Improving Link Prediction on Citation Graphs using LLM Embeddings

Robert Guan and JJ Jang
Rice University, Comp 459
rzg3, jj102

Abstract

Link prediction for citations of research papers is a key problem in analyzing scholarly communication and predicting research trends. Typical methods have relied on topological features or sparse textual representations, but these can potentially miss the deeper semantic connections between papers. Our project investigates the effectiveness of leveraging Large Language Models (LLMs) to generate rich node embeddings for citation graph link prediction. We utilize the `mistralai/Mistral-7B-v0.1` model to create embeddings from paper abstracts obtained from the `OpenAlex` dataset. These generated embeddings serve as input features for a Graph Neural Network (GNN) designed for link prediction. We then compare the GNN's performance against standard baselines sch as Common Neighbors. Our experiments performed demonstrate that the GNN model trained on Mistral-7B embeddings are able to significantly outperform the heuristic methods, achieving a test AUC of approximately 0.93, which highlights the possibility of LLM-derived semantic features for enhancing citation link prediction accuracy.

1 Introduction

Citation networks are graphical representations of scholarly communication, which is crucial for understanding the development and evolution of academic research. In these citation networks, nodes represent academic papers and directed edges represent citations, showing a map of knowledge flow and influence. Predicting future links/citations has applications such as identifying potential research collaborations, recommending relevant literature, and forecasting emerging research areas.

Traditional and typical solutions to the link prediction problem for paper citations often depend on graph topology. This often leads to use of heuristic methods such as Common Neighbors (CN), Adamic-Adar (AA), and the Katz index, which all quantify the likelihood of a link based on local or global network structure of the graph. While these methods have proved to be somewhat useful, they often fail to leverage the semantic content embedded in the papers themselves, or strictly use metadata of papers (authors, publication venues, etc.), but still fail to capture semantic relationships.

LLMs have proved large advancements recently in demonstrating capabilities to understand and represent complex textual information. At the same time, GNNs have also proven to be strong tools for learning representations and performing tasks on graph-structured data. This project aims to merge these two recent developments together by using an LLM to generate dense rich embeddings from paper abstracts, which will then be used as node features for a GNN trained specifically for citation link predictions.

Our hypothesis is that LLM-generated embeddings capture semantic similarity and conceptual relatedness between papers more effectively than traditional features, leading to improved link prediction performance when used with a GNN's ability to model graph structure. As aforementioned above, we will be using the `mistralai/Mistral-7B-v0.1` model and applying it to a subset of the large-scale OpenAlex citation dataset.

Our proposed distinctives and contribution are:

1. An end-to-end pipeline for generating LLM-based node embeddings from paper abstracts and constructing a citation graph suitable for GNN-based link prediction
2. Demonstration of the effectiveness of Mistral-7B abstract embeddings as node features for citation link prediction.
3. Comparison of our GNN model using the embeddings against heuristic link prediction baselines (Common Neighbors and Adamic-Adar), showing forth significant performance improvements.

2 Related Work

Citation link prediction, the task of forecasting future connections in scholarly networks, has been a long-standing problem with approaches spanning various disciplines. Early methods predominantly relied on the topological structure of the citation graph. Heuristics such as Common Neighbors Adamic and Adar [2003], Adamic-Adar Adamic and Adar [2003], and the Katz index Katz [1953] quantify the likelihood of a link based on shared neighbors or path counts. Other structural approaches include matrix factorization techniques Menon and Elkan [2011] and probabilistic models Lü and Zhou [2011]. While effective in capturing local or global network patterns, these methods inherently ignore the semantic content of the papers themselves.

More advanced approaches started integrating textual information. Methods have explored using keywords, topics derived from models like Latent Dirichlet Allocation (LDA) Blei et al. [2003], or explicit textual similarity measures between paper abstracts or full texts Lü and Zhou [2011]. These text-based features are often combined with topological features or used in isolation within traditional machine learning models. However, these textual representations were typically sparse or based on simpler embedding techniques compared to modern large language models.

With the rise of deep learning on graphs, Graph Neural Networks (GNNs) have become a powerful tool for link prediction. GNNs learn node representations by aggregating information from their neighbors, effectively capturing complex structural patterns. Standard GNN link prediction models often use an encoder-decoder framework, where a GNN encoder learns node embeddings, and a decoder (like a simple dot product or a multi-layer perceptron) predicts the link probability based on pairs of node embeddings Zhou et al. [2020], Fey and Lenssen [2019]. While powerful in leveraging graph structure, the quality of the node embeddings learned by the GNN is often dependent on the initial node features.

The recent advancements in Large Language Models (LLMs) have opened new avenues for generating rich, contextualized text representations. Pre-trained transformer models have demonstrated exceptional capabilities in capturing semantic nuances and relationships in text. Consequently, there has been growing interest in leveraging LLM-generated embeddings as initial node features for GNNs to enhance performance on various graph-based tasks Jin et al. [2023], He et al. [2023]. For scholarly graphs specifically, LLMs have been used for tasks like paper classification or topic modeling, but their direct application to generate *node features for citation link prediction within a GNN framework*, particularly using recent models like Mistral-7B, remains an active area of exploration. Our work directly contributes to this emerging intersection, specifically demonstrating the efficacy of using embeddings from intermediate LLM layers for this task compared to traditional baselines and different embedding strategies.

3 Methodology

Our approach contains several stages: dataset acquisition and processing, node feature extraction using an LLM, graph construction, definition of link prediction models, and finally the training and evaluation setup.

3.1 Dataset: OpenAlex

We utilized the OpenAlex dataset Priem et al. [2022], which is a large-scale, publicly available knowledge graph of scholarly communication. To manage our computation resources, we focused running our model on a specific subset of papers relevant to computer science and published since 2020. We queried the OpenAlex API using the following filters:

- Primary location's source ID: `S4306400194` (likely corresponding to a specific CS source like arXiv CS).
- Concept ID: `C41008148` (Computer Science).
- Publication date: From 2020-01-01 onwards.

We used the API's cursor pagination, limiting the retrieval to 100 pages at 200 papers per page which resulted in a raw dataset of 20,000 papers. For each paper, we extracted its OpenAlex ID, its list of cited OpenAlex IDs (`referenced_works`), and its abstract (from `abstract_inverted_index`). We then saved this data locally to `.json` files (`openalex_papers_raw.json`, `openalex_citations_raw.json`) to avoid unnessecary API calls.

3.2 Mistral-7B Node Feature Extractions

To capture the semantic content of each paper, we generated node embeddings using the `mistralai/Mistral-7B-v0.1` model Jiang et al. [2023].

Model Loading: We loaded the Mistral-7B model using the Hugging Face `transformers` library. To fit the model onto available GPU memory (NVIDIA Tesla T4 with 15GB), we employed 4-bit quantization using the `bitsandbytes` library Dettmers et al. [2023], with NF4 quantization type (`bnb_4bit_quant_type="nf4"`) and `float16` compute data type.

Input Processing: Each paper's reconstructed abstract served as the input text. The abstracts were tokenized using the Mistral-7B tokenizer, with left-padding applied. We set a maximum sequence length of 512 tokens, truncating longer abstracts. Empty abstracts were represented by the tokenizer's padding token to ensure consistent batch processing.

Embedding Strategy - Layer Choice: Our initial approach involved extracting hidden states from the final layer of the Mistral-7B model. However, when these embeddings were used as input features for the GNN, we observed poor performance: the model tended to overfit quickly, and the resulting validation AUC scores were close to random chance (0.5), indicating that these embeddings did not effectively capture the semantic relationships required for citation prediction.

This observation motivated a change in strategy based on the understanding of LLM internal representations. The final layer of a causal LLM like Mistral-7B is highly specialized for its pre-training objective, that is, predicting the next token. This specialization might cause it to lose some broader semantic information relevant for downstream tasks like semantic similarity. In contrast, intermediate layers, such as the second-to-last layer, often retain more general contextual and semantic information about the entire input sequence before the model narrows its focus to next-token prediction Jawahar et al. [2019]. Therefore, we hypothesized that embeddings derived from an earlier layer might be more suitable for our task.

Following this reasoning and common practices for obtaining sentence/document embeddings from transformers, we switched to using the hidden states from the second-to-last layer (`outputs.hidden_states[-2]`) for generating our node features.

Pooling Strategy: We processed the abstracts in batches (batch size of 8). For each abstract, we obtained the hidden states from the chosen second-to-last layer. To derive a single fixed-size vector representation, we applied **mean pooling** across the sequence length dimension of these hidden states, using the attention mask to exclude contributions from padding tokens. This resulted in a 4096-dimensional embedding vector for each paper node, which formed the basis for our successful GNN experiments.

3.3 Graph Construction

Using the fetched paper data and generated embeddings, we constructed a graph suitable for PyTorch Geometric (PyG) Fey and Lenssen [2019].

1. **ID Mapping:** We created a mapping from the unique OpenAlex paper IDs in our 20,000-paper set to contiguous integer indices (0 to N-1, where N=20,000).
2. **Node Features (x):** The 4096-dimensional Mistral-7B embeddings were ordered according to the ID mapping and stacked into a feature matrix $X \in \mathbb{R}^{N \times 4096}$.
3. **Edge Index ($edge\_index$):** We iterated through the citation data. For each paper u in our set, we checked if any of its cited papers v ($referenced\_works$) were also present in our 20,000-paper set. If both u and v were in the set, we added a directed edge (u, v) to our edge list, represented by their corresponding integer indices. This resulted in 6,114 edges within our subset. The edge list was converted into the PyG $edge\_index$ format (a tensor of shape $[2, num\_edges]$).
4. **Data Object:** The node features X and the $edge\_index$ were combined into a PyG 'Data' object. This object, containing $N = 20,000$ nodes and $E = 6,114$ edges, was saved to disk (`openalex_subset_mistral_base_graph.pt`).

3.4 Link Prediction Models

Heuristic Baselines: We implemented two standard link prediction heuristics as baselines:

- **Common Neighbors (CN):** The score for a potential link (u, v) is $|\Gamma(u) \cap \Gamma(v)|$, where $\Gamma(x)$ is the set of neighbors of node x.
- **Adamic-Adar (AA):** The score is $\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(w)|}$, down-weighting common neighbors with high degrees.

These scores were calculated based on the adjacency structure of the training graph (after splitting) and evaluated on the validation and test edge sets.

Graph Neural Network (GNN): We defined a GNN model for link prediction using PyG, following an encoder-decoder architecture.

- **Encoder:** A multi-layer GNN processes the input node features (Mistral embeddings from the second-to-last layer) and the graph structure ($edge\_index$) to produce final node embeddings Z. We used GraphSAGE Hamilton et al. [2017], layers with mean aggregation. The specific architecture trained consisted of 3 layers: Input(4096) $\rightarrow$ SAGE(128) $\rightarrow$ ReLU $\rightarrow$ LayerNorm $\rightarrow$ Dropout(0.5) $\rightarrow$ SAGE(64) $\rightarrow$ ReLU $\rightarrow$ LayerNorm $\rightarrow$ Dropout(0.5) $\rightarrow$ SAGE(64). LayerNorm and Dropout were applied after intermediate layers.
- **Decoder:** An MLP decoder takes the concatenated embeddings of a source node u and a target node v from the encoder output ($z_u || z_v$) and predicts a score (logit) for the edge (u, v). This involved projecting the concatenated dimension ($64 + 64 = 128$) to 1 via linear layers (potentially with intermediate ReLU activations as defined in the full $LinkPredictorGNN$ class).

The model takes node features X, the message-passing $edge\_index$, and the $edge\_label\_index$ (edges to be predicted) as input and outputs prediction logits for the edges in $edge\_label\_index$.

3.5 Training and Evaluation Setup

Data Splitting: We used PyG's 'RandomLinkSplit' transform to partition the edges of our graph (`data`) into training, validation, and test sets. We configured the split as follows:

- $num\_val = 0.1, num\_test = 0.1$: 10
- $is\_undirected = False$: Citations are directed.
- $add\_negative\_train\_samples = False$: We sample negative edges manually during the training loop.
- $neg\_sampling\_ratio = 1.0$: For validation and test sets, $RandomLinkSplit$ automatically samples an equal number of negative edges (non-existing links).

This resulted in:

- $train\_data$: Contains N nodes, 4892 message-passing edges ($edge\_index$). Positive training edges are implicitly these 4892 edges.
- val_data: Contains N nodes, 4892 message-passing edges (used for GNN propagation during eval), 1222 edges to predict ($edge\_label\_index$), with 611 positive and 611 negative examples ($edge\_label$).
- $test\_data$: Contains N nodes, 5503 message-passing edges (training edges + validation positive edges), 1222 edges to predict ($edge\_label\_index$), with 611 positive and 611 negative examples ($edge\_label$).

GNN Training:

- **Objective:** Binary classification using $torch.nn.BCEWithLogitsLoss$.
- **Negative Sampling:** In each training epoch, for every positive edge in the training message-passing graph ($train\_data.edge\_index$), we sampled one negative edge using PyG's $negative\_sampling$ function (sparse method).
- **Optimizer:** Adam with a learning rate of 1×10^{-5} and weight decay of 5×10^{-4}.
- **Epochs & Early Stopping:** The model was trained for a maximum of 150 epochs. Early stopping was employed with a patience of 15 epochs based on the validation AUC score. Training halts if the validation AUC does not improve for 15 consecutive epochs. The model state corresponding to the best validation AUC was saved.

Evaluation Metrics: We evaluated the models using standard link prediction metrics:

- Area Under the ROC Curve (AUC)
- Precision
- Recall
- F1-Score
- Accuracy

For Precision, Recall, F1-Score, and Accuracy, a classification threshold is required. We determined the optimal threshold by maximizing the F1-score on the validation set predictions generated by the best saved GNN model. This optimized threshold was then applied to the test set predictions for final reporting. Heuristics were evaluated using their raw scores for AUC and an illustrative threshold for other metrics.

4 Experiments and Results

4.1 Experimental Setup

All experiments were conducted on a Google Colab instance equipped with an NVIDIA Tesla T4 GPU (15 GB VRAM). Key software libraries included PyTorch (v2.6.0), PyTorch Geometric (v2.6.1), Transformers (v4.44.1, approx.), BitsAndBytes (v0.43.2, approx.), NumPy, and Scikit-learn. Key hyperparameters for the GNN training are listed in Section 3.5.

4.2 Baseline Results (Heuristics)

We evaluated the Common Neighbors (CN) and Adamic-Adar (AA) heuristics on the validation and test sets. The scores were calculated using the training graph structure ($train\_data.edge\_index$). Results are presented in Table. AUC provides the most reliable comparison, as other metrics depend heavily on threshold selection for heuristics.

4.3 GNN Model Performance

The GNN model using Mistral-7B second-to-last layer embeddings was trained with early stopping based on validation AUC. The training loss decreased steadily over the epochs, indicating successful optimization on the training data, as shown in Figure 2. The validation and test AUC scores improved

Table 1: Performance of Heuristic Baselines on Validation and Test Sets.

Set	Heuristic	AUC	Precision	Recall	F1-Score	Accuracy
Validation	CN	0.6113	0.5000	1.0000	0.6667	0.5000
Test	CN	0.6121	0.5000	1.0000	0.6667	0.5000
Validation	AA	0.5458	0.4831	0.9345	0.6369	0.4673
Test	AA	0.5516	0.4844	0.9394	0.6392	0.4697

rapidly in the initial epochs and then plateaued, as visualized in Figure 1. The model achieved the best validation AUC of 0.9288 at epoch 63. The corresponding test AUC at this epoch was 0.9298. Early stopping was triggered at epoch 78, as the validation AUC did not improve further for 15 consecutive epochs.

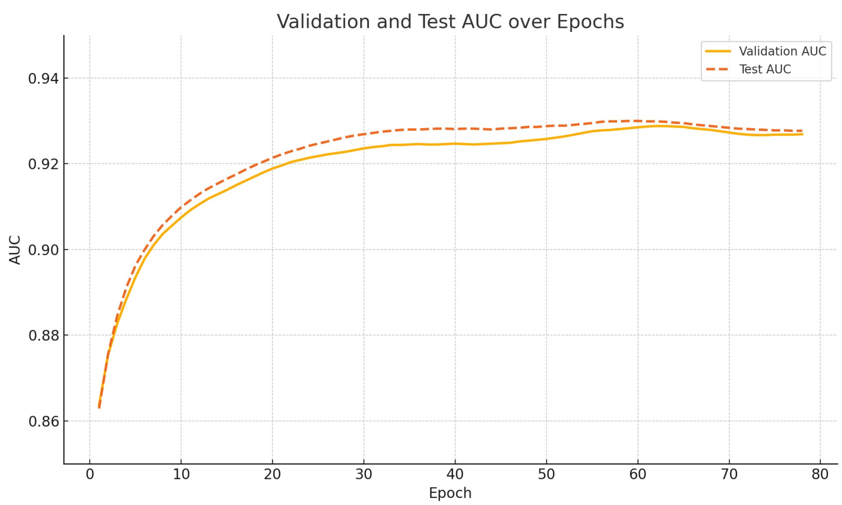

Figure 1: Validation and Test AUC scores over training epochs. The model reached peak validation performance around epoch 63.

Figure 2: Training loss (BCEWithLogitsLoss) over training epochs, showing consistent decrease.

The best saved model (from epoch 63) was evaluated on the test set. The optimal classification threshold determined on the validation set was 0.5918. Applying this threshold to the test set predictions yielded the final performance metrics, summarized and compared with the baselines in Table 1.

Table 2: Final Link Prediction Performance Comparison on the Test Set.

Method	AUC	Precision	Recall	F1-Score	Accuracy
Common Neighbors (CN)	0.6121	0.5000*	1.0000*	0.6667*	0.5000*
Adamic-Adar (AA)	0.5516	0.4844*	0.9394*	0.6392*	0.4697*
GNN + Mistral-7B (L-1)	**0.9298**	**0.8854**	**0.8347**	**0.8593**	**0.8633**

*P/R/F1/Acc for heuristics based on an illustrative threshold; AUC is more reliable.
GNN results use embeddings from the second-to-last (L-1) layer of Mistral-7B.
GNN P/R/F1/Acc based on threshold optimized on validation set (0.5918).

4.4 Comparison and Discussion

The results clearly demonstrate the significant advantage of using the GNN model combined with appropriate LLM embeddings over the purely topology-based heuristic baselines. The GNN using second-to-last layer Mistral-7B embeddings achieved a test AUC of 0.930, dramatically exceeding the 0.612 AUC of the best heuristic (CN). This substantial improvement underscores the value of incorporating semantic information from paper abstracts.

Crucially, the choice of LLM layer for embedding extraction was vital. Our initial experiments using the final hidden layer yielded poor results (AUC roughly 0.5), likely because the final layer's representations are overly specialized for next-token prediction and may lose broader semantic context valuable for similarity tasks. Switching to the second-to-last layer provided embeddings that were far more effective for the downstream link prediction task, aligning with observations that intermediate layers often capture more general semantics.

Our GNN leverages both the rich semantic features from the second-to-last LLM layer and the graph structure information learned through message passing. This two-fold approach allows it to identify potential citation links based on conceptual similarity, even in the absence of strong topological evidence like many shared neighbors. The balanced Precision (0.885) and Recall (0.835) further indicate the GNN's superior ability to discriminate between true and false potential links compared to the baseline methods.

The need for careful hyperparameter tuning (low learning rate, weight decay) highlights the challenges of training GNNs on high-dimensional LLM features, but the final performance demonstrates the effectiveness of the approach when configured correctly.

5 Challenges Faced

5.1 Challenges Encountered

Several challenges were faced during this project:

- **Computational Cost & GPU Memory:** Generating embeddings using Mistral-7B, even quantized, was resource-intensive. Training the GNN on 4096-dimensional features also required significant GPU memory. Given our limited resources as undergraduate students, we felt a bit constrained and wished we could have scaled this implementation and project to a significant level.
- **Embedding Strategy Choice:** Selecting the optimal strategy for extracting embeddings from the LLM was non-trivial. Our initial failure using the final hidden layer necessitated experimentation and relying on insights about LLM layer representations, leading to the successful use of the second-to-last layer. Further exploration of layers or pooling methods could be beneficial but adds to the experimental overhead.
- **Data Handling & API Limitations:** Fetching and processing data from the large OpenAlex dataset required careful management of API calls, pagination, and data reconstruction. Subsetting was necessary for feasibility. Again, this refers back to computational cost;

ideally, with enough resources this would not have been necessary and moving forward wish we could test on the entire dataset, or much more of it.

- **Hyperparameter Tuning:** Optimizing the GNN training process (learning rate, weight decay) was crucial for achieving good performance and stability with the high-dimensional LLM features. Given more time, we believe we could have optimized our model even more to further demonstrate the capabilities of GNN paired with an LLM.

5.2 Future Improvements

The success of our approach using Mistral-7B embeddings and a GNN for citation link prediction opens up several promising avenues for future research. A direct next step is to scale our method to a significantly larger subset of the OpenAlex dataset or ideally, the entire dataset, provided sufficient computational resources can be acquired. This would allow for training on a denser and more representative graph, potentially revealing performance characteristics on a real-world scale.

Further exploration into the choice of LLM and embedding strategy is warranted. Experimenting with different large language models (e.g., Llama, BERT variants, or models specifically fine-tuned for scientific text) could yield even richer semantic representations. Investigating alternative methods for deriving a fixed-size node embedding from the LLM's output, such as using the CLS token equivalent, different pooling strategies, or more sophisticated aggregation techniques across layers, could also lead to performance improvements.

The GNN architecture itself can be further optimized. Testing different GNN layers (e.g., Graph Attention Networks - GATs, Graph Convolutional Networks - GCNs), varying the number of layers, hidden dimensions, and regularization techniques could enhance the model's ability to leverage both the semantic features and the graph structure. Additionally, exploring different decoder architectures beyond simple concatenation and MLP could be beneficial.

A particularly interesting direction is to develop hybrid approaches that combine the LLM-derived semantic embeddings with traditional topological features (like degree, clustering coefficient) or metadata features (authors, venues, keywords) as initial node features for the GNN. This could allow the model to benefit from multiple complementary sources of information.

Finally, incorporating temporal dynamics into the model is crucial for real-world citation prediction. Future work could involve using time-aware GNN architectures or temporal graph sampling techniques to model the evolving nature of the citation network and predict links based on the temporal context of publications. Investigating the explainability of the model's predictions – understanding which semantic cues or structural patterns are most influential in predicting a citation – could also provide valuable insights into scholarly communication.

6 Conclusion

This project successfully demonstrated the effectiveness of using Large Language Model embeddings derived from an intermediate layer (second-to-last) of Mistral-7B as node features for improving citation link prediction. The choice of embedding strategy was critical, as initial attempts using the final layer embeddings failed to yield meaningful results. By combining the semantic understanding captured by the appropriately chosen LLM layer with the structural learning capabilities of a Graph Neural Network, our approach significantly outperformed standard topology-based heuristic methods, achieving a test AUC of approximately 0.930 on a subset of the OpenAlex dataset. This highlights the importance of both leveraging rich textual information via LLMs and carefully considering the embedding extraction process for downstream graph learning tasks. While challenges remain, particularly concerning computational resources and optimal feature representation, this work confirms the strong potential of integrating LLMs and GNNs for advancing our understanding and prediction of scholarly communication networks.

References

Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. In *Social Networks*, volume 25, pages 211–230. Elsevier, 2003. doi: 10.1016/S0378-8733(03)00009-1.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. ISSN 1532-4435.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. URL `https://arxiv.org/abs/2305.14314`.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL `https://arxiv.org/abs/1903.02428`.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 1024–1034. Curran Associates, Inc., 2017. URL `https://papers.nips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html`.

Zhikai He, Haitao Li, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*, 2023. URL `https://arxiv.org/abs/2307.03393`.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3651–3657. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1356. URL `https://aclanthology.org/P19-1356`.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio'ecalsin Lavaud, Marie-Anne Martinet, Marie-Emmanuelle Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. URL `https://arxiv.org/abs/2310.06825`.

Wenzheng Jin, Lingfei Zhao, Yu Zhang, Yang Liu, Shirui Pan, and Dongxiao He. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*, 2023. URL `https://arxiv.org/abs/2312.02783`.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. ISSN 0033-3123. doi: 10.1007/BF02289026.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. ISSN 0378-4371. doi: 10.1016/j.physa.2010.11.027.

Ashok Kumar Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2011)*, volume 6912 of *Lecture Notes in Computer Science*, pages 437–452. Springer, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-23783-6_28.

Jason Priem, Heather Piwowar, and Richard Orr. OpenAlex: a fully open scholarly knowledge graph. *arXiv preprint arXiv:2205.01833*, 2022. URL `https://arxiv.org/abs/2205.01833`.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001.