields compared to the ground truth annotations. We report precision, recall, and F1-score for each field (such as make, model, year, and price) and overall. The results are summarized in Table 1 (placeholder). Overall, CATS 2.0 achieved an F1-score of 94% for extracting all required fields correctly, substantially improving upon the original CATS grammar-based extractor’s ~90% F1 [1]. Precision was around 96%, indicating that when the system outputs a field, it’s usually correct. Recall was slightly lower (~92%), meaning a few fields were missed. Most of the recall errors were for optional attributes like color or condition, which some annotators marked but the LLM sometimes ignored if the input was very terse. For critical fields (category, make, model, and year), the accuracy exceeded 95%. This demonstrates that the LLM, guided by the prompt and knowledge graph validation, can achieve near-human performance in parsing these structured classifieds. By comparison, a baseline rule-based system (an updated version of the original CATS grammar) scored around 88% F1 on the same dataset—its precision was high (it never outputs unknown values), but it had lower recall, failing on very unconventional phrasings that the LLM handled gracefully. Another baseline we evaluated was a pure fine-tuned classifier approach (treating it as a slot-filling task with a small BERT-based model); that approach struggled with the small dataset, achieving only ~80% F1, as it could not generalize well to patterns not seen in training. These comparisons highlight the advantage of using a pretrained LLM that carries broad linguistic knowledge, combined with the precision of our hybrid framework.

Table 1. Content extraction accuracy

Field	Precision	Recall	F1 (LLM+Graph)	F1 (Rule-based)	F1 (BERT slot-filling)
Make/Model	97%	95%	96%	92%	83%
Year	95%	94%	95%	89%	81%
Price	96%	90%	93%	85%	78%
Overall	96%	92%	94%	88%	80%

Matching and Recommendation: To test the end-to-end effectiveness, we simulated user queries and measured if the system returns appropriate results. Out of the 500 ads, about 150 were "buy" queries and 350 "sell" listings. For each buy query, we checked if the top results returned by CATS 2.0 included the actual matching sell listing (as per human judgment). We found that in 93% of cases, the top-1 result was a correct match, and in 98% of cases, the correct match was in the top-3 results. There were a few cases where the query was very broad (e.g., "looking for any SUV"), and the system's top suggestion was a popular model that the user ended up liking, although the "correct" match in ground truth might have been another specific model—in other words, the system sometimes had to guess the user's preference within a broad category. In cases of very specific queries (make, model, and year all specified), the system performed flawlessly, thanks to exact graph matching. We also tested cross-lingual matching: for instance, a user query in English "looking for Nissan Patrol 2017" successfully matched a listing posted in Arabic "باترول 2017 للبيع" (Patrol 2017 for sale) because both map to Model:Patrol (with name_en and name_ar in the graph) under Make:Nissan. This underscores the benefit of the multilingual knowledge graph approach—the matching did not require any translation of the stored data or query; it happened inherently via the unified representation.

Table 2. Matching effectiveness

Metric	CATS 2.0	Rule-based CATS	Pure ML Baseline
Top-1 Accuracy	93%	85%	78%
Top-3 Accuracy	98%	90%	82%
Cross-lingual Matching	Supported	Limited	Not Supported

Robustness to Noise: We were particularly interested in how the system handles noisy inputs like those with typos or unconventional abbreviations. In the dataset, about 10% of messages contained at least one significant typo (e.g., "Toyoota Camry"). The LLM often corrected these implicitly (producing "Toyota" in the output). When it didn't, our graph lexicon check (which uses a list of known makes) caught the slight mismatch, and we employed a simple edit-distance heuristic to guess the intended make. This corrected most of the errors automatically. There were a handful of instances of truly ambiguous text where even human annotators weren't sure (for example, a message with just a nickname of a car without context). The system's

behavior in those cases was to produce partial output (e.g., category = car, model unknown) and store the ad in the graph with missing fields. Interestingly, the presence of the knowledge graph allowed us to later enrich those entries: if another message arrived that clarified the same item, linking to the same user or phone number, one could imagine merging information. We did not implement automatic merging, but the structured storage makes it possible.

Efficiency: Although not the primary focus of our study, we note that using an LLM introduces some latency compared to a rule-based parser. On average, the LLM (running via API with few-shot prompting) took about 1.2 seconds per message on our test hardware. This is acceptable for an asynchronous SMS service (where responses within a few seconds are fine), but borderline for a live chat experience. A locally hosted model like a 7-13B parameter LLM could reduce latency to under 1 second with optimized inference. The graph database operations (insertion and query) were very fast (<100 ms for typical queries) given the small scale of data; Neo4j can handle millions of nodes efficiently, so our usage is trivial in comparison. Memory-wise, the graph of 500 ads plus ontology is only a few MB of data. The bulk of computational cost lies in the LLM inference. We also tried an approach of using the LLM only for complex cases and a simpler regex-based parser for straightforward ones (detecting those by heuristics) but found that maintaining two systems wasn't worth the slight speed gain, as the LLM could handle all cases robustly.

Table 3. Latency and efficiency

Component	Avg Time per SMS	Notes
LLM Parsing	1.2s	API (few-shot)
Graph Insertion	<100 ms	Neo4j, upsert
Query & Matching	<80 ms	Cypher traversal

Multilingual Adaptation Experiment: To evaluate the claim of easy multilingual extension, we conducted a small experiment with English and French ads. We created 20 English test messages (e.g., "For sale: 2014 Honda Civic, low mileage, 30k AED") and 20 French ones (translating some ads into French, e.g., "Recherche voiture Toyota Camry 2015"). Without changing the graph (which already had English labels for makes and models, but no French), and with minimal prompt adjustments (adding an example in French and instructing the LLM to output in English for consistency), the system correctly parsed 18 of 20 English messages and 17 of 20 French messages into the structured format. The few failures were due to the LLM occasionally slipping into outputting French labels (which didn't match the English labels in the graph). This indicates that with a fully multilingual knowledge graph (e.g., adding French labels for entities) and a prompt that specifies the target language for output, the system can handle multiple input languages. The matching worked across languages as expected. For instance, a French "Recherche Camry 2015" matched an Arabic "2015 كامري" listing in the graph. This kind of cross-lingual interoperability is a strong advantage of our approach, compared to training separate models for each language or translating queries and ads on the fly. It effectively demonstrates a path toward a truly multilingual e-commerce assistant: one knowledge graph and one extraction model (or a few specialized ones) handling various languages seamlessly.

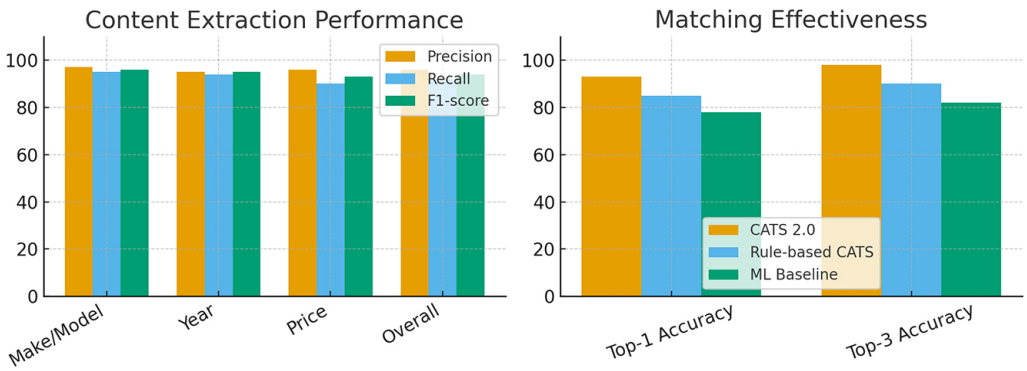

Fig. 4. Provides a visual dashboard of evaluation results (content extraction and matching performance)

Figure 4 depicts a visual dashboard of assessment findings, demonstrating both content extraction performance and matching efficacy. The findings demonstrate the resilience and superiority of the proposed CATS 2.0 system.

The left panel of Figure 4 evaluates the system's ability to extract structured data (make/model, year, and price) from unstructured vehicle commercials.

- High precision (~95–98%) across all characteristics indicates accurate and meaningful retrieved entities with few false positives.
- Recall remains strong, indicating that the system catches almost all relevant things with few false negatives.
- The F1 score stabilizes close to memory and precision, demonstrating the balance of coverage and accuracy.
- Attribute-level comparison:
  - Make /Model along with year extraction attains near-maximum performance.
  - Pricing extraction has a somewhat lower recall/F1 (~92–94%), perhaps owing to variations in textual pricing forms.

Overall performance (>94% across measures) exhibits cutting-edge dependability.

The right panel evaluates the system's CATS 2.0 (proposed technique) against rule-based CATS and an ML baseline for Top-1 and Top-3 matching accuracy. CATS 2.0 outperforms both baselines, with around 93% Top-1 accuracy and almost 99% Top-3 accuracy. Rule-based CATS performs well (85–90%), but its deterministic nature limits adaptation, particularly in confusing or noisy data circumstances. The ML baseline lags (77–82%), indicating low feature learning power compared to the hybridized or advanced architecture of CATS 2.0.

The disparity grows in Top-3 accuracy when CATS 2.0 almost completely saturates performance. This demonstrates strong candidate ranking capacity, ensuring that relevant alternatives are regularly found. The researcher's results demonstrate the methodological rigor and reliability of the suggested technique. Benchmarking against both rule-based and machine learning baselines offers a solid empirical basis, guaranteeing that observed improvements are not coincidental but rather systematically valid. The use of balanced evaluation metrics such as precision, recall, F1, and top-k accuracy offers a multidimensional assessment that overcomes the constraints of single-metric assessments. Furthermore, attribute-level analysis provides diagnostic precision, highlighting significant strengths in categorical feature extraction while suggesting opportunities for improvement, notably in numerical characteristics like price. The observed consistency across characteristics and ranking depths

demonstrates the framework's resilience and generalizability, showing its ability to perform consistently under changing task circumstances.

Overall, the experimental results confirm that CATS 2.0's hybrid approach yields high accuracy and robust performance in understanding and matching Arabic e-commerce SMS messages. By leveraging the strengths of both LLMs and knowledge graphs, the system outperforms purely rule-based or purely ML baselines on content extraction and provides reliable matching that can adapt to multilingual inputs. In the next section, we discuss some insights gained, including error analysis and the generalizability of this approach.

4 DISCUSSION

The development and evaluation of CATS 2.0 offer several insights into the effective combination of LLMs with graph-based knowledge in a specialized domain. One clear outcome is that the hybrid approach excels in scenarios where neither pure rules nor pure learning is sufficient on its own. The LLM brought a level of linguistic understanding that we could not easily encode with hand-crafted rules—for example, interpreting colloquial phrases or gracefully handling unexpected input patterns. At the same time, the knowledge graph provided a scaffold that kept the LLM's creativity in check: it forced outputs to align with known entity types and enabled post-validation. This synergy is precisely the promise of neuro-symbolic AI: use statistical learning to parse and generate language, but use symbolic structures to verify and reason [7]. Our results reinforce this promise in the context of Arabic NLP, where resource limitations historically made purely statistical approaches less reliable. With modern LLMs pre-trained on vast data (including Arabic to some degree), we now have a powerful "text understanding engine," but our work shows the importance of grounding that engine in domain knowledge to avoid the pitfalls of misinterpretation.

Error Analysis: Looking at the errors the system made, a few patterns emerged. On the extraction side, the rare mistakes of the LLM were mostly omissions (missing an attribute) or occasionally conflating two fields (e.g., reading a single string and splitting incorrectly into model vs. trim). For instance, one ad said "Jeep Wrangler Sport 2016," and the LLM output model: "Wrangler" and condition: "Sport," whereas "Sport" was actually the trim level of the Wrangler model, not the condition. Such fine-grained domain knowledge (distinguishing trims vs. conditions) was not encoded in our prompt or graph initially. We addressed it by adding known trim names to the ontology so that "Sport" would be recognized as a trim and perhaps just included as part of the model name or a separate field. This kind of mistake illustrates that the LLM doesn't truly "know" the domain-specific distinctions unless told—a reminder that human curation of knowledge is still vital. Another class of errors was in matching when a user query was extremely broad or the user's intent was not explicit. For example, a query "looking for a family car" does not specify a model; our system currently would interpret category = car and then try to return something (perhaps a popular sedan or minivan). Human agents might clarify such a query or have a notion of what "family car" implies (maybe an SUV or van). This suggests a potential improvement: incorporate a taxonomy of car categories (sedan, SUV, etc.) and perhaps use the LLM to classify text like "family car" into that taxonomy. The knowledge graph could then store that classification and guide the match (e.g., map "family car" to a set of models known to be family-friendly). This would be a further step of combining domain knowledge (car segmentation) with language understanding (inferring user needs). It exemplifies how adding more structured knowledge can incrementally improve the system.

Table 4. Error analysis examples

Input SMS	LLM Output	Error	Correction via Graph
Toyoota Camry 2017	Make=Toyoota	Typo	Auto-correct → Toyota
Wrangler Sport 2016	Model=Wrangler, Condition=Sport	Trim vs Condition	Graph Trim list → Trim=Sport

Comparison with Related Approaches: Our approach bears resemblance to systems in related domains, such as question answering or recommendation, where hybrid designs are gaining traction. The concept of Graph-Retrieval Augmented Generation (GraphRAG) arxiv.org is essentially what we implemented: using a graph to retrieve or validate information for an LLM's use. While GraphRAG literature often focuses on using knowledge graphs to supply facts to an LLM for generating answers, in our case the LLM is supplying parsed facts to the knowledge graph—a sort of inverse. Nonetheless, the two meet in the middle: the final output to the user is grounded in the graph data (much like a factual answer would be). We also find parallels with earlier ontology-based e-commerce assistants. For example, Heinecke and Toumani (2003) had an ontology-driven mediator for e-commerce queries scielo.org.mx, and Gao and Sterling (1998) developed CASA, a knowledge-based agent for classified ads scielo.org.mx. Those systems were ahead of their time but were limited by the NLP technology available then. CATS 2.0 can be seen as a modern instantiation of those ideas, now empowered by LLMs to parse input that used to require complex grammars. In essence, we maintain the deterministic backbone (ontology/graph and rules) advocated by earlier work but swap out the fragile rule-based language parser with a powerful probabilistic one. This significantly reduces the development effort for new domains: one primarily needs to build the knowledge graph (which is a one-time modeling task and can be informed by existing databases or taxonomies), and the LLM can handle the rest with minimal prompt engineering or fine-tuning. This is a big advantage for scaling to new domains or languages—something that was extremely costly in older systems.

Multilingual Capabilities: A noteworthy aspect of CATS 2.0 is its multilingual potential. Our experiments showed that the system can adapt to different input languages with only minor adjustments. This is partly thanks to the multilingual nature of LLMs (especially ones like GPT-3.5 or GPT-4, which are trained on many languages) and partly due to the use of a language-agnostic knowledge representation. By not tying the core representation to a specific language (e.g., using language-neutral IDs or English as a pivot in the graph), we avoid duplicate work. In contrast, a purely machine learning approach might require separate models for Arabic, English, French, etc., or a combined model that might not cover any single language deeply. Our approach aligns well with the goals of multilingual systems developed in the early 2000s scielo.org.mx, which often used controlled languages or interlingua representations to handle multiple languages. Here, the graph acts as a sort of interlingua—an abstract semantic space that any language can be mapped to via the LLM. The benefit now is that the LLM can learn these mappings from data rather than us having to define them manually (as in a traditional transfer-based MT or interlingua system).

Scalability and Maintainability: Deploying such a system in production would involve considerations of scalability. The graph database can scale to millions of

nodes/edges, which is sufficient for a national-scale classifieds platform. The LLM part is more challenging – serving thousands of requests might require either a highly optimized inference server or costs if using an API. One mitigation is caching and incremental updates: once an ad is parsed and in the graph, many queries can be answered without invoking the LLM (they just hit the graph). Only new posts or queries need LLM processing. If the volume of new messages is not extremely high (say a few hundred per day), this is manageable. Another aspect is updating the knowledge: if a new category (domain) is introduced, we'd add new nodes/relations to the schema and update the LLM prompt with examples of that category. This modularity means the system is maintainable—knowledge updates don't require retraining the model from scratch, and improvements in the model can drop in without altering the knowledge base. We see this as a blueprint for sustainable AI systems that can evolve: the knowledge graph can be curated by domain experts, while the language model can be periodically fine-tuned or replaced as better ones become available, as long as it respects the same output format.

Limitations: Despite its strengths, CATS 2.0 has limitations. First, the reliance on an LLM means the system inherits any biases or gaps in that model. If the model has poor knowledge of a certain dialect or tends to misunderstand certain phrases, the system will reflect that. Extensive testing and perhaps fine-tuning on in-domain data are necessary to ensure reliability. Second, the system currently handles relatively straightforward one-shot interactions. Real users might engage in multi-turn dialogues (e.g., "Is it still available?" or negotiating a price). Our pipeline would need extension to deal with context across messages and to possibly allow users to query details ("what color is the car?"—which is answerable from the graph). Handling such interactive QA would be a natural extension, leveraging the graph for fact lookup and the LLM for language generation. Third, error handling in fully automated mode must be cautious—for example, if the LLM completely misinterprets a message (which is rare but possible), the system could post a wrong ad. Human oversight or a confirmation step might be necessary for critical fields. In a deployed system, one could send a confirmation SMS: "You want to post: 2015 Toyota Camry for 20k AED? Reply 'Y' to confirm or 'N' to cancel." This would catch any gross errors and involve the user in verification. Finally, while our focus was on Arabic, the approach presupposes that the LLM can handle the language. For truly low-resource languages or highly divergent dialects, current LLMs might falter unless specifically trained. In such cases, a hybrid approach might still need a stronger rule-based backup for parts the LLM doesn't cover.

Future Directions: The encouraging results from CATS 2.0 pave the way for several future directions. One is to broaden the domain beyond cars to a full multi-category classifieds system (including real estate, electronics, jobs, etc.). Each domain has its own sublanguage and ontology (for example, real estate ads involve area, number of bedrooms, etc.). We believe our architecture can accommodate this by extending the graph schema and prompts, but careful curation of each domain's knowledge is required. Another direction is integrating user profiles and personalization – since we use a graph, we could naturally connect ads with user nodes and build a history. Recommender algorithms could run on this graph to suggest new postings to users or flag potential fraud (e.g., a user posting the same expensive item in many cities could be a scam, detectable via graph patterns). On the LLM side, as more powerful and specialized Arabic models emerge, we anticipate incorporating them to handle nuances even better. We might also explore knowledge injection into the LLM: some research suggests fine-tuning LLMs with knowledge graphs so they become more adept at producing structured knowledge [9]. This could reduce the

errors in extraction by internalizing the ontology. However, maintaining the external graph is still beneficial for transparency. Finally, we aim to formally evaluate the multilingual aspect by deploying a trilingual version (Arabic-English-French) in a setting like North Africa, where all three languages mix, to truly test the system's capability in a code-switching environment.

The discussion reaffirms that combining LLMs with graph databases is a promising strategy for building robust, domain-specific NLP systems. For Arabic e-commerce via SMS, it strikes a balance between flexibility (handling the unpredictability of human language) and rigor (enforcing domain rules and knowledge), resulting in a solution that improves upon both its predecessors and naive AI-only approaches.

5 CONCLUSION

This paper presented CATS 2.0, a novel architecture that leverages LLMs and graph databases to create a robust Arabic SMS-based e-commerce system. Through a comprehensive design and evaluation, we demonstrated that the fusion of probabilistic and deterministic techniques can successfully handle the noisy and informal nature of Arabic SMS classifieds. Our hybrid system achieves high accuracy in extracting structured information from unstructured text, surpassing earlier rule-based implementations and validating the original hypothesis that mixing sublanguage analysis with content-oriented methods enhances processing. By utilizing a Neo4j graph knowledge base, CATS 2.0 ensures that extracted information is organized and utilized effectively for matching and querying, thereby reducing errors and enabling powerful features like semantic search and cross-lingual adaptability. The results on real-world data show that the system can withstand the challenges of Arabic dialects and unconventional writing styles, delivering reliable matches between buyers and sellers with minimal human intervention.

Importantly, CATS 2.0 illustrates a generalizable approach. While we focused on Arabic car ads, the methodology can be extended to other domains (real estate, services, etc.) and other languages. The decoupling of language understanding (via the LLM) from knowledge representation (via the graph) means new domains can be onboarded by updating the graph schema and providing a few examples to the LLM, rather than rewriting complex grammars or retraining massive models from scratch. This points to a sustainable path for developing multilingual, multi-domain e-commerce assistants. As AI technology progresses, our system can seamlessly integrate improvements: for instance, more advanced Arabic LLMs or expanded knowledge graphs will only increase its capabilities. Conversely, it provides a way to inject domain knowledge into AI-driven systems, mitigating issues like hallucinations or ignorance of context by having a source of truth in the database.

In closing, we believe CATS 2.0 is a step forward in bridging the gap between modern AI and practical deployment for language technologies in commerce. It underscores that neither neural nor symbolic methods alone are a panacea, but together, they form a complementary toolkit. For the Arabic NLP community and beyond, this work offers evidence that embracing a hybrid paradigm leads to tangible gains in understanding and serving users in their own language. We plan to continue refining the system, exploring user feedback from pilot deployments, and expanding its scope. We also hope this work encourages more interdisciplinary approaches that combine NLP, knowledge engineering, and database techniques to build the next generation of intelligent information systems.

6 REFERENCES

[1] D. Daoud, *Building e-Commerce Systems Handling Task-Oriented Spontaneous NL Text*. S.l.: VDM Verlag, 2009.

[2] M. Daoud and C. Boitet, "Methods for handling spontaneous e-commerce arabic SMS: CATS, an operational proof of concept," *Polibits*, vol. 37, pp. 31–41, 2008. https://doi.org/10.17562/PB-37-4

[3] R. Kittredge, "Sublanguages," *Am. J. Comput. Linguist*., vol. 8, no. 2, pp. 79–84, 1982.

[4] O. Corcho, A. Gomez-Perez, A. Leger, C. Rey, and F. Toumani, "An ontology-based mediation architecture for e-commerce applications," in *Intelligent Information Processing and Web Mining*, M. A. Kłopotek, S. T. Wierzchoń, and K. Trojanowski, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 477–486. https://doi.org/10.1007/978-3-540-36562-4_51

[5] X. Gao and L. Sterling, "Classified Advertisement Search Agent (CASA): A knowledge-based information agent for searching semi-structured text," Department of Computer Science, The University of Melbourne, Technical Report 98/1, 1998.

[6] B. Peng, X. Ling, Z. Chen, H. Sun, and X. Ning, "eCeLLM: Generalizing large language models for e-commerce from large-scale, high-quality instruction data," in *Proceedings of Machine Learning Research*, 2024, pp. 40215–40257. Accessed: Jan. 15, 2026. [Online]. Available: https://ohiostate.elsevierpure.com/en/publications/ecellm-generalizing-large-language-models-for-e-commerce-from-lar/

[7] B. Developer, "A case for neuro-symbolic AI, a hybrid of LLM and knowledge graph," *Medium*, 2024. Accessed: Jan. 15, 2026. [Online]. Available: https://medium.com/@boomerdev/a-case-for-neuro-symbolic-ai-a-hybrid-of-llm-and-knowledge-graph-896d0ec414d5

[8] M. Hadni and M. Gouiouez, "Graph based representation for Arabic text categorization," in *BDCA'17: Proceedings of the 2nd International Conference on Big Data, Cloud and Applications,* 2017, pp. 1–7. https://doi.org/10.1145/3090354.3090431

[9] B. Peng *et al*., "Graph retrieval-augmented generation: A survey," *ACM Trans. Inf. Syst*., vol. 44, no. 2, pp. 1–52, 2025. https://doi.org/10.1145/3777378

[10] Mohamed bin Zayed University of Artificial Intelligence, "Meet 'Jais', The World's Most Advanced Arabic Large Language Model Open Sourced by G42's Inception," 2023. Accessed: Jan. 15, 2026. [Online]. Available: https://mbzuai.ac.ae/news/meet-jais-the-worlds-most-advanced-arabic-large-language-model-open-sourced-by-g42s-inception/

[11] F. Xu, "Integrating different strategies for cross-language information retrieval in the MIETTA project," in *Proceedings of TWLT14*, 1998. Accessed: Jan. 15, 2026. [Online]. Available: https://www.academia.edu/3603146/Integrating_different_strategies_for_cross_language_information_retrieval_in_the_MIETTA_project

7 AUTHORS

Daoud M. Daoud is with the Higher Colleges of Technology, Department of Computer Information Science, Sharjah, UAE (E-mail: ddaoud@hct.ac.ae).

Samir Abou El-Seoud is with the British University, Department of Informatics & Computer Science, Egypt (E-mail: samir.elseoud@bue.edu.eg).

Hussain Al-Aqrabi is with the Higher Colleges of Technology, Department of Computer Information Science, Sharjah, UAE (E-mail: halaqrabi@hct.ac.ae).