Knowledge and Language 2025/2026
Literature Review

Miguel Castela, uc2022212972
DEI, Universidade de Coimbra
uc2022212972@student.uc.pt

Miguel Martins, uc2022213951
DEI, Universidade de Coimbra
uc2022213951@student.uc.pt

I. Abstract

This literature review explores the integration of Knowledge Graphs (KGs) and Retrieval-Augmented Generation (RAG) techniques for domain-specific question answering. The aim of this review is to understand the technologies and state of the art methods involved in the area of building accurate context and specification specific goal driven chatbots using predefined data and Knowledge Representation. We will use RecipeRAG [1] a KG-driven recipe generation framework, as reference for the methodologies to build a food chatbot. Definitions and concepts are supported by the class materials on Knowledge Graphs, Vocabularies and Ontologies, as well as recent literature on Knowledge-Aware tasks [2]. We discuss how RDF/OWL-based KGs can represent structured knowledge from datasets, how NLP techniques enable knowledge retrieval via SPARQL and Knowledge Graph Embeddings (KGEs), and how RAG architectures combine symbolic retrieval with neural generation to improve factuality.

II. Connections with the Topics Covered by the Course

Large Language Models (LLMs) demonstrate strong generalization but fail in domain-specific reasoning due to three main limitations: "a failure to comprehend the question due to lack of context, insufficient knowledge to respond accurately, or an inability to recall specific facts" [2]. This motivates the integration of RDF/OWL-based graphs with LLMs and RAG for inference, to ensure contextual understanding and factual accuracy. RDF can be serialized in formats such as N-TRIPLES, XML, JSON-LD, or Turtle [3], ensuring interoperability across systems . Vocabularies define terms, relationships and ontologies define classes, hierarchies, and property restrictions. OWL (Web Ontology Language) can be used with RDF to express richer logical constraints, such as domain/range restrictions and class relations [3].
This symbolic foundation serves as the base for the three "knowledge-aware" dimensions of KG-augmented NLP methods outlined in [2]:

- Knowledge-Aware Inference: Knowledge graphs enhance inference in LLMs by providing structured, factual context that improves reasoning, retrieval, and generation. They serve as external symbolic memory, allowing models to access curated, semantically linked knowledge instead of relying solely on internal parameters. Through structured queries or textualized triples, KGs support more accurate and interpretable reasoning, enabling LLMs to decompose complex tasks, ground their outputs in verifiable facts, and reduce hallucinations during text generation.
- Knowledge-Aware Training: During both pre-training and fine-tuning, knowledge graphs inject structured factual information into language models to strengthen their semantic understanding and factual accuracy. Integrating KG-based entities and relations enriches training output and improves representations of real-world concepts. Fine-tuning with KG data helps adapt models to specific domains or new knowledge. In this way, KGs reinforce the model's internal knowledge base and ensure that learned representations remain consistent with factual, structured information.
- Knowledge-Aware Validation: Knowledge graphs act as symbolic verification tools that validate and explain the outputs of language models. By providing a structured reference of entities and relations, KGs enable automated fact-checking and logical consistency checks, ensuring that generated text aligns with verified knowledge. This symbolic validation increases model transparency and reliability, bridging neural text generation with interpretable, fact-grounded reasoning.

This emphasizes the distinction between symbolic reasoning (FOL-based) [3] and neural reasoning, emphasizing that symbolic models provide factual precision, while neural models capture language generation variabilty.

III. RDFS/OWL Knowledge Graphs for representing knowledge from datasets

In RDF, knowledge is represented through triples, for example:

```
<Bacalhau_a_Brás> <hasIngredient> <Egg>
```

Each triple expresses a simple `<subject> <predicate> <object>` relationship [3], which can be enriched using RDFS with class and property hierarchies, domain and range constraints, and subclass relationships, allowing for the representation of more structured and semantically rich knowledge.

OWL further extends RDF/RDFS with an ontology for logical reasoning, such as cardinality restrictions, equivalence, disjointness, and complex class expressions, enabling more sophisticated inference over the data.

IV. Knowledge Retrieval via NLP

Knowledge retrieval connects natural language input to KG representations. SPARQL is the declarative query language for RDF graphs [3], enabling semantic retrieval using triples and filters. SPARQL query construction can be made in 3 different ways:

- The user, using the Chatbot, needs to input the SPARQL query directly or use a form-based interface that generates the query based on user selections.
- LLMs can be fine-tuned or prompted to generate SPARQL queries from natural language questions, using few-shot examples or templates to guide the generation.
- NLP tools like semantic parsers can convert natural language into SPARQL queries by identifying entities, intent, relations, restrictions for the query structure.

NLP-based query interpretation is the lighter and more deterministic approach. For this, Natural Language Processing interfaces require extracting entities and intents (via tools like spaCy [4] or BERT [5]) and mapping them to KG entities (URIs [3]). BERT can be fine-tuned to identify the user's intent, which enables the system to correctly interpret what the user wants even when questions are phrased in different ways. BERT's [5] bidirectional transformer architecture provides deep contextual understanding, improving both the identification of user intents and entity disambiguation in complex queries. spaCy [4] is used to extract entities and linguistic features from user queries. Its an efficient, rule and model-based pipeline that enables fast and accurate identification of nouns, verbs, entities, and syntactic relations, helping map natural language input to knowledge graph entities and improving the system's overall understanding of user intents.

V. Retrieval-Augmented Generation (RAG)

Lewis et al. (2020) [6] introduced the Retrieval-Augmented Generation (RAG) model, a hybrid architecture that combines parametric and non-parametric memory systems to improve knowledge-intensive natural language generation. RAG models "use the input sequence x to retrieve text documents z and use them as additional context when generating the target sequence y." The parametric memory is represented by a pre-trained seq2seq model [7] (a transformer base model [8]), specifically BART [9], which stores implicit knowledge within its parameters. However, such knowledge is static, cannot be easily updated or interpreted, and may lead to factual inaccuracies or "hallucinations." To address these issues, RAG incorporates a non-parametric memory which is a dense vector index of Wikipedia built using the Dense Passage Retriever (DPR) [10]. Each passage is encoded as a dense vector, allowing the model to retrieve the top-k most relevant documents for a given query through Maximum Inner Product Search (MIPS):

$$p_\eta(z \mid x) \propto \exp\left(d(z)^\top q(x)\right)$$

Where the retrieval component of RAG, denoted $p_\eta(z \mid x)$, represents the probability of retrieving a document z given a query x. Each document z is encoded into a dense vector representation $d(z)$, while the query x is encoded into a corresponding dense vector $q(x)$. and retrieves the documents that maximize this inner product:

$$\arg\max_z d(z)^\top q(x).$$

This operation identifies the documents whose embeddings are most similar to the query embedding, ensuring that the most relevant passages are selected for the generator to use during response generation. During generation, RAG first uses DPR [10] to retrieve relevant documents and then conditions the generator [9] on both the input and the retrieved passages, generating more factual and interpretable text while maintaining the fluency of neural models

Lewis et al. [6] proposed two variants of the Retrieval-Augmented Generation (RAG) model: *RAG-Sequence* and *RAG-Token*. *RAG-Sequence* conditions generation on a single retrieved document for the entire output, while *RAG-Token* allows token-level retrieval, enabling each generated token to draw information from different documents. For each token y_i, the model computes a weighted sum over the probabilities conditioned on each of the top-k retrieved documents, providing finer-grained factual grounding. Unlike language models that rely solely on fixed parametric memory, RAG can provide explicit evidence for its outputs through retrieved passages while reducing hallucinations. Experiments [6] demonstrated better results across knowledge-intensive tasks, outperforming both parametric-only models and purely extractive retrieval systems. Moreover, RAG models produced more specific, diverse, and factually accurate generations.

VI. Knowledge retrieval via KGEs

RecipeRAG [1] builds upon this foundation, substituting text retrieval with KG-based retrieval and applying it to domain-specific recipe generation. KGE-based retrieval [1] significantly outperforms text-based retrieval when handling multiple user constraints. As noted: "Performance declined as the number of criteria increased... Nevertheless, the results underscore the effectiveness of knowledge graph embeddings (KGEs) in supporting multi-criteria recipe retrieval, while simultaneously revealing the limitations of purely text-based embeddings." KGEs map entities and relations from symbolic graphs into continuous vector spaces [11], enabling reasoning and retrieval; **TransE** [11] represents relations as translation vectors such that the tail entity is approximated by the head plus the relation ($\mathbf{h} + \mathbf{r} \approx \mathbf{t}$) and works well for 1-to-1 relations but struggles with complex patterns. For this, **RotatE** [12] models relations as rotations in the complex plane ($\mathbf{t} \approx \mathbf{h} \circ \mathbf{r}$) to naturally capture symmetry, antisymmetry, and inversion. **QuatE** [13] extends this idea to quaternions ($\mathbf{t} \approx \mathbf{h} \otimes \mathbf{r}$) for 4D rotations, allowing modeling of more complex relational patterns.

”We chose RotatE, the strongest-performing knowledge graph embedding (KGE) model, to bootstrap the recipe generation process. The generation pipeline operates as follows: first, the system retrieves the top-ranking recipes based on aggregated relevance scores. From these, the top 5 most relevant recipes are selected. These retrieved recipes, along with the original user criteria, are then fed into DeepSeek-R1-0528, which generates new, tailored recipe suggestions. By leveraging both the user's input and the 5 retrieved examples as context, the model produces a customized recipe.” This pipeline combines the retrieval mechanism of [6] with structured KG enrichment, turning symbolic graph knowledge into context for generative models.

VII. Conclusion and Future Work

The reviewed literature converges on the insight that combining symbolic and neural reasoning enables systems that are both factually grounded and contextually adapative. Empirical evidence from RAG [6] and RecipeRAG [1] demonstrates that knowledge graph–based retrieval significantly improves generation quality. The taxonomy presented in [2] elucidates how such hybrid architectures mitigate hallucinations through Knowledge-Aware Inference, Learning, and Validation.

References

[1] T. Loesch, E. Durmuş, and R. Celebi, “Reciperag: A knowledge graph-driven approach to personalized recipe retrieval and generation,” in *Proceedings of RAGE-KG 2025*, 2025.

[2] G. Agrawal, T. Kumarage, Z. Alghamdi, and H. Liu, “Can knowledge graphs reduce hallucinations in llms?: A survey,” in *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024, accepted paper in NAACL 2024; also available as arXiv:2311.07914 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2311.07914

[3] “Course slides,” 2025, course materials, PowerPoint slides.

[4] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, “spacy: Industrial-strength natural language processing in python,” https://zenodo.org/record/1215370, 2020, software release.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2019, submitted 11 Oct 2018 (v1); last revised 24 May 2019 (v2). [Online]. Available: https://doi.org/10.48550/arXiv.1810.04805

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *arXiv preprint arXiv:2005.11401*, 2020, accepted at NeurIPS 2020; v4. [Online]. Available: https://arxiv.org/abs/2005.11401

[7] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *arXiv preprint arXiv:1409.3215*, 2014. [Online]. Available: https://arxiv.org/abs/1409.3215

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017, submitted 12 Jun 2017 (v1); last revised 2 Aug 2023 (v7). [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1910.13461

[10] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, “Dense passage retrieval for open-domain question answering,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, eMNLP 2020; arXiv preprint. [Online]. Available: https://doi.org/10.48550/arXiv.2004.04906

[11] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2787–2795.

[12] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in *International Conference on Learning Representations (ICLR)*, 2019, accepted to ICLR 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1902.10197

[13] S. Zhang, Y. Tay, L. Yao, and Q. Liu, “Quaternion knowledge graph embeddings,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, accepted by NeurIPS 2019. [Online]. Available: https://arxiv.org/abs/1904.10281