Geometric Algebra based encoding for graph prompting

Sotirios Panagiotis Chytas [1] **Rudrasis Chackrabotry** [2] **Vikas Singh** [1]

Abstract

Recent results show that modern Large Language Models (LLM) are indeed capable of understanding and answering questions about structured data such as graphs. Existing proposals often use some description of the graph to create an "augmented" prompt fed to the LLM. For a chosen class of graphs, if a well-tailored graph encoder is deployed to play together with a pre-trained LLM, the model can answer graph-related questions well. Existing solutions to graph-based prompts range from graph serialization to graph transformers. In this work, we show that the use of a parameter-free graph encoder based on Algebraic representations, a concept borrowed from mathematical physics, is remarkably versatile in this problem setting. The simple construction, inherited directly from the theory with a few small adjustments, can provide rich and informative graph encodings, for a wide range of different graphs. We investigate the use of this idea for prefix-tuned prompts leveraging the capabilities of a pre-trained, frozen LLM. The modifications lead to a model that can answer graph-related questions – from simple graphs to proteins to hypergraphs – effectively and with minimal, if any, adjustments to the architecture. Our work significantly simplifies existing solutions and generalizes well to multiple different graph-based structures effortlessly.

1. Introduction

Large Language Models (LLMs) excel at tasks like question answering, sentence completion, translation, and even solving undergraduate-level math problems (Liu et al., 2024; Johansson, 2024). However, they sometimes need additional data unavailable during training. For instance, a model trained on data up to a specific date may struggle with the ever-changing news cycle (Vu et al., 2023; Mousavi et al., 2024). To prevent responses from becoming outdated, or to integrate non-public/proprietary data and domain-specific terminology, models need extra context. This need has led to strategies like Retrieval Augmented Generation (RAG) (Guu et al., 2020; Ding et al., 2024; Dong et al., 2022; Zoph et al., 2022; Min et al., 2022). RAG allows additional information to be included with a prompt, guiding the model to generate responses aligned with the extra context. This method is beneficial as it does not require retraining the LLM and can be applied to proprietary models like GPT (Brown et al., 2020) by adding a text description of the extra information.

RAG-type ideas are also being studied for utilizing not just additional/new data but also novel input formats/modalities, such as tables and graphs (Sui et al., 2024; Lu et al., 2024; Wang et al., 2023; Guo et al., 2023). Several recent results have reported success at "serializing" such structured datatypes into a text-form description that can be easily used within RAG. For tables, the serialization is not too complicated (Sui et al., 2024; Lu et al., 2024), but more care is needed for graphs. While different types of graphs can all be handled by the same pipeline, the efficacy of the overall model varies from one setting to the other (Fatemi et al., 2024; Wang et al., 2023; Guo et al., 2023). Further, it has been observed that specific design choices to "textify" the graph can influence performance and additionally, prompting techniques can have more than a small impact on the results (Fatemi et al., 2024). What will work well in a specific setting depends on both the question at hand as well as the characteristics of the data (Perozzi et al., 2024; Chai et al., 2023).

Prefix-tuning. One emerging option to address the mentioned issues is "prefix-tuning" (Li & Liang, 2021). A specialized graph encoder translates the underlying graph into embeddings that can be fed directly to an LLM, eliminating the need for a textual description. Though not training-free, the LLM remains *frozen*, and only the *relatively smaller* graph encoder is trained. This approach has shown impressive performance, often surpassing RAG-based methods. However, using a specialized graph encoder can be challenging due to the variety of graph types. For example,

[1]University of Wisconsin-Madison [2]Lawrence Livermore National Lab. Correspondence to: Sotirios Panagiotis Chytas <chytas@wisc.edu>.

Proceedings of the ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery, Vienna, Austria. 2024. Copyright 2024 by the author(s).

GraphToken (Perozzi et al., 2024) can encode only simple graphs, while GNP (Tian et al., 2024) constructs a complex pipeline to handle large graphs and extract subgraphs out of them. GraphLLM (Chai et al., 2023) combines a transformer and a GNN (about $100M$ parameters). Despite sophisticated designs, adapting these models to different graph types (e.g., protein-derived graphs or hypergraphs) is difficult, and even familiar graph types need adjustments for new tasks.

Context of this paper. RAG-based approaches for graphs primarily involve converting graphs to text, while prefix-tuning with graphs uses modules to extract richer, *task-relevant* structures, requiring larger sample sizes and higher compute power. A key question is whether we can achieve powerful, task-agnostic graph representations that are as easy to obtain as RAG-based methods. Could a lightweight adapter map these rich (but task-independent) representations into the LLM embedding space, making prefix-tuning effective for various tasks? Recent results hint that this may be viable (Moayeri et al., 2023). For instance, a *single linear layer* can transform an arbitrary image encoder's outputs to align with CLIP's (Radford et al., 2021) text encoder embeddings. If our graph encoding captures the graph's information and structure well enough, a similar adapter could work with a pre-trained LLM to offer good performance. We ensure this by invoking a mature concept from mathematical physics, called Fock Spaces. Our findings show that a linear adapter with these representations yields competitive performance, handling complex graph questions and diverse structures like hypergraphs and proteins.

2. Deriving Fock space based Graph Representation

Setup/rationale. Assume we are given a graph $G = (V, E)$ with a vertex set V and an edge set E. To obtain a complete representation for G, we start with the Dirac operator, leveraging the structure of G (Hatcher, 2002). A significant body of work focuses on representing graphs through various techniques, such as graph spectra derived from the Laplacian's eigenvalues (which we will also calculate below). Spectral graph theory provides powerful tools for studying global properties of graphs like connectivity and symmetries (e.g., Courant Fischer theorem, Fiedler's theorem (Fiedler, 1973; 1989)). However, it is less effective for capturing relationships between individual elements (nodes and edges) within the graph, as the eigenvalues are scalar quantities. Our approach below will seek to exploit the richness of an algebraic structure that will allow more sophisticated manipulations, explore the interplay between different parts of the graph if needed and thereby, offer more flexibility.

By borrowing ideas from quantum mechanics (Weyl, 1950; Prugovecki, 2006) and Algebraic representations from Clifford algebra, we represent G equipped with the *Dirac operator* with a representation in the Fock space.

Representing nodes, sums, and products. Our default approach assigns a vector to each concept (node, edge, etc.) by generating random, orthogonal vectors in high dimensions. When the number of vectors exceeds the dimensionality, preserving linear independence is not possible. Although there are workarounds (e.g., space-filling frames (Sustik et al., 2007; Casazza et al., 2008)), we address this simply by sampling vectors from a normal distribution $\mathcal{N}(\mathbf{0}, 1/d)$, resulting in nearly orthogonal vectors. Practically, the maximum absolute cosine similarity between any two vectors remains below 0.1, consistent with known results (Blum et al., 2020). For Fock space calculations like sums and products, we prefer operations that preserve dimensionality rather than tensor products, which significantly increase dimensionality (Wolff et al., 2018). This simplifies implementation, as all resultant embeddings maintain the same dimensionality regardless of the encoding method. We define the sum $(\oplus)$ as element-wise addition, and the product $(\otimes)$ as circular convolution. This can be seen as element-wise multiplication of the vectors' Fourier representations followed by an inverse Fourier transformation. For vectors $\in \mathbb{R}^d$, the complexity is $\mathcal{O}(d \log d)$. This also allows us to define the inverse vector, i.e., for any vector b, we have a vector a such that a's Fourier representation is the inverse of b's. So, the identity $a \otimes b = \mathbf{1}$ holds. But our experiments are not tied to this specific implementation, and improved choices can be dropped in.

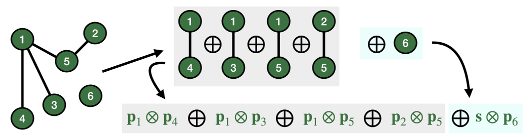
2.1. Systematic Encoding recipes for Graphs

Notice that, we use a parameter-free scheme to obtain rich graph embeddings. Our approach is general and can handle a large spectrum of different graph types, and its extension to novel graph-types is straightforward. Diverse graph types such as hypergraphs, attributed graphs, as well as proteins can all be modeled easily providing an alternative or a good initialization for more intensive trainable models.

For a graph $G = (V, E)$ we have a vector $\mathbf{p}_i$, using i to index the nodes. We also use an extra vector $\mathbf{s}$ for the graph's size, a practical design choice we will explain shortly. Then, with these $n + 1$ vectors, we obtain a lossless Fock-space based representation $\mathbf{g}$ as:

$$\mathbf{g} = \left(\mathbf{s} \otimes \mathbf{p}_n\right) \oplus \bigoplus_{(i,j)\in E} \left(\mathbf{p}_i \otimes \mathbf{p}_j\right) \tag{1}$$

Each edge's endpoints are fused together using $\otimes$ and then we aggregate all edges together using $\oplus$. Finally, the graph's size is also added using the special vector $\mathbf{s}$.


$\mathbf{p}_1 \otimes \mathbf{p}_4 \oplus \mathbf{p}_1 \otimes \mathbf{p}_3 \oplus \mathbf{p}_1 \otimes \mathbf{p}_5 \oplus \mathbf{p}_2 \otimes \mathbf{p}_5 \oplus \mathbf{s} \otimes \mathbf{p}_6$

Lossless representation. The above representation is lossless. Assuming we use (1) to get a graph's embedding $\mathbf{g}$. Then, simply by evaluating the expression $\mathbf{p}_j^T(\mathbf{p}_i^{-1} \otimes \mathbf{g})$, we can determine whether the edge (i, j) exists in the edge set of that particular graph. In this way, we can recover, one by one, all edges of the graph and correctly reconstruct it, if desired. It is instructive to check the importance of $\mathbf{s}$. By evaluating the expression $\mathbf{p}_i^T(\mathbf{s}^{-1} \otimes \mathbf{g})$, $\forall i$, we can first obtain the size of the graph. This can inform the edge retrieval above because an expression of the form $\mathbf{p}_{n+x}^T(\mathbf{p}_i^{-1} \otimes \mathbf{g})$ could, in practice, produce a number close to 1, although there is no such edge. By first obtaining the size of the graph, we have a "safeguard" against such phantom edges beyond the real vertex-set.

Vertex attributes. Consider a graph $G = (V, E, Attr)$, where the set $Attr$ (with $|Attr| = |V|$) consists of attributes, one for each vertex. There is no restriction on the type of attributes: it can denote numerical values or text or any other concept. Let $\mathbf{a}_i$ be the vector associated with the attribute of vertex $i \in V$ (using an appropriate text-encoder if needed). Then, we can augment (1) to absorb the extra information in the following way:

$$\mathbf{g} = (\mathbf{s} \otimes \mathbf{p}_n) \oplus \bigoplus_{(i,j)\in E} (\mathbf{p}_i \otimes \mathbf{p}_j) \oplus \bigoplus_{i \in V} (\mathbf{p}_i \otimes \mathbf{a}_i) \tag{2}$$

The graph is again, fully reconstructable. We have also encoded each vertex's attribute (which can be recovered by the expression $\mathbf{a}_j^T(\mathbf{p}_i^{-1} \otimes \mathbf{g})$). We should think of proteins as a graph with vertex attributes where each vertex is a specific amino acid (possibly with 3-D coordinates).

Hypergraphs (Theory versus Practice). Hypergraphs are generalizations of graphs: each edge is connected to an arbitrary number of vertices, instead of just 2. In theory, we can easily augment (1) so that we can handle hypergraphs as follows: $\mathbf{g} = \big(\mathbf{s} \otimes \mathbf{p}_n\big) \oplus \bigoplus_{(k_1,\cdots k_m)\in E} \bigotimes_{i=1}^{m} \mathbf{p}_{k_i}$. In practice, aggregating many multiple vectors together may be unstable. This is true for our particular design choices for calculations (e.g., circular convolution), so we use an alternative approach. We can start by observing that each edge can be interpreted as a unique cluster of vertices, so we simply assign a unique vector $\mathbf{e}_i$, $i \in \big[|E|\big]$ to each edge in the hypergraph. This modification allows us to encode the hypergraph similar to how a graph is encoded as a dictionary, in the following way $\mathbf{g} = \big(\mathbf{s} \otimes \mathbf{p}_n\big) \oplus \left(\bigoplus_{i=1}^{|E|} \left(\mathbf{e}_i \otimes \bigoplus_{j \in E_i} \mathbf{p}_j\right)\right)$.

Recent works showed that (a) textualizing a graph and pre-appending it to a question results in better-than-random responses from the LLM (although far from perfect), and (b) using a specialized graph encoder such as a GNN or a graph transformer and training along with a frozen LLM results in a big improvement in performance, resulting essentially in LLMs that can understand, to some extent, graphical structures. One takeaway is that we can bypass the most tedious stage of designing application-specific graph encoders. Instead, we can use a parameter-free method for a wide range of graph types, as we described above. Thus, the only trainable parts of the pipeline are simple linear adapters that convert the raw graph encodings to a format "understandable" by an LLM. Our FockLLM is shown in Fig. 1. After getting the graph encodings, we train one/more linear adapters and append the transformed encodings to the question's embeddings fed to the LLM.

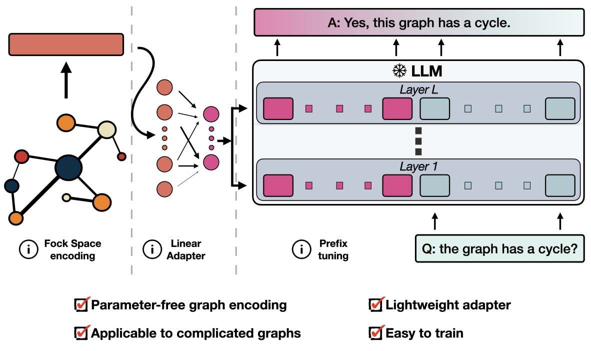

A: Yes, this graph has a cycle.
LLM
Layer L
Layer 1
Fock Space encoding
Linear Adapter
Prefix tuning
Q: the graph has a cycle?
Parameter-free graph encoding
Lightweight adapter
Applicable to complicated graphs
Easy to train

Figure 1. FockLLM overview. Using a parameter-free graph encoder we get graph embeddings for a range of different graphs. Then, we use linear adapters with a frozen LLM for *prefix tuning*.

Summary and Takeaway. We highlight some qualitative advantages. *First*, our graph encoding is parameter-free and efficient. The complexity of aggregation is $\mathcal{O}(d \log d)$ (d is vectors' dimension). The number of aggregation operations is linear (in graph size). *Second*, our encoder is not restricted to specific graph types: works easily for simple graphs, for proteins and for hypergraphs just via small modifications. In contrast, GraphToken (Perozzi et al., 2024) uses a specific GNN whose output size is dependent on the underlying task. These properties simplify our training and eliminates any tunable components. *Third*, our open-source code offers a scalable way to train FockLLM even on consumer GPUs, by using FSDP (Zhao et al., 2023). As a reference, GraphToken (Perozzi et al., 2024) is trained on TPUs (code unavailable).

3. Experimental Results

We examine the performance in two real biomedical datasets including (PPI (Hamilton et al., 2017), Table 1) and (obnbbench (Liu & Krishnan, 2024), Table 2). We see that our approach is among the best unsupervised approaches (for PPI), and is also competitive (if not better) to the best supervised approaches that leverage trainable, graph-specific models such as GCN (Bruna et al., 2014) and GAT (Liu &

Zhou, 2020). Obnbench is a collection of extensive open biomedical repository containing different tasks, and our model is competitive across across all tasks of the benchmark datasets, achieving results better than the best reported performance so far in almost half of the tasks. These results provide encouraging evidence that (a) our approach gives "rich" graph embeddings for a range of different graph types and styles, and (b) our graph embeddings can be used as an extra, grounding input to a powerful LLM without the need to design/train a specialized model, e.g., GNN (Scarselli et al., 2009; Wu et al., 2022) or a Graph Transformer (Dwivedi & Bresson, 2020).

Table 1. Micro F1-score

	Model	F1
	Random	39.2
	Node2Vec (Yun et al., 2022)	40.9
	Raw features (Yun et al., 2022)	42.2
Unsupervised	GraphSAGE-min (Hamilton et al., 2017)	46.5
	GraphSAGE-max (Hamilton et al., 2017)	50.2
	DGI (Veličković et al., 2019)	63.8
	GRACE (Zhu et al., 2020)	66.2
	Ours	**99.2**
Supervised	GraphSAGE-min (Hamilton et al., 2017)	50.0
	GraphSAGE-max (Hamilton et al., 2017)	61.2
	LGCN (Gao et al., 2018)	77.2
	GAT (Liu & Zhou, 2020)	97.3
	GCNII (Chen et al., 2020)	**99.5**

Setup and Results (graph understanding). In this setup, we check whether we can use an LLM to answer questions about graph's structure: number of nodes, whether it has a cycle, and so on. We consider GraphToken and we perform a suite of 6 different experiments. Using GraphQA, although our encodings are not specific to each underlying task, we perform competitively with specialized models. The results are shown in Table 3. Even when GraphToken uses a different embedding for each node or edge (*node degree* and *edge existence* tests resp.), we are able to report a performance quite close to GraphToken, even when using a single embedding for the entire graph. Our model trails only in predicting the node degree (GraphToken uses a different embedding per node that better captures its degree).

4. Conclusions

We have described a novel strategy to encode a graph into a vector form for direct downstream use or to augment prompts fed to LLMs. Our approach, grounded in Clifford algebra and Fock space operations, is rigorous and offers numerous advantages in practice demonstrated via experiments. We can obtain encodings of arbitrary graphs instantly, with no trainable parameters that nicely encapsulates the important information content in the underlying graph. Our model, accompanied with a simple-to-train open-source codebase, performs favorably relative to highly specialized models while at the same time handling classes of graphs where other alternatives fall short or need adjustments.

Table 2. APOP metric (Liu & Krishnan, 2024)

Network	Model	DISEASES	DisGeNET	GOBP
BioGRID	LabelProp	1.210	0.931	1.858
	LogReg	1.556	1.026	2.571
	GCN+BoT	1.511	1.014	2.442
	SAGE+BoT	1.486	1.031	2.402
	GIN+BoT	1.410	1.007	2.386
	GAT+BoT	**1.609**	1.037	**2.624**
	GatedGCN+BoT	1.547	1.038	2.517
	Ours	1.599	**1.062**	2.433
HumanNet	LabelProp	3.728	3.098	3.806
	LogReg	3.812	3.158	**4.053**
	GCN+BoT	3.552	3.053	3.921
	SAGE+BoT	3.401	3.052	3.816
	GIN+BoT	3.513	3.054	3.861
	GAT+BoT	3.761	3.100	3.809
	GatedGCN+BoT	3.677	3.086	3.889
	Ours	**3.853**	**3.254**	3.916
COMPPIHumanInt	LabelProp	1.352	1.106	2.076
	LogReg	1.644	1.240	**2.806**
	GCN+BoT	1.648	1.211	2.685
	SAGE+BoT	**1.694**	1.210	2.629
	GIN+BoT	1.608	1.219	2.611
	GAT+BoT	1.665	1.230	2.755
	GatedGCN+BoT	1.672	1.218	2.735
	Ours	1.660	**1.241**	2.586
BioPlex	LabelProp	0.964	0.939	1.714
	LogReg	**1.358**	**0.939**	2.587
	GCN+BoT	1.324	0.911	2.553
	SAGE+BoT	1.246	0.865	2.513
	GIN+BoT	1.349	0.868	2.504
	GAT+BoT	1.355	0.873	2.548
	GatedGCN+BoT	1.301	0.859	2.590
	Ours	1.273	0.879	**2.599**
HuRI	LabelProp	0.545	0.598	1.086
	LogReg	0.650	0.656	1.084
	GCN+BoT	0.634	0.693	1.129
	SAGE+BoT	0.593	0.679	1.190
	GIN+BoT	0.583	0.702	1.143
	GAT+BoT	0.667	0.687	1.174
	GatedGCN+BoT	0.596	0.695	**1.195**
	Ours	**0.684**	**0.729**	1.070
OmniPath	LabelProp	1.358	0.897	1.593
	LogReg	1.542	**1.093**	**2.125**
	GCN+BoT	**1.577**	1.068	2.071
	SAGE+BoT	1.478	1.062	1.986
	GIN+BoT	1.452	1.073	1.993
	GAT+BoT	1.552	1.048	2.068
	GatedGCN+BoT	1.516	1.049	2.071
	Ours	1.511	1.085	2.102

Table 3. GraphToken vs FockLLM on GraphQA. Column *1* stands for a single embedding for the entire graph; $\mathcal{O}(n)$ stands for a single embedding per node. In all 6 tasks, although we use a parameter-free, predetermined graph encoding, we see a performance similar/better relative to a trainable graph encoder linked with a larger LLM (PaLM-2). For reference, we also include the best performance with any RAG-based technique (Fatemi et al., 2024; Perozzi et al., 2024).

	RAG	GraphToken		**FockLLM**
Tokens	$\mathcal{O}(n^2)$	1	$\mathcal{O}(n)$	1
num of nodes	26.9%	**99.6%**	-	97.2%
num of edges	12.8%	42.6%	-	**45.1%**
cycle existence	83.2%	95.6%	-	**97.9%**
num of triangles	16.2%	34.8%	-	**37.7%**
node degree	28.0%	-	96.2%	62.7%
edge existence	54.4%	-	73.8%	**74.3%**

Acknowledgements. We thank Anthony Gitter for their suggestions. Research was partially supported by the Vilas board of Trustees award and the Bodosaki scholarship.

References

Blum, A., Hopcroft, J., and Kannan, R. *Foundations of Data Science*. Cambridge University Press, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020.

Bruna, J., Zaremba, W., Szlam, A., and Lecun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Casazza, P. G., Kutyniok, G., and Li, S. Fusion frames and distributed processing. *Applied and Computational Harmonic Analysis*, 2008. doi: https://doi.org/10.1016/j.acha.2007.10.001.

Chai, Z., Zhang, T., Wu, L., Han, K., Hu, X., Huang, X., and Yang, Y. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2020.

Ding, Y., Fan, W., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. A survey on rag meets llms: Towards retrieval-augmented large language models. *arXiv preprint arXiv:2405.06211*, 2024.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 1973.

Fiedler, M. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 1989.

Gao, H., Wang, Z., and Ji, S. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219947.

Guo, J., Du, L., and Liu, H. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2020.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2002.

Johansson, M. What can large language models do for theorem proving and formal methods? In *Bridging the Gap Between AI and Reality*, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-46002-9.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353.

Liu, H., Zheng, Z., Qiao, Y., Duan, H., Fei, Z., Zhou, F., Zhang, W., Zhang, S., Lin, D., and Chen, K. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. 2024.

Liu, R. and Krishnan, A. Open biomedical network benchmark: A python toolkit for benchmarking datasets with biomedical networks. In *Machine Learning in Computational Biology*. PMLR, 2024.

Liu, Z. and Zhou, J. *Graph Attention Networks*. Springer International Publishing, Cham, 2020. ISBN 978-3-031-01587-8. doi: 10.1007/978-3-031-01587-8_7.

Lu, W., Zhang, J., Zhang, J., and Chen, Y. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*, 2024.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical*

Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759.

Moayeri, M., Rezaei, K., Sanjabi, M., and Feizi, S. Text-to-concept (and back) via cross-model alignment. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2023.

Mousavi, S. M., Alghisi, S., and Riccardi, G. Is your llm outdated? benchmarking llms & alignment algorithms for time-sensitive knowledge. *arXiv preprint arXiv:2404.08700*, 2024.

Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., and Halcrow, J. Let your graph do the talking: Encoding structured data for llms, 2024.

Prugovecki, E. *Quantum Mechanics in Hilbert Space: Second Edition*. Dover Books on Physics. Dover Publications, 2006. ISBN 9780486453279.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009. doi: 10.1109/TNN.2008.2005605.

Sui, Y., Zhou, M., Zhou, M., Han, S., and Zhang, D. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703713. doi: 10.1145/3616855.3635752.

Sustik, M. A., Tropp, J. A., Dhillon, I. S., and Heath, R. W. On the existence of equiangular tight frames. *Linear Algebra and its Applications*, 2007. doi: https://doi.org/10.1016/j.laa.2007.05.043.

Tian, Y., Song, H., Wang, Z., Wang, H., Hu, Z., Wang, F., Chawla, N. V., and Xu, P. Graph neural prompting with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17), 2024. doi: 10.1609/aaai.v38i17.29875.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, D. Deep graph infomax. In *ICLR 2019*, 2019.

Vu, T., Iyyer, M., Wang, X., Constant, N., Wei, J., Wei, J., Tar, C., Sung, Y.-H., Zhou, D., Le, Q., et al. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*, 2023.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023.

Weyl, H. *The Theory of Groups and Quantum Mechanics*. Dover Books on Mathematics. Dover Publications, 1950. ISBN 9780486602691.

Wolff, M., Wirsching, G., Huber, M., beim Graben, P., Römer, R., and Schmitt, I. A fock space toolbox and some applications in computational cognition. In *Speech and Computer*, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99579-3.

Wu, L., Cui, P., Pei, J., Zhao, L., and Guo, X. Graph neural networks: Foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3542609.

Yun, S., Jeong, M., Yoo, S., Lee, S., Yi, S. S., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks: Learning meta-path graphs to improve gnns. *Neural Networks*, 2022. doi: https://doi.org/10.1016/j.neunet.2022.05.026.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., and Li, S. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12), 2023. doi: 10.14778/3611540.3611569.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Zoph, B., Raffel, C., Schuurmans, D., Yogatama, D., Zhou, D., Metzler, D., Chi, E. H., Wei, J., Dean, J., Fedus, L. B., Bosma, M. P., Vinyals, O., Liang, P., Borgeaud, S., Hashimoto, T. B., and Tay, Y. Emergent abilities of large language models. *TMLR*, 2022.