Fact Verification in Knowledge Graphs Using LLMs

Farzad Shami
farzad.shami@studenti.unipd.it
Dept. of Information Engineering,
University of Padua
Padua, Italy

Stefano Marchesin
stefano.marchesin@unipd.it
Dept. of Information Engineering,
University of Padua
Padua, Italy

Gianmaria Silvello
gianmaria.silvello@unipd.it
Dept. of Information Engineering,
University of Padua
Padua, Italy

Abstract

Automated fact-checking systems often struggle with trustworthiness, as they lack transparency in their reasoning processes and fail to handle relationships in data. This work presents FactCheck, a fact verification system topped by a web platform that shows how Large Language Models (LLMs) can be collectively used to verify facts within Knowledge Graphs (KGs). While the underlying verification engine implements a system that combines Retrieval Augmented Generation (RAG) with an ensemble of LLMs to validate KG facts, the platform focuses on making the results of this complex process as transparent and accessible as possible. Users can explore how different models interpret the same evidence, compare their reasoning patterns, and understand the factors that lead to the final verification result. The platform supports technical users who want to analyze the model behavior and general users who need to verify whether the facts in the dataset are correct.

CCS Concepts

• **Information systems** → **Web searching and information discovery**; • **Search interfaces**;

Keywords

Fact Verification, Knowledge Graphs, Large Language Models, Retrieval Augmented Generation

ACM Reference Format:

Farzad Shami, Stefano Marchesin, and Gianmaria Silvello. 2025. Fact Verification in Knowledge Graphs Using LLMs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Knowledge Graphs (KGs) have become an indispensable component of modern information systems, playing an essential role in a wide range of applications, including search engines[11], recommendation systems [13], and question-answering platforms [1, 19]. KGs comprise knowledge that links entities through relationships to form a network that machines can analyze and interpret. However, the effectiveness and reliability of KGs depend on the accuracy of the individual pieces of information they contain – i.e., statements or facts represented as subject-predicate-object triples – and how these pieces are connected [6, 20].[1]

Auditing facts stored by KGs is a critical and time-consuming task [9, 18]. Indeed, to evaluate a claim conveyed by a specific triple, humans would need to search through potentially many sources to find supporting or conflicting evidence, assess the reliability of each source, and make a decision. This process can take assessors several minutes per claim [16]. Therefore, given the size of current large-scale KGs, comprising hundreds of millions to billions of facts, manual validation of the entire contents of a KG is unfeasible [9].

To overcome scalability issues, a standard approach consists of automated fact-checking systems. Traditional fact-checking methods are heavily based on probabilistic or rule-based techniques [2, 8]. While these methods are practical for commonly defined claims, they falter when asked to verify ambiguous statements or require extensive contextual interpretation [23]. In response to these limitations, the last decade has witnessed a growing adoption of fact-checking systems based on machine/deep learning solutions [14]. Notably, the recent Large Language Models (LLMs) have shown great promise in this area, given their extensive knowledge and robust reasoning capabilities [12, 17]. Nevertheless, current LLMs are also known to produce hallucinated and unfaithful responses especially for fact-based tasks [24], making their adoption as "black-boxes" a risky, non-transparent, and potentially harmful choice. Hence, there is a need for systems that can leverage the cutting-edge capabilities of LLMs, while ensuring a transparent, "white-box" solution for fact verification.

Contributions. To this end, we present FactCheck, a fact verification system topped by a web-based platform enabling users to inspect every step involved in the KG fact-checking process, and publicly available at https://factcheck.dei.unipd.it/. FactCheck employs multiple LLMs to verify facts, empowering them through a Retrieval Augmented Generation (RAG) pipeline, and labels facts for correctness based on the consensus between the LLM responses. Specifically, the system first converts the target fact into several natural language queries and retrieves, for each query, relevant evidence through a Web search. Then, it independently employs four state-of-the-art LLMs to analyze such evidence, asking them to reason about the veracity of the fact and provide a conclusion on its correctness. Finally, a consensus mechanism determines the correctness of the fact when at least 3 out of 4 LLMs agree.

FactCheck has been adopted to estimate the accuracy of three prominent KG datasets: *FactBench* [10], *YAGO* [18], and *DBpedia* [16]. Table 1 outlines the core characteristics of the datasets.

[1] In this work, we use the terms fact and triple interchangeably.

SIGIR '25, July 13–18, 2025, Padua, Italy

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Table 1: Number of facts and gold accuracy of FactBench, YAGO, and DBpedia datasets

	FactBench	YAGO	DBpedia
Num. of Facts	2,800	1,386	9,344
Gold Accuracy (μ)	0.54	0.99	0.85

Each dataset comes with a gold accuracy, which represents the proportion of true versus false facts, as originally determined by the dataset creators. *FactBench* consists of true facts from DBpedia [4] and Freebase [5] KGs, while it generates false facts by modifying true ones. We consider the configuration where false facts are a mix produced by different negative example generation techniques. This process resulted in a gold accuracy of 0.54, which indicates an almost equal split between true and false facts. On the other hand, both *YAGO* and *DBpedia* contain exclusively human-made fact assessments. *YAGO* presents 1,386 facts with a gold accuracy of 0.99, while *DBpedia* 9,344 facts and a gold accuracy of 0.85.

When used to predict the correctness of facts stored within these KGs, FactCheck achieved prediction performance of 90% on *FactBench*, 87% on *YAGO*, and 70% on *DBpedia*, obtained by comparing FactCheck output with gold standard labels. Hence, FactCheck represents an effective system for KG verification that, paired with the presented web platform, provides a transparent and explainable process, where users can inspect inputs, outputs, and reasoning steps involved in every component of the verification pipeline.

The rest of the paper is organized as follows: Section 2 presents an overview of the related work, discussing previous approaches to fact verification in KGs and the tools that exist for general fact verification. Section 3 describes the FactCheck system in detail, including its verification engine, the architecture of the Web platform, and user interaction capabilities within the system. Section 4 provides concluding remarks, summarizing the key contributions of this work and outlining potential directions for future research.

2 Related Work

Fact-checking methods can be broadly divided into two categories: those that rely on unstructured textual sources [10, 25, 26] and those that exploit structured information [25]. In this study, we focus primarily on approaches based on unstructured textual data, as they are most relevant to our research objectives. For a detailed discussion of methods using structured information sources, refer to [25]. DeFacto [10] is designed to validate KG triples using web-based evidence retrieval in multiple languages. It employs supervised learning methods that rely on large-scale training data and combines trustworthiness metrics with textual evidence to compute an evidence score for each fact. Syed et al. [26] presented a fact validation approach based on machine learning and textual evidence collected from a reference corpus. They convert triple into natural language and retrieve similar sentences from a static corpus to create evidence. Then they find the confidence score for the triple by extracting reliable evidence and features and feeding them to a trained machine learning model. In contrast to [10, 26], our FactCheck integrates multiple LLMs within an RAG framework for fact verification. There are also tools for general claim verification, the Google FactCheck Tool[2] allows users to easily browse and search for fact checks. This tool does not endorse or create any of these fact checks, simply collecting the results from various publishers. Originality.ai's Automated Fact Checker[3] evaluates each fact by assigning it a status that indicates whether it is potentially true or false. The system offers an explanation and links to sources for its assessment, ensuring that users understand the reasoning behind the classification; however, it does not offer complete transparency and requires payment for access. In contrast, our FactCheck architecture enables users to use the power of LLMs and even replicate the entire system on their own machines.

3 FactCheck

In this section, we first present the FactCheck verification engine and then introduce the architecture of the web platform, together with its features and user capabilities.

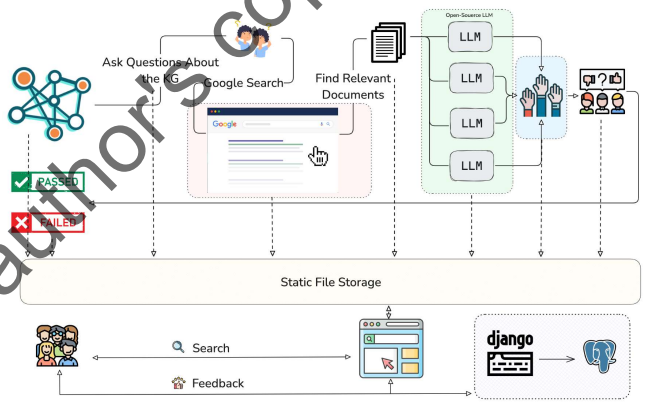

Figure 1: The FactCheck verification engine. A KG triple is converted into a natural language format to generate questions. The evidence is filtered, ranked, and analyzed by an ensemble of LLMs.

Verification Engine. Figure 1 shows the core verification engine of the FactCheck system.

We implement a multistage pipeline that begins with transforming KG triples into human-readable text through an LLM. Let a KG triple be denoted by t. The transformation function $f_{\text{LLM}}(\cdot)$ converts t into a sentence: $s = f_{\text{LLM}}(t)$.

The system then generates 10 distinct questions about each transformed triple. That is, we generate a set of questions $Q = \{q_1, q_2, \ldots, q_{10}\}$ for the sentence s, where each q_i is extracted from the corresponding LLM response. We adopt a cross-encoder architecture to compute similarity scores between the transformed triple s and each generated question q_i. The questions are then sorted in descending order of similarity, and the top 3 questions are selected for further processing. In other words, we select a subset $Q_{\text{top}} \subset Q$, consisting of top-3 questions.

[2] https://toolbox.google.com/factcheck/explorer/search
[3] https://originality.ai/automated-fact-checker

The top 3 questions and the transformed triple s are submitted to Google Search to retrieve the first 100 results. The query parameters used for fetching the data are as follows: lr = "lang_en", gl = "us", hl = "en", num = "100". For each submitted query $q \in \{s\} \cup \mathcal{Q}_{\text{top}}$, denote the retrieved documents by $\mathcal{D}(q) = \{d_1, d_2, \ldots, d_{100}\}$. We consider the documents to be composed only of the textual content that can be extracted from the crawling of the URLs on the Google Search Engine Results Page (SERP) page. The total pool of documents is thus $\mathcal{D} = \bigcup_{q \in \{s\} \cup \mathcal{Q}} \mathcal{D}(q)$. A filtering step is then applied to remove references to the original KG source – e.g., Wikipedia pages when auditing facts from *DBpedia* and *FactBench* – thereby preventing circular verification and ensuring independence. We denote the set of documents that pass this filtering phase as $\mathcal{D}_{\text{filtered}}$.

We again use a cross-encoder architecture to identify the 10 most similar documents. For each document $d \in \mathcal{D}_{\text{filtered}}$, a similarity score is computed. The documents are ranked in descending order of similarity, and the top 10 compose the final set: $\mathcal{D}_{\text{final}}$. These documents are divided into smaller, overlapping chunks using a sliding-window chunking strategy and processed using the *bge-small-en-v1.5* model [27] to generate semantic embeddings.

The processed documents are then analyzed by an ensemble of 4 distinct lightweight LLMs: *Qwen2.5:7b* [21], *LLama3.1:8b* [3, 7], *Mistral:7b* [15], and *Gemma2:9b* [22]. Let the predictions from these models be p_1, p_2, p_3, and p_4 respectively, where each p_i is the prediction of the ith LLM about the correctness of a triple t. The majority vote determines the final prediction label $y = \text{mode}\{p_1, p_2, p_3, p_4\}$. In cases where there is a tie (2 models versus 2), the system identifies the two models whose outputs differ the most in terms of their consistency in performance. One of these models has produced more consistent results across the dataset, while the other has shown less consistent performance. The system then makes the final decision by referring to a higher-parameter variant of the most/least consistent model. Finally, the procedure results in a binary classification categorizing the facts as correct (true) or incorrect (false). As **RAG components setup**, we set *Gemma2:9b* [22] as transformation function and question generator, *Jina-reranker-v1*[4] for question similarity, and *ms-marco-MiniLM-L-6-v2*[5] for document re-ranking.

Web Platform Architecture. The FactCheck web platform consists of (i) a static file storage that contains all the procedural logs and data assets of the core engine, (ii) a web-based front-end interface built with React.js, and (iii) a back-end built using the Django framework integrated with a PostgreSQL DB. When a user searches for a specific fact representing their query, the system searches the stored KG triples by examining exact and partial matches in the subject, predicate, and object fields. All relevant suggestions are then displayed, ensuring users receive a complete set of possible matches for their query. After selecting a specific triple, the relevant data are retrieved from static file storage and shown to the user step by step, following the process described in Paragraph §3.

User Interaction and Features. FactCheck enables users to verify facts through a dedicated fact search engine. The system supports structured query input and is complemented by an auto-completion module. The verification interface contains (1) a search history, (2) a fact verification procedure, (3) an error analysis, and (4) a feedback system.

The sidebar interface allows users to track the **search history** and manage previous fact-checking queries in the browser's local storage. History entries show a verified fact, its source database, and precise timestamp. At the bottom, a "Clear History" button enables users to remove all stored searches at once.

The KG **fact verification procedure** presents the results of each component of the verification engine, organized into five sections, labeled from A to E as shown in Figure 2. Section A presents the KG triple processing interface, where users can view the input triple structure. The example shows a triple about "Romain_Padovani", presenting subject, predicate, and object components (A.1). The system generates a human-readable output, displayed in a blue-highlighted box, which transforms the structured data into natural language (A.2). The "Show/Hide" button indicated by (A.3) in the interface allows users to see the prompt used in each section.

Moving to Section B, the question generation and ranking interface provides insight into fact exploration. The questions generated and ranked by FactCheck based on similarity with the transformed triple are depicted. The interface displays 10 questions, each with an assigned cross-encoder similarity score (B.4). The top-3 questions are distinguished with unique tags – i.e., "Best Match", "Second Best", and "Third Best" – thus clarifying which questions are the most similar to the target KG triple (B.5). We recall that these top-3 questions are those submitted to Google Search.

The SERP management interface, shown in Section C, provides access to the sources collected as evidence for each triple. The interface displays four distinct sources, each retrieved by Google Search, with individual "View Content" buttons. These results refer to the top-3 questions and the transformed triple. The interface includes two clickable buttons, labeled (C.6) and (C.7). Clicking the "View Content" button (C.6) opens a pop-up displaying the Google page retrieved during the verification process. On the other hand, clicking the link button (C.7) redirects the user to the up-to-date Google results page, allowing users to compare whether the data has been affected over time.

Section D introduces the document management interface, which manages the top 10 most relevant documents retrieved with the original query (i.e., the transformed triple). Each document includes a "View" button (D.8), allowing users to access the actual text content of the crawled webpage for verification purposes.

The LLM verification interface in Section E displays results from four different LLMs. Each LLM response includes a verification status indicator, either a green checkmark for "Verified" or a red X for "Not Verified" (E.10), along with the response generated by the LLM itself (E.11). This allows users to see the LLM's output in response to the input prompt. The upper section (E.9) shows the actual label assigned to the fact based on the KG dataset. In addition, there is a bar indicator that clearly visualizes the response distribution. Finally, the majority vote decides the final predicted label, which is visible at the top of each page.

The final interface, called the tiebreaker verification interface, is only activated when the majority vote results in a 2-2 tie. In such cases, the same interface as Section E is displayed, but only with two models chosen based on their consistency for the dataset. These models are now responsible for making the decision.

[4] https://huggingface.co/jinaai/jina-reranker-v1-turbo-en
[5] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

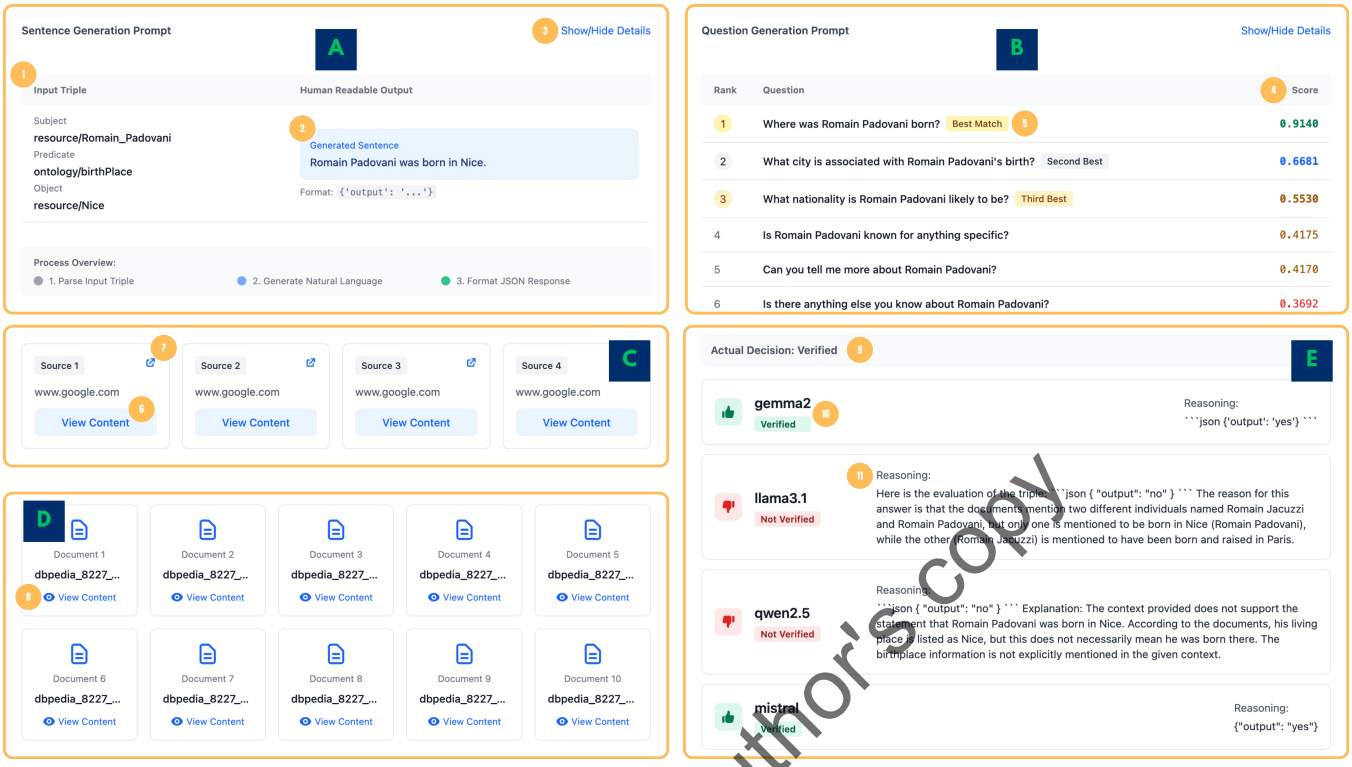

Figure 2: FactCheck system visualization for the triple *Romain_Padovani birthPlace Nice*. The interface is structured into cards: (A) sentence generation—converting the KG triple into human-readable text; (B) question generation—ranking generated questions with similarity scores; (C) evidence retrieval—Google search results for the top-3 questions and transformed triple; (D) document processing—top-10 most similar documents; and (E) model verification—results from four language models.

The **error analysis section** offers insights to experts by identifying specific weaknesses and reasoning fallacies behind LLM's incorrect decisions. This section highlights areas for improvement by categorizing the errors observed in the wrongly predicted triples. Errors are grouped into the following categories: Unlabeled, Relationship, Role attribute, Geo/nationality, Genre/classification, and Identifier/biographical errors. Furthermore, for *DBpedia*, FactCheck offers a stratified error analysis that groups errors by the popularity of affected triples, helping users to quickly identify which triples are more commonly represented or prone to errors.

In addition, in the **feedback section**, users can log in using their Google or ORCID accounts to submit feedback, which is stored in the database and made accessible to other users. The interactive feedback mechanism within the FactCheck interface allows users to contribute to real assessments, thereby improving the effectiveness of human-LLM interaction. Feedback is collected using the options: "uncertain," "agree," "disagree," and free text input. These feedbacks are then displayed anonymously, helping to create a transparent and collaborative environment for further investigation.

4 Final Remarks

We presented FactCheck, which shows both the potential and the challenges of using multiple LLMs for fact verification. The prediction performance of FactCheck – measured against gold standard labels – are 90% on *FactBench*, 87% on *YAGO*, and 70% on *DBpedia*. These results emphasize the potential of using LLMs to address fact verification in KGs. The FactCheck web-based interface allows users to inspect the inputs, outputs, and reasoning steps involved in each component of the verification process by LLMs. While FactCheck is designed for KG fact verification, the same model can be adapted to other fact-checking tasks, such as verifying claims in news articles, scientific literature, or social media content.

One of our observations is that LLMs can reach different conclusions despite accessing the same evidence. This variation occurs because they rely on contextual reasoning, and differences in their underlying architectures lead them to interpret the evidence differently. Also, in fact verification tasks, when a binary evaluation is requested (i.e., determining whether a statement is correct or incorrect), LLMs often explain when they classify a statement as incorrect. However, when predicting a statement is correct, they typically do not explain further.

One direction for future research is to address (1) how to minimize disagreement in predicting the correctness of the same fact, (2) how to combine the LLMs output better, and (3) how to improve verification transparency.

Acknowledgments

We used Claude [claude.ai] to help rephrase some sentences.

References

[1] Garima Agrawal, Dimitri Bertsekas, and Huan Liu. 2023. Auction-Based Learning for Question Answering over Knowledge Graphs. *Information* 14, 6 (2023). https://doi.org/10.3390/info14060336

[2] Naser Ahmadi, Joohyung Lee, Paolo Papotti, and Mohammed Saeed. 2019. Explainable Fact Checking with Probabilistic Answer Set Programming. arXiv:1906.09198 [cs.DB] https://arxiv.org/abs/1906.09198

[3] Meta AI. 2023. Introducing LLaMA 3: Advancing Open Foundation Models. https://ai.meta.com/blog/meta-llama-3-1/. Accessed: 2024-10-17.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference* (Busan, Korea) *(ISWC'07/ASWC'07)*. Springer-Verlag, Berlin, Heidelberg, 722–735.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 1247–1250. https://doi.org/10.1145/1376616.1376746

[6] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. 2013. Building, maintaining, and using knowledge bases: a report from the trenches. In *Proc. of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. ACM, 1209–1220. https://doi.org/10.1145/2463676.2465297

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[8] Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 87–95. https://doi.org/10.1145/3289600.3290996

[9] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. 2019. Efficient Knowledge Graph Accuracy Evaluation. arXiv:1907.09657 https://arxiv.org/abs/1907.09657

[10] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and René Speck. 2015. DeFacto—Temporal and multilingual Deep Fact Validation. *Journal of Web Semantics* 35 (2015), 85–101. https://doi.org/10.1016/j.websem.2015.08.001 Machine Learning and Data Mining for the Semantic Web (MLDMSW).

[11] Google-Blog. 2012. Introducing the Knowledge Graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/ Accessed: 2024-12-09.

[12] Tanya Goyal and Greg Durrett. 2021. Annotating and Modeling Fine-grained Factuality in Summarization. arXiv:2104.04302 [cs.CL] https://arxiv.org/abs/2104.04302

[13] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. arXiv:2003.00911 [cs.IR] https://arxiv.org/abs/2003.00911

[14] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics* 10 (02 2022), 178–206. https://doi.org/10.1162/tacl_a_00454 arXiv:https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00454/1987018/tacl_a_00454.pdf

[15] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[16] Stefano Marchesin, Gianmaria Silvello, and Omar Alonso. 2024. Utility-Oriented Knowledge Graph Accuracy Estimation with Limited Annotations: A Case Study on DBpedia. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 12, 1 (Oct. 2024), 105–114. https://doi.org/10.1609/hcomp.v12i1.31605

[17] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 12076–12100. https://doi.org/10.18653/v1/2023.emnlp-main.741

[18] Prakhar Ojha and Partha Talukdar. 2017. KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Martha Palmer, Rebecca Hwa, and Sebastian Riedel (Eds.). Association for Computational Linguistics, Copenhagen, Denmark, 1741–1750. https://doi.org/10.18653/v1/D17-1183

[19] Reham Omar, Ishika Dhall, Panos Kalnis, and Essam Mansour. 2023. A Universal Question-Answering Platform for Knowledge Graphs. arXiv:2303.00595 [cs.AI] https://arxiv.org/abs/2303.00595

[20] J. Pujara, E. Augustine, and L. Getoor. 2017. Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short. In *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. ACL, 1751–1756. https://doi.org/10.18653/v1/d17-1184

[21] Qwen-Team. 2024. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5/

[22] Morgane Riviere, Shreya Pathak, and Pier Giuseppe Sessa et al. 2024. Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118 [cs.CL] https://arxiv.org/abs/2408.00118

[23] Prakhar Singh, Anubrata Das, Junyi Jessy Li, and Matthew Lease. 2022. The Case for Claim Difficulty Assessment in Automatic Fact Checking. arXiv:2109.09689 [cs.CL] https://arxiv.org/abs/2109.09689

[24] Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? A.K.A. Will LLMs Replace Knowledge Graphs?. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (Eds.). Association for Computational Linguistics, 311–325. https://doi.org/10.18653/V1/2024.NAACL-LONG.18

[25] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2019. Unsupervised Discovery of Corroborative Paths for Fact Validation. In *The Semantic Web – ISWC 2019*, Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). Springer International Publishing, Cham, 630–646.

[26] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2018. FactCheck: Validating RDF Triples Using Textual Evidence. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1599–1602. https://doi.org/10.1145/3269206.3269308

[27] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

Received 18 February 2025