Leveraging ChatGPT API for Enhanced Data Preprocessing in NatUKE

Pit Fröhlich[†], Jonas Gwozdz[†] and Matthias Jooß[†]

Hochschule für Technik, Wirtschaft und Kultur Leipzig, Karl-Liebknecht-Straße 132, 04277 Leipzig, Deutschland

Abstract

This scientific paper presents an approach for enhancing the performance of machine learning models by utilizing ChatGPT, a state-of-the-art language model developed by OpenAI, for data preprocessing. The study focuses on the existing Project NatUKE (A Benchmark for Natural Product Knowledge Extraction from Academic Literature) and investigates the impact of incorporating ChatGPT in the preprocessing pipeline. By leveraging the natural language processing capabilities of ChatGPT, we aim to improve the quality and relevance of the data used as input for the knowledge graph embedding algorithms. This paper provides a detailed description of the methodology employed, the experimental setup, and the results obtained, highlighting the benefits and limitations of this approach.

Repository: https://github.com/joossm/aksw-bike-natuke

Dataset: https://1drv.ms/f/s!AlepeZ0hNJ0osVyGcyIXdNpGubAB?e=yQXpQe

Keywords

ChatGPT, OpenAI, Data Preprocessing, Machine Learning, Natural Language Processing

1. Introduction

Chemical compounds from living organisms or derived from them contribute as natural products to 67 % of all drugs approved worldwide [1]. The knowledge about these natural products mainly comes from written scientific papers. To extract information from text manually is a time-consuming task. This brought up the approach of using machine learning to automate the extraction. NatUKE [2] provides a benchmark for extracting chemical compound characteristics from academic literature through an unsupervised pipeline based on graph embedding methods. For the First International Biochemical Knowledge Extraction Challenge [3] (BiKE) at the Extended Semantic Web Conference (ESWC) 2023 the goal was set to reuse or design new biochemical extraction methods. As participants, we decided to use the existing NatUKE benchmark and elevate its preprocessing with ChatGPT (Generative Pretrained Transformer). ChatGPT is an advanced conversational artificial intelligence (AI) with the OpenAI GPT-3.5 [4] model being publicly available during this project. According to its documentation, it has

BiKE'23: First International Biochemical Knowledge Extraction Challenge, May 28 - Jun 1, 2023, co-located with Extended Semantic Web Conference (ESWC), Hersonissos, Greece

[†] These authors contributed equally.

pit.froehlich@stud.htwk-leipzig.de (P. Fröhlich); jonas.gwozdz@stud.htwk-leipzig.de (J. Gwozdz); jooss.matthias@gmail.com (M. Jooß)

https://github.com/PitFroehlich (P. Fröhlich); https://github.com/jonasgwozdz (J. Gwozdz); https://github.com/joossm (M. Jooß)

training data up to September 2021 from various internet texts, without going into detail about what these texts are [4, 5]. Since NatUKE uses trained models to identify information, our approach is based on the idea that data that already contains information that got classified by ChatGPT combined with the original data will lead to better results since it is easier for the models to identify the correct information.

2. Related Works

Since this was written in the scope of BiKE and is based on NatUKE [2] and we are using this project's pipeline for our work, we have to mention it at this point. NatUKE provides a benchmark of an unsupervised pipeline for the extraction of knowledge about natural compounds from scientific papers. It uses four methods: DeepWalk [6], Node2Vec [7], Metapath2Vec [8] and EPHEN [9] in a similarity-based graph completion evaluation scenario. These graph embedding methods worked on unsupervised knowledge extraction and are the foundation for this work.

3. ChatGPT

Other notable language models include BERT [10] (Bidirectional Encoder Representations from Transformers) and RoBERTa [11] (Robustly Optimized BERT Pretraining Approach), developed by Google and Facebook, respectively. These models, like GPT, use transformer architectures but are more focused on understanding context in both directions (left-to-right and right-to-left), which makes them great for tasks like question-answering and sentence classification.

Another interesting model is CTRL [12] (Conditional Transformer Language Model), which was developed by Salesforce and allows for more controlled generation of text.

ChatGPT and other transformer-based language models leverage deep learning, a subfield of machine learning. The model is trained on a vast amount of text data, learning to predict the next word in a sentence given the context of the previous words. This learning process involves adjusting millions (or even billions) of internal parameters to minimize the difference between its predictions and the actual words in the training data. This is achieved through a process called back-propagation and an optimization technique known as stochastic gradient descent.

Generative models like GPT [5] and CTRL [12] excel at generating contextually relevant text, making them ideal for tasks like text completion and translation. However, they can sometimes produce text that, while plausible-sounding, may be factually incorrect. Agnostic embedding models like BERT [10] and ELMo [13], on the other hand, focus on understanding the context of words within a sentence. They are effective for tasks requiring semantic understanding, such as sentence classification and sentiment analysis, but do not generate text like GPT and CTRL. In essence, the choice between generative and agnostic embedding models depends on whether text generation or deep semantic understanding is required. However, it's worth noting that the line between these models is blurring, with newer models being designed to perform both tasks effectively.

The transformer architecture that GPT is based on utilizes a mechanism called attention, which allows the model to focus on different parts of the input when generating each word in

the output [14]. This makes it particularly good at handling long-range dependencies in the language.

The training data for GPT models like ChatGPT comes from a diverse range of internet text [5]. However, OpenAI, the organization behind ChatGPT, has not publicly disclosed the specifics of the individual datasets used. It's also important to note that while GPT-3, the basis for the version 3.5 of ChatGPT that was used in this paper, was trained on a broad range of internet text, it does not know specifics about which documents were in its training set or has access to any specific documents or sources.

The dataset will contain information up to the model's training cut-off date. In the case of ChatGPT, it is September 2021 [4]. In the end we decided to use this model since we already had some experience with its API and limited local processing power.

4. Approach

The approach chosen for this task involves an extensive, multistep process to leverage OpenAI's GPT-3.5 Turbo model. By leveraging this model, we are able to process high-quality text data derived from a variety of PDF documents. Our approach, while currently time-consuming, leverages the capabilities of GPT-3.5 Turbo. We took the foundation for this process from the NatUKE Pipeline.

4.1. Process

Our method involves a multistep process that leverages the capabilities of the OpenAI GPT-3.5 Turbo model to extract high-quality text data from PDF documents, process it, and generate a knowledge graph. This process includes PDF extraction, data cleaning, text slicing, prompt generation, building the OpenAI call, invoking the model, retrieving the results, post-processing, and finally training and benchmarking.

4.1.1. PDF Extraction

The first step in processing text data from PDFs involves reading the files and extracting the text. There are several libraries in Python, that allow you to read and extract text data from PDFs. However, it's worth noting that text extraction from PDFs can be a non-trivial task due to the complex nature of PDF formatting. PDF files often contain a mix of textual data, images, tables, and other elements which can make extraction challenging.

4.1.2. Data Cleaning

Once you have extracted the raw text data from the PDFs, you will likely need to perform numerous cleaning steps. This might include removing unwanted characters (like newline characters $\backslash n$ or excessive whitespace), dehyphenation (i.e., the removal of hyphens used to break words at the end of lines), and other custom cleaning based on the peculiarities of your documents. This stage also involves the detection and rectification of potential errors from

Optical Character Recognition (OCR). Ensuring data cleanliness is an iterative and critical process that significantly contributes to the overall quality and reliability of the data.

4.1.3. Text Slicing

Slicing or segmenting the text into smaller units is the next step. The main purpose is to generate pieces of text that are of a manageable size for the model to process.

4.1.4. Prompt Generation

A key aspect of working with GPT-3.5 or any other language model is the generation of prompts [15]. The prompt is the input given to the model, which it will attempt to complete or respond to. The design of the prompt will depend on what kind of task you want the model to perform. It can be as simple as the start of a sentence, or it can be a more complex instruction.

4.1.5. Setting up OpenAI Call

The OpenAI call involves the use of OpenAI's API to interact with the model. This requires setting up the API key, specifying the request URL, and configuring other parameters such as the maximum response length, temperature (randomness), and the top_p/top_k sampling settings.

4.1.6. Model Invocation

Once the setup is complete, you can call the model with your prompts and receive generated text in response. Due to the large size of the model and the computational cost of running it, OpenAI's GPT-3.5 has some limitations, like the number of tokens it can generate in a minute or in a single request. You will need to manage your requests to stay within these limits.

4.1.7. Result Retrieval

The results generated by the GPT-3.5 model are retrieved and stored for further processing. The output will be in the form of text generated by the model, based on the provided prompts. These results could be raw text or structured data, depending on how the prompts are designed and how the model is used.

4.1.8. Post-Processing

This step involves cleaning and structuring the output data from the model. The goal is to remove any irrelevant or incorrect data (like "No Information" categories) and to format the remaining data in a way that is useful for your specific use case.

4.1.9. Training and Benchmarking

The cleaned and structured output data can then be used for further model training. In this process, we blend the training data from GPT with PDF phrases to enrich the dataset and

improve the model's ability to understand and generate relevant responses. Once the training is complete, the performance of the model can be benchmarked against various metrics.

4.1.10. Prompt

Prompt engineering is a crucial and iterative process in the utilization of language models. It involves fine-tuning the language model's input to elicit the desired output, which often requires multiple iterations and careful adjustment of the prompt's wording, structure, and context. The quality and specificity of the prompt can significantly impact the effectiveness of the AI, making it an essential aspect of model deployment.

In our scenario, we explicitly requested the output in a structured CSV format, ensuring ease of use in further data processing and analysis.

The refined and final version of the prompt we utilized is as follows:

> "You are ChemiScope, an expert in extracting specific information about molecules from scientific papers. The text you receive is the copied text from the paper PDF. Your purpose is to identify and extract the following details for each compound discussed in the text. Each compound should have its separate output in CSV format:
>
> 1. Compound name (C), referred to as 'rdfs:label' in the text.
> 2. Bioactivity (B), referred to as 'nubbe:biologicalActivity' in the text.
> 3. Species (S) from which the natural products were extracted, referred to as 'nubbe:collectionSpecie' in the text.
> 4. Collection site (L), referred to as 'nubbe:collectionSite' in the text.
> 5. Isolation type (T), referred to as 'nubbe:collectionType' in the text.
>
> If there is no information about a specific compound or no compound name is given, print "No information" for that compound and do not include it in the CSV output. By doing so, you will greatly assist researchers and scientists in their work and contribute to the advancement of scientific knowledge."

4.2. Call

In the context of our exploration, the provided Python script serves as an interface with the OpenAI API, utilizing the GPT-3.5 language model, to process a dataset derived from hundreds of scientific articles provided in the scope of BiKE. These articles provide diverse information regarding more than 2,521 instances of natural product extraction, vital for the challenge set by BiKE. By storing this dataset in a Parquet file format.

The script operates through a sequence of orchestrated tasks. Initially, the data – encapsulated as a DataFrame - is extracted from the Parquet file. Subsequently, the data in the 'phrases' column undergoes a string conversion process, if not already in string format.

The script is attuned to the length of the text in the 'phrases' column and processes it accordingly. For shorter phrases, the script interfaces directly with the OpenAI API, using the

text as input for processing. However, for longer phrases, the script resorts to splitting the text into smaller segments, more manageable for the API's limitations on input length.

The model processes each segment independently, and the script accumulates the responses to replace the original content in the DataFrame. Concurrently, the script maintains the OpenAI API's rate limits. Once the processing of a predetermined amount of data is completed, the script halts its operation for 30 seconds. This break ensures the script's usage of the API remains within acceptable thresholds and avoids system overload.

Upon completion, the processed data is stored back into a Parquet file, maintaining the columnar format demanded by NatUKE for optimized read and write operations in the future.

This script provides a practical batch processing solution designed for the GPT-3.5 model, enabling efficient handling of large datasets in segments. While it's currently used to enhance the data quality for machine learning algorithms in Project NaTUKE, its flexible design allows for potential adaptations to other tasks. It demonstrates the utility of incorporating language models into data preprocessing pipelines.

4.3. Problems

The preprocessing of the extracted pdf-data confronted us with new obstacles. To get better results from the ChatGPT API, the data needed to be cleaned. All tokens that could not be read got replaced with a token accordingly. This may have worked in most cases, but there is still potential for missed errors that could not be identified due to the limited time frame and the amount of data. The generating of the prompt was an iterative process that got changed until the data represented the scope of the data we needed. There is still the opportunity left to generate an even more effective prompt or integrate parts that were not integrated into this work. Since the token limit of 4096 tokens could not be exceeded for the number of tokens contained in requests and responses while 90,000 tokens was the maximum amount per minute [4]. This led to a nine-hour call, which could have been a call with less time if the limitation would not be there. Another problem that was unreachable to our means was a general overload that sometimes appeared when the API was used by too many users [16].

5. Evaluation

The evaluation was performed using the official BiKE benchmark NatUKE [2].

5.1. Models

Following a short overview of the models used.

- **DeepWalk:** DeepWalk is an unsupervised machine learning algorithm used for learning latent representations of nodes in a network, which can be used for a variety of prediction tasks. The algorithm operates by treating the input graph as if it were a document, and the walks on this graph as if they were sentences. It begins by generating random walks on the graph, treating these walks as 'sentences'. These 'sentences' are then used to train a Skip-gram model, a popular word2vec model that predicts context words given a target

word. The result is a vector representation for each node in the graph that captures the structural information of the graph [17].

- **Node2Vec:** Node2Vec is an extension of the DeepWalk algorithm that introduces additional flexibility in the random walk procedure, allowing for better exploration of the graph structure. It does so by defining a flexible notion of a node's neighborhood and uses a biased random walk procedure that efficiently explores diverse neighborhoods. This yields embeddings that can better capture the connectivity patterns in a graph, making Node2Vec potentially more effective for tasks such as link prediction or node classification [18].
- **Metapath2Vec:** Metapath2Vec is an extension of the DeepWalk algorithm that considers the types of connections (edges) in a graph. In Metapath2Vec, random walks are guided by a metapath, a pre-determined sequence of node and edge types. This metapath-guided procedure can help capture the semantic and structural correlations in the graph and can be beneficial in heterogeneous graph structures where multiple types of nodes and edges exist. These guided walks are then used as input to a Skip-gram model to learn node embeddings [8].
- **Embedding Propagation on Heterogeneous Networks (EPHEN):** EPHEN is a method for learning node embeddings in heterogeneous networks, which are networks that contain different types of nodes and edges. The method operates by distributing an initial embedding across the network using a regularization function. This initial embedding is obtained using Sentence-BERT in a multilingual model. Each sentence in a paper is represented as an embedding, and the sum of all these embeddings is used as the initial embedding for the paper. The regularization function then propagates this initial embedding across the network, effectively distributing the information contained in the initial embedding to other nodes in the network. This results in a final embedding for each node that captures both the original information of the node and the information of its neighbors in the network. The EPHEN method is unsupervised, meaning it does not require any labeled data, and it can handle both text and structured data, making it highly versatile [9].

5.2. Results

During our research, we implemented a novel approach aimed at enhancing the performance of knowledge extraction algorithms. The methodology hinged on preprocessing with OpenAI's ChatGPT. As can be seen in Table 1, we compared the output from our approach with the initial results obtained from the NatUKE algorithm. The improvements in our results, showcased in Table 2, are quite noteworthy. The quantification of the changes in each category is documented in Table 3.

As Table 4 shows, Our innovative approach has produced a remarkable improvement in the performance of the DeepWalk, Node2Vec, and EPHEN algorithms compared to the NatUKE baseline. Specifically, the benefits were most noticeable in the extraction of bioactivity (B), collection site (L), and isolation type (T) for DeepWalk and Node2Vec. As for the EPHEN algorithm, considerable improvements were observed across all properties, with bioactivity (B) and isolation type (T) being the most significantly impacted.

Table 1
Results table for extracting: compound name (C), bioactivity (B), specie (S), collection site (L), and isolation type (T). Performance metric with the `hits@k` and k is respectively: 50, 5, 50, 20, and 1. The best results for each extraction are bold.

Property	DeepWalk			
	1st	**2nd**	**3rd**	**4th**
C	0.08	0.00	0.00	0.00
B	0.41	0.12	0.10	0.07
S	0.37	0.24	0.27	0.25
L	0.56	0.41	0.38	0.29
T	0.25	0.14	0.14	0.09

Property	Node2Vec			
	1st	**2nd**	**3rd**	**4th**
C	0.08	0.00	0.00	0.00
B	0.41	0.07	0.03	0.03
S	0.36	0.22	0.25	0.24
L	**0.57**	0.36	0.28	0.23
T	0.10	0.07	0.05	0.01

Property	Metapath2Vec			
	1st	**2nd**	**3rd**	**4th**
C	**0.10**	**0.08**	**0.09**	**0.20**
B	0.27	0.17	0.13	0.12
S	**0.40**	**0.41**	**0.42**	**0.44**
L	0.40	0.42	0.42	0.40
T	0.28	0.22	0.19	0.19

Property	EPHEN			
	1st	**2nd**	**3rd**	**4th**
C	0.09	0.02	0.03	0.04
B	**0.55**	**0.57**	**0.60**	**0.64**
S	0.36	0.24	0.29	0.30
L	0.53	**0.52**	**0.55**	**0.55**
T	**0.71**	**0.66**	**0.75**	**0.75**

Table 2
Results table for extracting with preprocessing with ChatGPT: compound name (C), bioactivity (B), specie (S), collection site (L), and isolation type (T). Performance metric with the `hits@k` and k is respectively: 50, 5, 50, 20, and 1. The best results for each extraction are bold.

Property	DeepWalk with ChatGPT			
	1st	**2nd**	**3rd**	**4th**
C	0.09	0.00	0.00	0.00
B	0.47	0.19	0.18	0.18
S	0.36	0.23	0.27	0.27
L	**0.56**	0.44	0.38	0.32
T	0.22	0.15	0.14	0.09

Property	Node2Vec with ChatGPT			
	1st	**2nd**	**3rd**	**4th**
C	0.08	0.00	0.00	0.01
B	0.46	0.11	0.07	0.05
S	0.38	0.22	0.25	0.24
L	**0.56**	0.39	0.31	0.23
T	0.11	0.05	0.04	0.02

Property	Metapath2Vec with ChatGPT			
	1st	**2nd**	**3rd**	**4th**
C	**0.11**	**0.08**	**0.10**	**0.18**
B	0.22	0.13	0.11	0.13
S	**0.41**	**0.41**	**0.44**	**0.40**
L	0.40	0.39	0.40	0.44
T	0.25	0.20	0.20	0.23

Property	EPHEN with ChatGPT			
	1st	**2nd**	**3rd**	**4th**
C	0.09	0.00	0.00	0.00
B	**0.62**	**0.69**	**0.69**	**0.69**
S	0.35	0.24	0.30	0.30
L	**0.56**	**0.60**	**0.62**	**0.64**
T	**0.76**	**0.77**	**0.80**	**0.81**

The empirical evidence suggests that the language understanding capabilities of advanced language models like ChatGPT, when used for preprocessing, can augment the performance of these knowledge extraction algorithms. This phenomenon can be particularly seen in homogeneous information networks, where the additional contextual understanding supplied by ChatGPT proves advantageous.

Table 3
Results table for extracting with preprocessing with ChatGPT: compound name (C), bioactivity (B), specie (S), collection site (L), and isolation type (T). Performance metric with the `hits@k` and k is respectively: 50, 5, 50, 20, and 1. The best results for each extraction are bold.

Property	**DeepWalk with ChatGPT**			
	1st	**2nd**	**3rd**	**4th**
C	+0.01	0.00	0.00	0.00
B	+0.06	+0.07	+0.08	+0.07
S	-0.01	-0.01	0.00	+0.02
L	0.00	+0.03	0.00	+0.03
T	-0.03	+0.01	0.00	0.00

Property	**Node2Vec with ChatGPT**			
	1st	**2nd**	**3rd**	**4th**
C	0.00	0.00	0.00	+0.01
B	+0.05	+0.04	+0.04	+0.02
S	+0.02	0.00	0.00	0.00
L	-0.01	+0.03	+0.03	0.00
T	+0.01	-0.01	-0.01	+0.01

Property	**Metapath2Vec with ChatGPT**			
	1st	**2nd**	**3rd**	**4th**
C	+0.01	0.00	+0.01	-0.02
B	-0.05	-0.04	-0.02	+0.01
S	+0.01	0.00	+0.02	-0.04
L	0.00	-0.03	-0.02	+0.04
T	-0.03	-0.02	-0.01	+0.04

Property	**EPHEN with ChatGPT**			
	1st	**2nd**	**3rd**	**4th**
C	0.00	-0.02	-0.03	-0.04
B	+0.07	+0.12	+0.09	+0.05
S	-0.01	0.00	-0.01	0.00
L	+0.03	+0.08	+0.07	+0.09
T	+0.05	+0.11	+0.05	+0.06

Table 4
Overview of the change in all categories and attempts.

Algorithm	DeepWalk	Node2Vec	Metapath2Vec	EPHEN
Improvement	+ 8,87 %	+ 6,55 %	- 2,24 %	+ 8,91 %

However, an unexpected outcome was the slightly diminished performance observed with the Metapath2Vec algorithm. A plausible explanation for this discrepancy could be the unique functioning of Metapath2Vec. Unlike the other algorithms, Metapath2Vec operates on heterogeneous information networks and employs random walks along predefined metapaths to generate paths [8]. These metapaths, which denote specific relationships between different entity types, are predetermined and may not benefit significantly from additional context provided by ChatGPT. The inherent structure and semantics of these networks might be such that the preprocessing information might not translate into beneficial paths, which could potentially account for the less efficient use of information provided by ChatGPT.

In summary, the performance of knowledge extraction algorithms, especially those used for homogeneous information networks, can be considerably improved through preprocessing with advanced language models such as ChatGPT. However, the case of Metapath2Vec reveals the necessity for further investigation to discern the most effective utilization of these models for heterogeneous information networks.

6. Conclusion & Future Works

This project demonstrates a proof of concept on how AI preprocessing, using the ChatGPT model, can enhance the NatUKE approach for generating knowledge graphs. Our efforts have yielded a proof of concept, laying a solid foundation for future explorations that seek to improve knowledge graph generation based on data preprocessed by AI.

One potential avenue for improvement lies in the adoption of a model capable of processing more tokens simultaneously, thus capturing a broader context from the data. With the recent public availability of advanced models that can handle larger token volumes, the scope of our project could be significantly expanded.

However, to ensure a smooth implementation of such models, a stable and dedicated connection is crucial. This would prevent any disruptions or compromises caused by concurrent API usage by other parties. While our project utilized a single model, future research should experiment with a variety of models to compare and contrast their effectiveness. Our proof-of-concept study underscores the potential of AI in knowledge graph generation, and we anticipate exciting advancements in this field as newer, more capable models are developed and deployed.

References

[1] D. J. Newman, G. M. Cragg, Natural products as sources of new drugs from 1981 to 2014, Journal Of Natural Products 79 (2016) 629–661.

[2] P. V. do Carmo, E. Marx, R. Marcacini, M. Valli, J. V. S. e Silva, A. Pilon, NatUKE: A Benchmark for Natural Product Knowledge Extraction from Academic Literature, in: 17th IEEE International Conference on Semantic Computing, IEEE, 2023.

[3] AKSW, First international biochemical knowledge extraction challenge, 2023. URL: https://aksw.github.io/bike/, accessed on 2023.06.20.

[4] OpenAI, Models, 2023. URL: https://platform.openai.com/docs/models, accessed on 2023.06.20.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020. arXiv:2005.14165.

[6] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014. URL: https://doi.org/10.1145/2623330.2623732. doi:10.1145/2623330.2623732.

[7] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, 2016. arXiv:1607.00653.

[8] Y. Dong, N. V. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, Association for Computing Machin-

ery, New York, NY, USA, 2017, p. 135–144. URL: https://doi.org/10.1145/3097983.3098036. doi:10.1145/3097983.3098036.

[9] P. do Carmo, R. Marcacini, Embedding propagation over heterogeneous event networks for link prediction, in: 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 4812–4821. doi:10.1109/BigData52589.2021.9671645.

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

[12] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, R. Socher, Ctrl: A conditional transformer language model for controllable generation, 2019. arXiv:1909.05858.

[13] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, 2018. arXiv:1802.05365.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. arXiv:1706.03762.

[15] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, Y. Zhang, Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. arXiv:2303.12712.

[16] OpenAI, Error codes, 2023. URL: https://platform.openai.com/docs/guides/error-codes, accessed on 2023.06.20.

[17] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[18] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.