# KGGEN: EXTRACTING KNOWLEDGE GRAPHS FROM PLAIN TEXT WITH LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent interest in building foundation models for KGs has highlighted a fundamental challenge: knowledge-graph data is relatively scarce. The best-known KGs are primarily human-labeled, created by pattern-matching, or extracted using early NLP techniques. While human-generated KGs are in short supply, automatically extracted KGs are of questionable quality. We present a solution to this data scarcity problem in the form of a text-to-KG generator (KGGen), a package that uses language models to create high-quality graphs from plaintext. Unlike other KG extractors, KGGen clusters related entities to reduce sparsity in extracted KGs. KGGen is available as a Python library (`pip install REDACTED`), making it accessible to anyone with an OpenAI API key. Along with KGGen, we release the first benchmark that tests an extractor's ability to produce a useful KG from plain text. We benchmark our new tool against existing extractors and demonstrate far superior performance.

## 1 INTRODUCTION

Knowledge graph (KG) applications and Graph Retrieval-Augmented Generation (RAG) systems are increasingly bottlenecked by the scarcity and incompleteness of available KGs. KGs consist of a set of subject-predicate-object triples, and have become a fundamental data structure for information retrieval (Schneider, 1973). Most real-world KGs, including Wikidata (Wikidata contributors, 2024), DBpedia (Lehmann et al., 2015), and YAGO (Suchanek et al., 2007), are far from complete, with many missing relations between entities (Shenoy et al., 2021). The lack of domain-specific and verified graph data poses a serious challenge for downstream tasks such as KG embeddings, graph RAG, and synthetic graph training data.

Embedding algorithms such as TransE (Bordes et al., 2013) rely on abundant relational data to learn high-quality KG representations. In particular, TransE represents relationships as vector translations between entity embeddings and has demonstrated strong performance in link prediction when trained on large KGs (e.g., 1M entities and 17m training samples). However, if the KG is sparse or incomplete, embedding models struggle – they cannot learn or infer missing links effectively, degrading performance on knowledge completion and reasoning tasks (Pujara et al., 2017; Pote, 2024).

Consider retrieval-augmented generation (RAG) with a language model (LM) – this requires a rich external knowledge source to ground its responses. For instance, GraphRAG integrates a KG into the RAG pipeline (Edge et al., 2024). In GraphRAG, a language model (LM) like GPT-4o is used to extract a KG from a text corpus automatically, and this graph is used for retrieval and reasoning. This structured, graph-based augmentation has been shown to improve multi-hop reasoning and synthesis of information across documents (Larson & Truitt, 2024). By traversing relationships in the constructed graph, GraphRAG can "connect the dots" between disparate pieces of information, outperforming baseline RAG that relies only on semantic search over text. However, GraphRAG's performance ultimately depends on the quality of the extracted graph (Zhang et al., 2024). In practice, automatically constructed graphs can be noisy and incomplete – some false nodes and edges may be introduced and some important ones omitted, which can hinder downstream reasoning (Thakur, 2024).

An emerging line of work that builds on graph-based RAG trains neural networks on KG retrieval. For example, GFM-RAG (Graph Foundation Model for RAG) (Luo et al., 2025) trains a dedicated graph neural network on an extensive collection of KGs, encompassing 60 graphs with over 14 million triples to serve as a foundation model for graph-based retrieval. By learning from diverse KGs, GFM-RAG's retriever can generalize to unseen graphs and better handle the noise/incompleteness in automatically extracted KGs. These efforts underscore the importance of having dense, well-connected KGs to feed into RAG systems.

In this work, we propose KGGen (Text-to-Knowledge-Graph), a package that leverages LMs and a clustering algorithm to extract high-quality, dense KGs from text. KGGen addresses knowledge scarcity by enabling the automatic construction of KGs from any textual source rather than being limited to pre-existing databases like Wikipedia. The package uses an LM-based extractor to read unstructured text and predict subject-predicate-object triples to capture entities and relations. KGGen then applies an iterative LM-based clustering to refine the raw graph. Inspired by crowdsourcing strategies for entity resolution (Wang et al., 2012), the clustering stage has an LM examine the set of extracted nodes and edges to identify which ones refer to the same underlying entities or concepts. Variations in tense, plurality, stemming, or capitalization are normalized in this process - e.g., "labors" might be clustered with "labor" and "New York City" with "NYC." The resulting KG has far less redundancy and is densely interlinked, making it suitable for downstream use.

In addition to KGGen, we provide the first benchmark to measure text-to-knowledge-graph extraction. Our benchmark feeds 100 Wikipedia-length articles into a KG extractor, then uses RAG to answer questions about the articles. On our benchmark, KGGen outperforms leading existing text-to-KG extractors by $18\%$. KGGen paves the way for a data-rich future when training next-generation KG foundation models and RAG systems.

To summarize our contributions:

1. We introduce KGGen, an open-source package that uses LMs to extract high-quality KGs from plain text. Our package is available as a Python library.
2. We develop the first-ever benchmark for text-to-KG extractors, allowing for a fair comparison of existing methods.
3. We show that KGGen outperforms existing extraction methods by $18\%$ on this benchmark, exhibiting its potential to produce functional KGs using LMs.

## 2 Related Work

Interest in automated methods to produce structured text to store ontologies dates back to at least 2001 when large volumes of plain text began to flood the fledgling internet (Maedche & Staab, 2001). KG extraction from unstructured text has seen significant advances through rule-based and LM-powered approaches in the last 15 years. Early work (Suchanek et al., 2007) used hard-coded rules to develop YAGO, a KG extracted from Wikipedia containing over five million facts, and rules-based extraction still has appeal for those producing KGs in multi-modal domains today (Norabid & Fauzi, 2022; Oramas et al., 2015). With the development of modern natural language processing, hard-coded rules generally ceded to more advanced approaches based on neural networks. For instance, OpenIE (Angeli et al., 2015) provides a two-tiered extraction system: first, self-contained clauses are identified by a classifier; then, Angeli et al. run natural logic inference to extract the most representative entities and relations from the identified clauses. Stanford KBP (Angeli et al., 2013) presents another seminal early approach to using deep networks for entity extraction.

As early as 2015, some hypothesized that extracting KGs would go hand-in-hand with developing better language models (Domeniconi et al., 2015). More recently, evidence has emerged that transformer-based architectures can identify complex relationships between entities, leading to a wave of transformer-based KG extraction techniques, which range from fully automatic (Qiao et al., 2022; Arsenyan et al., 2023; Zhang & Soh, 2024) to human-assisted (Kommineni et al., 2024). Our contribution to the extraction literature is to build KGs conducive to embedding algorithms such as TransE and TransR (Bordes et al., 2013; Lin et al., 2015). We observed that when one extracts KGs from plaintext, the nodes and relations are often so specific that they are unique. This causes the estimation of embeddings to be under-specified. We develop a method for automatic KG extraction from plain text that clusters similar nodes and edges to prevent this under-specification. This leads to a KG with better connectivity and more functional nodes and edges.

Evaluating the quality of knowledge graphs is important to ensure usefulness and reliability in downstream applications. Early evaluation methods focused primarily on directly assessing aspects such as completeness and connectivity or using rule-based statistical methods, while recent approaches emphasize usability in downstream applications and incorporation of semantic coherence(Xue & Zou, 2023).

In the late 2000s, research focused on assessing the correctness and consistency of KGs. The evaluations relied on expert annotations by selecting random facts from the generated KG and then calculating the accuracy of those facts. (Suchanek et al., 2007) This proved to be laborious and prone to errors. This led to accuracy approximation methods like KGEval (Ojha & Talukdar, 2017) and Two-State Weight Clustering Sampling(TWCS) (Gao et al., 2018), which employed sampling methods with statistical guarantees as well as use less annotation labor. As the KGs became larger and more diverse, particularly with the rise of automated extraction techniques from web data, this generated more pressure on annotators, leading to methods like Monte-Carlo search being used for the interactive annotation of triples (Qi et al., 2022). Furthermore, because accuracy alone did not fully capture the complexity of the knowledge graph, more evaluation metrics like completeness were used to characterize the quality of knowledge graphs. (Issa et al., 2021).

In recent years, the evaluation of knowledge graphs (KGs) has increasingly focused on their role in downstream AI applications, such as augmenting language models (Schneider et al., 2022) and recommendation systems (He et al., 2020). As a result, semantic coherence and usability have become key criteria for assessing the quality of extracted knowledge graphs.

Two notable approaches to KG evaluation are the LP-Measure and the triple trustworthiness measurement (KGTtm) model. LP-Measure assesses the quality of a KG through link prediction tasks, eliminating the need for human labor or a gold standard (Zhu et al., 2023). This method evaluates KGs based on their consistency and redundancy by removing a portion of the graph and testing whether the removed triples can be recovered through link prediction tools. Empirical evidence suggests that LP-Measure can effectively distinguish between "good" and "bad" KGs. The KGTtm model, on the other hand, evaluates the coherence of triples within a knowledge graph Jia et al. (2019). Based on these evaluation methods, frameworks like Knowledge Graph Evaluation via Downstream Tasks(KGrEaT) and DiffQ(differential testing) emerged. KGrEaT provides a comprehensive assessment of KGs by evaluating their performance on downstream tasks such as classification, clustering, and recommendation (Heist et al., 2023) rather than focusing solely on correctness or completeness. In contrast, DiffQ uses embedding models to evaluate the KG's quality and assign a DiffQ Score, resulting in improved KG quality assessment. Tan et al. (2024)

This shift towards task-based evaluation underscores the importance of usability and accessibility in KGs. Factors such as expressiveness, context information, and ease of integration into downstream AI applications are now central to evaluating their quality and effectiveness.

# 3 KGGen: KGs From Plain Text

Unlike most previous methods of LLM-based KG extraction, we rely on a multi-stage approach involving an LLM (in our case, GPT-4o) to (1) extract entity and relations from each source text, (2) aggregate graphs across sources and (3) iteratively cluster entities and relations. We implement these stages in a modular fashion via a new 'NAME REDACTED' Python toolkit consisting of a 'generate' module for extraction, an 'aggregate' module for source consolidation, and a 'cluster' module for dynamic entity resolution. We use the DSPy framework throughout these stages to define signatures that ensure that LLM responses are consistent JSON-formatted outputs. In our case, we use GPT-4o, although the implementation may be used with any model supported by DSPy.

We impose strong constraints on the LLM via prompting to reduce the likelihood of semantically dissimilar duplicate entities. We introduce multiple passes through our extracted edges and relations to cluster similar entities and consolidate the number of edge types. Consolidation and clustering prevent the formation of sparse KGs, which may produce meaningless KG embeddings under standard algorithms such as TransE.

Our extraction method involves several steps, which we outline below. The exact prompts for each step can be found in Appendix A.

### 3.1 ENTITY AND RELATION EXTRACTION (‘GENERATE‘)

The first stage takes unstructured text as input and produces an initial knowledge graph as extracted triples. We invoke the GPT-4o model for each input text through a DSPy signature that instructs the model to output detected entities in a structured format. Then, we invoke a second LLM call through DSPy that instructs the model to output the subject-predicate-object relations, given the set of entities and source text. We find this 2-step approach works better to ensure consistency between entities.

### 3.2 AGGREGATION (‘AGGREGATE‘)

After extracting triples from each source text, we collect all the unique entities and edges across all source graphs and combine them into a single graph. All entities and edges are normalized to be in lowercase letters only. The aggregation step reduces redundancy in the KG. Note that the aggregation step does not require an LLM.

### 3.3 ENTITY AND EDGE CLUSTERING (‘CLUSTER‘)

After extraction and aggregation, we typically have a raw graph containing duplicate or synonymous entities and possibly redundant edges. The clustering stage is a key innovation in our KG extraction methodology that aims to merge nodes and edges representing the same real-world entity or concept. We take an iterative LLM-based approach to clustering, inspired by how a group of humans might gradually agree on consolidating terms. Rather than attempting to solve the entire clustering in one shot (which is intractable for an extensive list of entities), KGGen performs a sequential series of clustering operations for entities:

1. The entire entities list is passed in context to the LLM, and it attempts to extract a single cluster. An optional cluster-instruction string may be passed to decide how to cluster. The default instructions account for close synonyms and differences in tense and plurality.

2. Validate the single cluster using an LLM-as-a-Judge call with a binary response. If it passes, then add the cluster and remove the cluster entities from the entities list.

3. Assign a label to the cluster that most closely captures the shared meaning of entities in the cluster.

4. Repeat steps 1–3 until $n$ loops happen without a successful cluster extraction.

5. Remaining entities are checked batch-by-batch, with batch size $b$, for whether they should be added to an existing cluster.

6. For each new addition to a cluster, validate the cluster once more using an LLM-as-a-Judge call with a binary response.

7. Repeat steps 5–6 until there are no remaining entities to check.

The same operations are performed on edges, albeit with slightly modified prompts.

The clustering process allows us to create dense KGs that admit meaningful embeddings. To give a real example of the usefulness of our process, in one of our raw KGs, we found the entities “vulnerabilities”, “vulnerable”, and “weaknesses”. Although these are different words, they have similar meanings and should be viewed as equivalent in our KG.

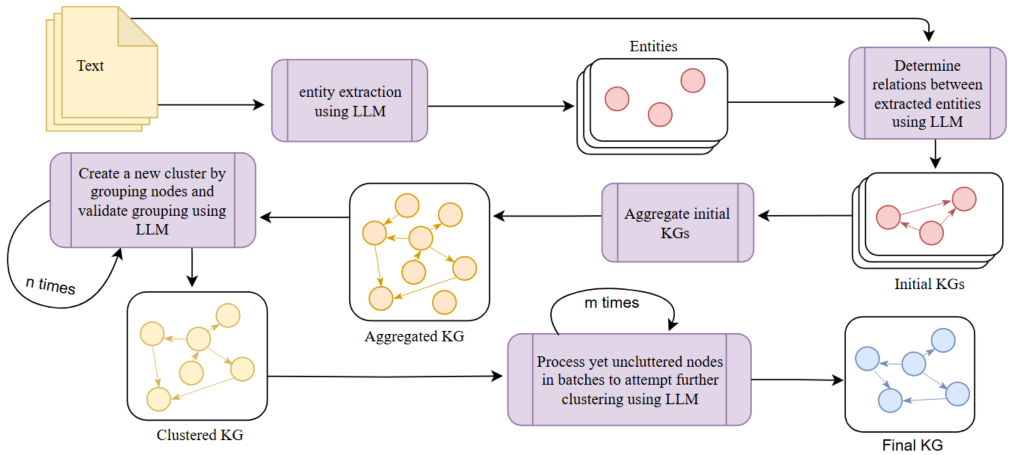



Figure 1: KGGen extraction method

## 4 A Benchmark for Extraction Performance

Although a handful of existing methods attempt to extract KGs from plain text, it is difficult to measure progress in the field due to the lack of existing benchmarks. To remedy this, we produce the Measure of Information in Nodes and Edges (MINE), the first benchmark that measures a knowledge-graph extractor's ability to capture and distill a body of text into a KG.

### 4.1 MINE Description

MINE involves generating KGs for 100 articles, each representing a distinct source of textual data. Each article is approximately 1,000 words long and is generated by an LLM based on a diverse list of 100 topics that range from history and art to science, ethics, and psychology. To evaluate the quality of the generated KGs, we develop a metric to assess how effectively they capture critical information from the articles.

We extract 15 facts–here defined as statements present in the plain text article–from each article by providing an LLM with the article and the extraction prompt found in Appendix C. We manually verify that the 15 facts are accurate and contained in the article. MINE assesses how well a text-to-KG extractor captures the information present in the text by determining whether these 15 facts are captured by the KG generated from the article.

For each article, KGs are generated using three methods: KGGen, OpenIE, and GraphRAG. The nodes of the resulting KGs are then vectorized using the all-MiniLM-L6-v2 model from Sentence-Transformers, enabling us to use cosine similarity to assess semantic closeness between the short sentence information and the nodes in the graph.

For each KG generation method, the KG for each article is queried for each of the 15 facts from that article. We do this by determining the top-k nodes most semantically similar to each fact. Next, we determine all the nodes within two relations of one of the top k-nodes. Finally, we return all these nodes along with their relations as the result of the query. This result is subsequently evaluated using an LLM, provided it is queried for and a specific prompt to produce a binary output: 1 if the fact could be inferred from only the information in the queried nodes and relations, and zero otherwise. The prompt can be found in Appendix C.

The final MINE score of each KG on a given article was calculated as the percentage of 1s across all 15 evaluations. This systematic approach objectively compares the methods based on their ability to capture and retrieve information from the articles accurately.

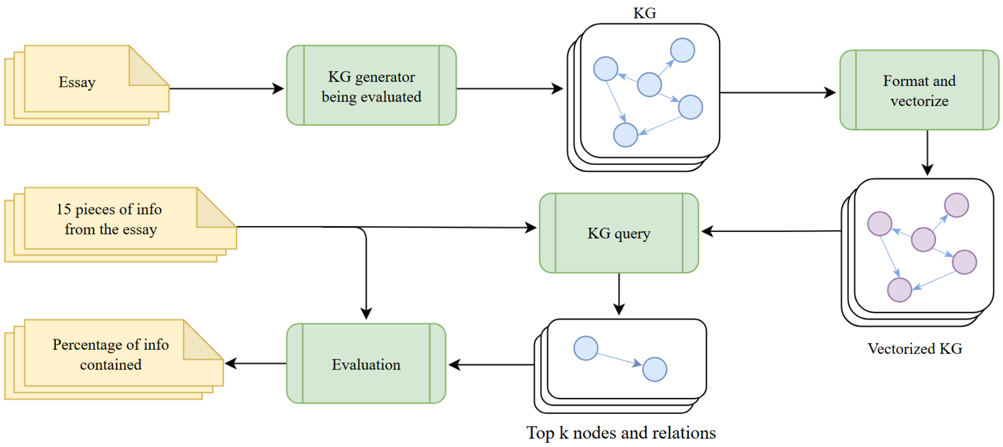



Figure 2: Evaluation process used in MINE

# 5 Results

This section presents the quantitative performance of all three methods on MINE. This is followed by a qualitative discussion on the deficiencies of KGs produced by OpenIE and GraphRAG when compared to those made by KGGen.

## 5.1 Quantitative Results

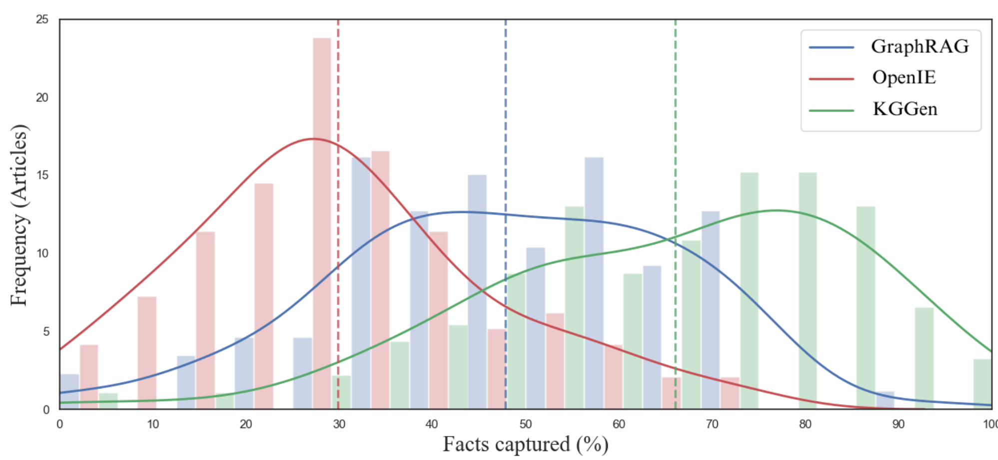


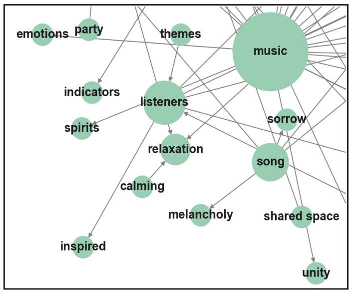

Figure 3: Distribution of MINE scores across 100 articles for GraphRAG, OpenIE, and KGGen. Dotted vertical lines show average performance. KGGen scored 66.07% on average, significantly outperforming GraphRag 47.80% and OpenIE 29.84%.

Figure 3 illustrates the distribution of the percentage of information captured by the KGs across all articles for the three methods: KGGen, OpenIE, and GraphRAG. KGGen scores 66.07%, a significant improvement over the competition: OpenIE scored 29.84%, and GraphRAG scored 47.80%.

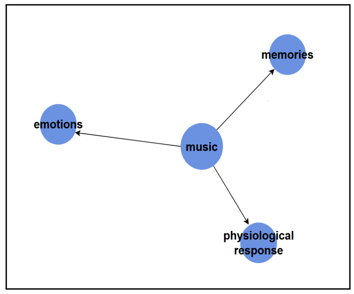
## 5.2 Qualitative Results

As seen in Figure 4b and 4e, GraphRAG often generates a minimal number of nodes and connections for an entire article. This sparsity results in the omission of critical relationships and information.

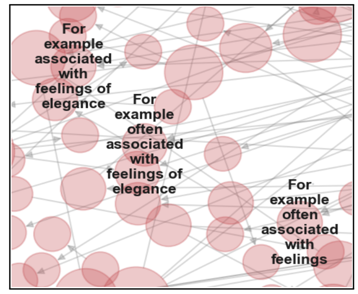
For compression, Figure 4a and 4d illustrate sections of the KGs generated by KGGen for the same articles. Figure 4c illustrates one of many issues in OpenIE's KGs. Firstly, most nodes are unreasonably long, incoherent phrases. Many of these nodes are redundant copies of one another, adding unnecessary complexity to the graph. Additionally, as seen in 4f OpenIE frequently produces generic nodes such as "it" and "are." Due to their frequency, these nodes, which contain no useful information, often end up as some of the most well-connected nodes in the graph. By contrast, KGGen consistently generates KGs that are dense and coherent, effectively capturing critical relationships and information from the articles.



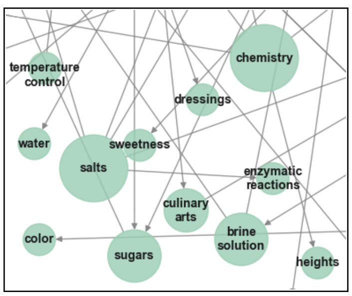

(a) Section of KG generated by KGGen on "How Music Influences Mood"



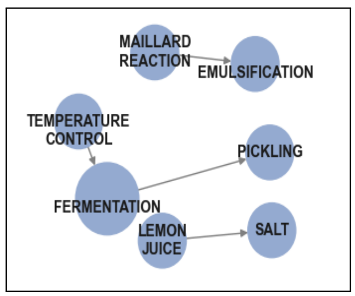

(b) Full KG generated by GraphRAG on "How Music Influences Mood"



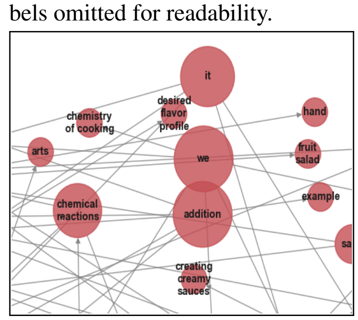

(c) Section of KG generated by OpenIE, on "How Music Influences Mood", with most node labels omitted for readability.




(d) Section of KG generated by KGGen on "The Chemistry of Cooking"




(e) Full KG generated by GraphRAG on "The Chemistry of Cooking"




(f) Section of KG generated by OpenIE on "The Chemistry of Cooking"

Figure 4: Visual comparison of KGs generated using KGGen, GraphRAG, and OpenIE. Results show that KGGen discovers more informative nodes to estimate a richer graph compared to GraphRAG, and collapses synonyms to discover a more informative graph than OpenIE.

## 6 Future Work

We propose MINE – the first benchmark for KG extraction from plain text. To solve the data-shortage hindering development of graph-based foundation models, we present KGGen, a plain-text-to-KG extractor that outperforms existing approaches by up to $18\%$ on MINE.

Although KGGen beats existing methods by significant margins, the graphs still exhibit problems, like over or under-clustering. More research into better forms of clustering could improve the quality of our KGs. Additionally, our benchmark, MINE, currently measures performance on relatively short corpora, whereas KGs are primarily used to handle massive amounts of information efficiently. Future expansions of our benchmark could focus on larger corpora to better measure the practicality of different extraction techniques.

## References

Gabor Angeli, Arun Tejasvi Chaganty, Angel X. Chang, Kevin Scott Reschke, Julie Tibshirani, Jean Wu, Osbert Bastani, Keith Siilats, and Christopher D. Manning. Stanford's 2013 kbp system.

*Theory and Applications of Categories*, 2013. URL https://api.semanticscholar.org/CorpusID:14273633.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 344–354, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1034. URL https://aclanthology.org/P15-1034.

Vahan Arsenyan, Spartak Bughdaryan, Fadi Shaya, Kent Small, and Davit Shahnazaryan. Large language models for biomedical knowledge graph construction: Information extraction from emr notes. In *Workshop on Biomedical Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:256390090.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. A study on term weighting for text categorization: A novel supervised variant of tf.idf. 07 2015. doi: 10.5220/0005511900260037.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. URL https://arxiv.org/abs/2404.16130.

Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. Efficient knowledge graph accuracy evaluation. *ACM Transactions on Information Systems*, 36(2): 1–21, 2018. doi: 10.14778/3342263.3342642. URL https://dl.acm.org/doi/pdf/10.14778/3342263.3342642. Duke University and Amazon.com.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*, pp. 639–648. ACM, 2020. doi: 10.1145/3397271.3401063. URL https://doi.org/10.1145/3397271.3401063.

Nicolas Heist, Sven Hertling, and Heiko Paulheim. Kgreat: A framework to evaluate knowledge graphs via downstream tasks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, pp. 3938–3942. ACM, 2023. doi: 10.1145/3583780.3615241. URL https://doi.org/10.1145/3583780.3615241. Published on 21 October 2023.

Subhi Issa, Onaopepo Adekunle, Fayçal Hamdi, Samira Si-Said Cherfi, Michel Dumontier, and Amrapali Zaveri. Knowledge graph completeness: A systematic literature review. *IEEE Access*, 9:31322–31339, 2021. doi: 10.1109/ACCESS.2021.3056622. URL https://ieeexplore.ieee.org/document/9344615.

Shengbin Jia, Yang Xiang, Xiaojun Chen, Kun Wang, and Shijia. Triple trustworthiness measurement for knowledge graph. In *Proceedings of the World Wide Web Conference (WWW '19)*, pp. 2865–2871. ACM, May 2019. doi: 10.1145/3308558.3313586. URL https://doi.org/10.1145/3308558.3313586.

Vamsi Krishna Kommineni, Birgitta König-Ries, and Sheeba Samuel. From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *ArXiv*, abs/2403.08345, 2024. URL https://api.semanticscholar.org/CorpusID:268379482.

Jonathan Larson and Steven Truitt. Graphrag: Unlocking llm discovery on narrative private data. February 2024. URL https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015. doi: 10.3233/SW-140.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2181–2187. AAAI Press, 2015. ISBN 0262511290.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: Graph foundation model for retrieval augmented generation, 2025. URL https://arxiv.org/abs/2502.01113.

Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16:72–79, 03 2001. doi: 10.1109/5254.920602.

Idza Aisara Norabid and Fariza Fauzi. Rule-based text extraction for multimodal knowledge graph. *International Journal of Advanced Computer Science and Applications*, 2022. URL https://api.semanticscholar.org/CorpusID:249304784.

Prakhar Ojha and Partha Talukdar. KGEval: Accuracy estimation of automatically constructed knowledge graphs. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1741–1750, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1183. URL https://aclanthology.org/D17-1183/.

Sergio Oramas, Mohamed Sordo, and Luis Espinosa-Anke. A rule-based approach to extracting relations from music tidbits. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pp. 661–666, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334730. doi: 10.1145/2740908.2741709. URL https://doi.org/10.1145/2740908.2741709.

Manita Pote. Survey on embedding models for knowledge graph and its applications, 2024. URL https://arxiv.org/abs/2404.09167.

Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1751–1756, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1184. URL https://aclanthology.org/D17-1184/.

Yifan Qi, Weiguo Zheng, Liang Hong, and Lei Zou. Evaluating knowledge graph accuracy powered by optimized human-machine collaboration. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*, pp. 1368–1378. ACM, 2022. doi: 10.1145/3534678.3539233. URL https://doi.org/10.1145/3534678.3539233.

Bin Qiao, Zhiliang Zou, Yurong Huang, Buyue Wang, and Changlong Yu. A joint model for entity and relation extraction based on BERT. *Neural Computing and Applications*, 34(5):3471–3483, 2022. ISSN 1433-3058. doi: 10.1007/s00521-021-05815-z. URL https://doi.org/10.1007/s00521-021-05815-z.

Edward W. Schneider. Course modularization applied: The interface system and its implications for sequence control and data analysis. In *Association for the Development of Instructional Systems (ADIS)*, Chicago, Illinois, April 1973. Presented in April 1972.

Phillip Schneider, Tim Schopf, Juraj Vladika, Mikhail Galkin, Elena Simperl, and Florian Matthes. A decade of knowledge graphs in natural language processing: A survey. 11 2022. doi: 10.18653/v1/2022.aacl-main.46.

Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely. A study of the quality of wikidata, 06 2021.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 697–706, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242667. URL https://doi.org/10.1145/1242572.1242667.

Jiajun Tan, Dong Wang, Jingyu Sun, Zixi Liu, Xiaoruo Li, and Yang Feng. Towards assessing the quality of knowledge graphs via differential testing. *Available online, Version of Record*, 2024. URL https://doi.org/10.1016/j.jss.2024.07.005. Received 3 October 2023, Revised 15 June 2024, Accepted 26 June 2024, Available online 29 June 2024.

Harish Thakur. Automatic knowledge graphs: The impossible grail. *Towards AI*, January 2024. URL https://pub.towardsai.net/automatic-knowledge-graphs-the-impossible-grail-ef71f9c8aad8.

Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Crowder: crowdsourcing entity resolution. *Proc. VLDB Endow.*, 5(11):1483–1494, July 2012. ISSN 2150-8097. doi: 10.14778/2350229.2350263. URL https://doi.org/10.14778/2350229.2350263.

Wikidata contributors. Wikidata: A Free Collaborative Knowledge Base, 2024. URL https://www.wikidata.org. Accessed: 2024-02-05.

Bingcong Xue and Lei Zou. Knowledge graph quality management: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4969–4988, May 2023. ISSN 1041-4347. doi: 10.1109/TKDE.2022.3150080. URL https://doi.org/10.1109/TKDE.2022.3150080. Published on 10 February 2022.

Bowen Zhang and Harold Soh. Extract, define, canonicalize: An llm-based framework for knowledge graph construction. In *Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://api.semanticscholar.org/CorpusID:268987666.

Jian Zhang, Wei Liu, Shuo Wang, and Muhan Zhang. Mindful-rag: A study of points of failure in retrieval augmented generation. *arXiv*, March 2024. URL https://arxiv.org/abs/2407.12216.

Ruiqi Zhu, Alan Bundy, Jeff Pan, Kwabena Nuamah, Fangrong Wang, Xue Li, Lei Xu, and Stefano Mauceri. Assessing the quality of a knowledge graph via link prediction tasks. In *Proceedings of the 7th International Conference on Natural Language Processing and Information Retrieval (NLPIR 2023)*, pp. 1–10, Seoul, Republic of Korea, December 2023. ACM. doi: 10.1145/3639233.3639357. URL https://doi.org/10.1145/3639233.3639357. School of Informatics, University of Edinburgh, United Kingdom; Huawei Ireland Research Centre, Ireland.

## A PROMPTS FOR KG EXTRACTION

This section provides the exact prompts used to extract KG's from the text.

The initial KG is extracted using the following two prompts.

**Prompt for extracting entities:** `Extract key entities from the given text. Extracted entities are nouns, verbs, or adjectives, particularly regarding sentiment. This is for an extraction task, please be thorough and accurate to the reference text.`

**Prompt for extracting relations:** `Extract subject-predicate-object triples from the assistant message. A predicate (1-3 words) defines the relationship between the subject and object. Relationship may be fact or sentiment based on assistant's message. Subject and object are entities. Entities provided are from the assistant message and prior conversation history, though you may not need all of them. This is for an extraction task, please be thorough, accurate, and faithful to the reference text.`

After extracting the entities and relations from each unit of text, we begin the clustering process, which is performed using the following prompts.

**Prompt for clustering entities:**

```
Find ONE cluster of related entities from this list.  A
cluster should contain entities that are the same in
meaning, with different:
- tenses
- plural forms
- stem forms
- upper/lower cases
Or entities with close semantic meanings.
Return only if you find entities that clearly belong
together.
If you can’t find a clear cluster, return an empty list.
```

**Prompt for validating node clusters:**

```
Verify if these entities belong in the same cluster.
A cluster should contain entities that are the same in
meaning, with different:
- tenses
- plural forms
- stem forms
- upper/lower cases
Or entities with close semantic meanings.
Return the entities that you are confident belong together
as a single cluster.
If you’re not confident, return an empty list.
```

**Prompt for clustering edges**

```
Find ONE cluster of closely related predicates from this
list.
A cluster should contain predicates that are the same in
meaning, with different:
- tenses
- plural forms
- stem forms
- upper/lower cases
Predicates are the relations between subject and object
entities.  Ensure that the predicates in the same cluster
have very close semantic meanings to describe the relation
between the same subject and object entities.
Return only if you find predicates that clearly belong
together.
If you can’t find a clear cluster, return an empty list.
```

**Prompt for validating cluster edges**

```
Verify if these predicates belong in the same cluster.
A cluster should contain predicates that are the same in
meaning, with different:
- tenses
- plural forms
- stem forms
- upper/lower cases
Predicates are the relations between subject and object
entities.  Ensure that the predicates in the same cluster
have very close semantic meanings to describe the relation
between the same subject and object entities.
Return the predicates that you are confident belong together
as a single cluster.
If you’re not confident, return an empty list.
```

## B VALIDATION OF KG EXTRACTION

This section provides the LLM generations used to validate our KG extraction method.

**Prompt for extracting entities:** `Extract key entities from the given text. Extracted entities are nouns, verbs, or adjectives, particularly regarding sentiment. This is for an extraction task, please be thorough and accurate to the reference text.`

**Prompt for extracting relations:** `Extract subject-predicate-object triples from the assistant message. A predicate (1-3 words) defines the relationship between the subject and object. Relationship may be fact or sentiment based on assistant's message. Subject and object are entities. Entities provided are from the assistant message and prior conversation history, though you may not need all of them. This is for an extraction task, please be thorough, accurate, and faithful to the reference text.`

## C PROMPTS FOR MINE

In this section, we provide the LLM prompts used by MINE to evaluate KGs.

**Prompt for extracting a fact from article:** `Extract 15 basic, single pieces of information from the following text that describe how one object relates to another. Present the pieces of info in short sentences and DO NOT include info not directly present in the text. Your output should be of the form [ "info1", "info2" ,..., "info15" ]. "Make sure the strings are valid Python strings."`

**Prompt for evaluating if a fact is contained in the query result:**

```
ROLE: "You are an evaluator that checks if the correct
answer can be deduced from the information in the context.
TASK: Determine whether the context contains the information
stated in the correct answer.
Respond with "1" if yes, and "0" if no. Do not provide any
explanation, just the number.
```