Automatic Pseudocode Extraction at Scale

1st Levent Toksoz
Computer Science and Engineering
The Pennsylvania State University
University Park, Pennsylvania, 16802
lkt5297@psu.edu

2nd Gang Tan
Computer Science and Engineering
The Pennsylvania State University
University Park, Pennsylvania, 16802
gtan@psu.edu

3rd C. Lee Giles
Information Sciences and Technology
The Pennsylvania State University
University Park, Pennsylvania, 16802
clg20@psu.edu

***Abstract*—Pseudocode in a scholarly paper provides a concise way to express the algorithms implemented therein. Pseudocode can also be thought of as an intermediary representation that helps bridge the gap between programming languages and natural languages. Having access to a large collection of pseudocode can provide various benefits ranging from enhancing algorithmic understanding, facilitating further algorithmic design, to empowering NLP or computer vision based models for tasks such as automated code generation and optical character recognition (OCR). We have created a large pseudocode collection by extracting around 320,000 pseudocode examples from arXiv papers. This process involved scanning over 2.2 million scholarly papers, with 1,000 of them being manually inspected and labeled. Our approach encompasses an extraction mechanism tailored to optimize the coverage and a validation mechanism based on random sampling to check its accuracy and reliability, given the inherent heterogeneity of the collection. In addition, we offer insights into common pseudocode structures, supported by clustering and statistical analyses. Notably, these analyses indicate an exponential-like growth in the usage of pseudocodes, highlighting their increasing significance.**

***Index Terms*—clustering, latex, pseudocode**

I. Introduction

Pseudocode serves as an instrumental device, employed to express algorithms in a concise, syntactical constraint free format. Pseudocode also incorporates elements from both programming languages and natural languages, making it an ideal candidate to be used as an intermediate representation, bridging the gap between these languages. The elements incorporated from programming languages are often represented as universally recognized constructs employed throughout contemporary programming paradigms, thus highlighting the versatility of pseudocode and solidifying its role as a bridge between programming languages and natural languages. Given the importance of pseudocode, a large collection of pseudocode can enhance algorithmic understanding, facilitate algorithmic design, and empower NLP or computer vision based models for tasks such as automated code generation and optical character recognition (OCR). Some published works have explored the role of pseudocode in automated code generation tasks. A brief overview of these articles is presented in the related work section.

A notable example of a pseudocode dataset is the SPOC dataset Kulal et al. (2019) [1] with approximately 20,000 pseudocodes along with their implementations and test cases. The pseudocodes in the SPOC dataset were manually written by programmers contracted through Amazon Mechanical Turk. As such this dataset captures a limited scope of pseudocode examples, in stark contrast to the diverse range found in research papers such as those in arXiv, which ranges from high-level pseudocodes to ones resembling actual code. It should be noted that the main purpose of the SPOC dataset is to provide a training/testing data-set for pseudocode to code conversion. Other notable examples include datasets from Oda et al. (2015) [3] and Zavershynskyi, Skidanov, and Polosukhin (2018) [4]. The dataset provided by Oda et al. (2015) [3] offers around 16,000 manually written pseudocodes for statistical machine translation, and the one from Zavershynskyi, Skidanov, and Polosukhin (2018) [4] provides approximately 2,000 manually written pseudocodes for the program synthesis task.

Our intent is to create a large collection of pseudocode extracted from arXiv papers, encompassing a diverse spectrum of pseudocode representations. Thus our collection can be employed for a variety of tasks. For instance, it can be used as a benchmark for automated code generation models and provide training/testing data for bimodal machine learning models to extract information such as text and figures from PDFs by pairing the pseudocode with the texts describing the pseudocode in the PDFs. Here, we focus on pseudocode collection, while leaving its applications to future work.

In summary, our contributions are:

- a method for finding and extracting pseudocode from arXiv papers,
- a large collection of approximately 320,000 pseudocodes extracted from these papers, accompanied by 1000 PDFs with human-provided labels for evaluation,
- analysis of the increasing growth of pseudocode in arXiv papers,
- clustering pseudocode based on topics.

II. Related Work

Pseudocode stands as a foundational concept in computer science, frequently used to articulate algorithms in a concise manner and serving as a link between natural languages and coding. Its instructional potential is explored in various articles, such as those by Peltsverger and Debnath (2019) [6] and Odisho, Aziz, and Giacaman (2016) [7]. Pseudocode is also employed across multitude of tasks, showcasing its versatility in diverse applications. Work such as Zhang, Xu, Xiao, and Xue (2022) [8] utilizes the universality of the pseudocode to

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00061

obtain a binary code similarity measure. Additionally, the work by Mishra et al. (2023) [9] explores the use of pseudocode as a prompt for Large Language Models.

One of the most thoroughly investigated applications of pseudocode in scholarly papers is its use in translating pseudocode into code and generating pseudocode for a variety of software tasks. To that extent, Kulal et al. (2019) [1] presents a machine learning model that can translate pseudocode to C++ code and a SPOC dataset that contains roughly around 20000 programs along with their human-authored pseudocodes and test cases. Similar approaches are found in other work Oda et al. (2015) [3] and Zavershynskyi, Skidanov, and Polosukhin (2018) [4]. As an example Oda et al. (2015) [3] focuses on pseudocode-based statistical machine translation, presenting a collection of approximately 16,000 manually crafted pseudocodes. On the other hand, Zavershynskyi, Skidanov, and Polosukhin (2018) [4] addresses the program synthesis task, offering a dataset that includes roughly 2,000 manually written pseudocodes for this task.

Given the challenges associated with finding extensive pseudocode databases, much of the scholarly work on pseudocode to code generation relies heavily on the SPOC dataset Kulal et al. (2019) [1]. Zhong, Stern, and Klein (2020) [2] uses a hierarchical beam search method that concentrates on specific semantic and syntactic constraints inherent in a program to further improve the model Kulal et al. (2019) [1]. Other work Yasunaga and Liang (2020) [10] utilizes the compiler output to repair outputs generated by program synthesis tasks on the SPOC dataset Kulal et al. (2019) [1]. Others Shi, Bieber, and Sutton (2020) [12] and Xie, Ma, and Liang (2021) [11] adopt a transformer based model that undertakes pseudocode to code generation on a line-by-line basis, with the assistance of the SPOC dataset Kulal et al. (2019) [1]. The generation of pseudocode from code conversion is frequently influenced by variety of models, as explored by Feng et al. (2020) [20], Guo et al. (2020) [21], Xu, Alon, Neubig, and Hellendoorn (2022) [22], Fried et al. (2023) [23], Rozière et al. (2021) [24], Lu et al. (2021) [25] and covers tasks such as code synthesis, infilling, and docstring generation.

Additionally, pseudocode can be extracted from various sources, such as PDF documents. While existing work Nassar, Livathinos, Lysak, and Staar (2022) [14], Hu, Li, Cao, Meyerzon, and Zheng (2005) [17], Rane, Subramanya, Endluri, Wu, and Giles (2021) [19], Hou, Jochim, Gleize, Bonin, and Ganguly (2021) [15], Blecher, Cucurull, Scialom, and Stojnic (2023) [13] explore various methods for extracting different types of information from diverse origins, there exists a notable gap in the literature regarding the extraction of pseudocode content within documents.

III. Pseudocode Data

We extract pseudocodes from scholarly papers in the arXiv.[1] Approximately 10 TB of arXiv data is downloaded and stored across both Amazon S3 buckets and Google Cloud. In the arXiv dataset, there are 2.2 million PDF files, comprising roughly half of the total data size. The remaining data consists of supplementary files, such as images, figures, and LaTeX files. PDF files are retrieved using Google Cloud, while supplementary files such as images, figures, and LaTeX files are extracted through Amazon S3 buckets. It should be noted that not all papers have associated supplementary files. The number of submissions to arXiv has been growing exponentially, underlining the significant contributions made in recent years as indicated by the arXiv website [16].

[1]The dataset can be accessed via https://github.com/letoksoz/arxiv-pseudocode.

IV. Automatic Extraction

To automatically detect and extract the pseudocode, we designed an extraction pipeline tailored to capture a diverse range of extracted pseudocode examples. Our pipeline has the following stages: preprocessing, pseudocode detection, and pseudocode extraction. It scans over 2.2 million scholarly papers starting with year 1991 and ending in June of 2023. It should be noted that due to the large size of the dataset and the performance profile of the detection, extraction, and validation tools, running the pipeline requires a substantial amount of time, typically on the order of several days.

A. Preprocessing

Some of the supplementary files in Amazon S3 are either stored as ZIP files, contain ZIP files, or even include both. Each ZIP file is extracted until no more ZIP files remain. Papers stored in arXiv have unique identifiers. By matching these unique identifiers, papers linked to Amazon S3 files (i.e., LaTeX files) are combined with their corresponding Google Cloud files (i.e., PDFs) as a single folder to be processed.

B. Pseudocode Detection

To detect pseudocodes in each article, we first verify whether it contains LaTeX files. If such a file exists, we proceed to search for specific LaTex keywords: '\begin{algorithm}' and '\end{algorithm}' within the files. If an article's LaTeX files contain those keywords, it is forwarded to the extraction stage; otherwise, it is utilized to gather statistical information.

Some additional information about the papers are also stored, including PDF files themselves, supplementary files such as LaTeX and HTML files, extracted text snippets from PDFbox [18] and arXiv metadata. The arXiv metadata information includes the arXiv identifier of the paper, version of the PDF, the arXiv link, the abstract, the year in which it was uploaded to arXiv, the topic and subtopic of the article, and finally the title of the PDF. Our detection algorithm identified pseudocode in $141,939$ out of the 2.2 million papers as shown in table IV.

C. Pseudocode Extraction

In this stage, papers containing LaTeX files identified in the detection stage are processed. The pseudocode in these LaTeX files exhibit a heterogeneous structure due to the absence of a

TABLE I
SAMPLED COUNTS

Manually Inspected 1000 Papers	Number
Has Pseudocode	101
Does not have Pseudocode	899

TABLE II
FALSE POSITIVE AND FALSE NEGATIVE RATES

Type	Percentage
FPR	%0.6
FNR	%33.7

standardized LaTeX notation. For instance, \begin{equation} and \begin{align} might serve the same purpose. Therefore, it might be unrealistic to expect the extraction algorithm to cover all possible representations. As we will explain further in the validation and statistical analyses section, utilizing \begin{algorithm} tags to detect and extract pseudocode from LaTeX papers is reliable. Our analyses also indicate that the majority of the papers in arXiv have LaTeX files, underlining the significance of extracting information from LaTex files. For that reason, the extraction algorithm relies on \begin{algorithm} and \end{algorithm} tags. Specifically, our algorithm identifies the locations of those tags and then extracts the pseudocode found between them. When pseudocode includes references to other LaTex contents such as equations, we search for the corresponding label of that reference within the file and then extract it as supplementary information. Our extraction mechanism obtained $323,303$ pseudocodes and saved each as a JSON file, along with metadata information such as the arXiv identifier, any equations referenced by the pseudocode, and the year it was stored in arXiv.

V. VALIDATION

The validation aims to understand the accuracy of our pseudocode exaction mechanism. A false positive is a paper that does not contain pseudocode but our extraction extracted pseudocode from it. A false negative is a paper that does contain pseudocode but our extraction cannot extract pseudocode.

Due to the extensive size and characteristics of the dataset, it is not feasible to manually obtain the ground truth such as whether a paper contains pseudocode or not for all the papers in the dataset. To that extent, we utilized a sampling based approach. We uniformly sampled 1000 PDFs among all the scanned papers spanning from the year 1991 to 2023. Each sampled paper was then manually inspected to determine whether it contains pseudocode and labeled accordingly. The distribution of pseudocode counts within these inspected papers is shown in Table I. In addition to labeling each pseudocode, we store additional information about it and its associated PDF in a separate JSON file. This information includes details such the arXiv ID, and additional metadata about the paper.

By cross-checking the results obtained from our detection-extraction mechanism with this manually labeled sampled set, we computed the false negative and false positive rates presented in Table II. These results indicate that the extraction-detection mechanism, based on the '\begin{algorithm}' tag, is a reliable method for detecting and extracting pseudocodes in LaTeX. However, it could be improved, as it sometimes misses certain pseudocode patterns. Upon manual inspection of some of these overlooked patterns, we observed that there were manually structured pseudocodes with '\begin{enumerate}' tags and other pseudocodes embedded within paragraphs. Some papers also utilized '\begin{algorithm}' to set up problems rather than pseudocode descriptions. However, the overall number was very low within the sampled set, contributing to the low false positive rate of our extraction process.

TABLE III
NUMBER OF PAPERS

From year 1991 to year 2023	Number
Total papers	$2,285,111$
Papers with LaTex	$2,054,422$

TABLE IV
PAPERS WITH PSEUDOCODE

From year 1991 to year 2023	Number
Papers with keywords indicating the presence of pseudocode	$241,275$
Papers with tag "\begin{algorithm}"	$141,939$

VI. STATISTICAL ANALYSES

Results of our extraction mechanism are briefly summarized by Table III and Figure 1. Significantly, around 90% of the papers were accompanied by LaTeX files, reinforcing the choice to extract pseudocodes from LaTeX files instead of PDFs. Another notable result is the number of papers with pseudocode grows almost exponentially over the years, as shown by Figure 1, underlining the increasing significance of pseudocode. We hypothesize that this increasing trend correlates with the growing prominence of computer science subjects in arXiv, as well as the increased availability of powerful computing tools and growing computational capacity.

We also analyzed the papers that use keywords indicative of the presence of pseudocode, such as "Algorithm n", where n is a number, and "Pseudocode". Full list of words can be found in Table V. Note that given the popularity of the word "Algorithm", the algorithm keyword alone does not directly correlate with the presence of a pseudocode. To identify such papers, we perform a direct search within the PDF files using Apache PDFBox [18]. For the corresponding LaTeX source code, similar methodology mentioned in Pseudocode detection section is employed. PDFBox [18] extracts various information from the PDF, including text, formatting options, and metadata about the article. We conduct a search within the extracted text of PDF files for specific keywords such as 'Pseudocode' and 'Algorithm 1'. The number distribution of these papers are shown by Table IV and figure 2. Both keyword paper counts exhibit exponential-like growth. To validate whether the presence of these indicative words indicates the presence of

TABLE V
INDICATIVE WORDS.

Keyword	Searched Words
Pseudocode Algorithm	”Pseudocode”, ”pseudocode”, ”Pseudo-code”, ”pseudo-code” ”Algorithm N”, ”algorithm N”, ”Algorithm-N”, ”algorithm-N”, ”Algorithm:”, ”algorithm:”

TABLE VI
INDICATIVE WORD INSPECTION.

Has Pseudocode	Sampled 1000 Papers	Contains Indicative Keywords
Yes	101	75
No	899	20

pseudocode, we used our 1000 manually examined validation paper set. The results are shown in Table VI. It shows that the indicative words alone are relatively insufficient to reliably indicate the presence of pseudocode given their relatively large false positive count compared to our mechanism. However, false negative count decreases, enhancing the detection of various pseudocode types. Using the corresponding metadata information of each PDF that contains these indicative words, a category plot for each document is displayed in Figure 3, indicating that most of these keywords occur in computer science-themed papers.

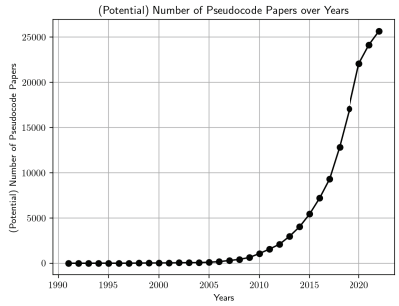

Fig. 1. Number of papers with LaTeX and ”\begin{algorithm}” tag.

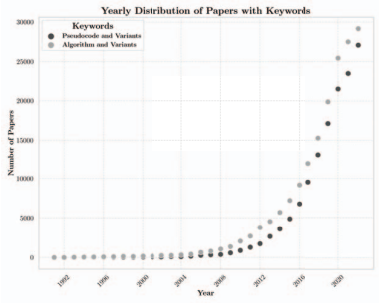

Fig. 2. Yearly Distribution of Papers with Keywords (See Table V for keyword details)

VII. CLUSTERING

We investigate how the topics of pseudocodes evolve over time by clustering. Using the arXiv metadata, the topics of each paper containing pseudocode can be plotted, similar to Figure 3. However, arXiv topics are often too broad and do not precisely represent the pseudocode topics. For instance, a biology-themed paper may include pseudocode related to computer science or graph theory. To address this, we have designed our own clustering mechanism based on the text snippets where pseudocode is mentioned.

The clustering mechanism utilizes text snippets that reference the pseudocode. These snippets are created as a result of our reference detection mechanism and are cleaned from irrelevant LaTeX syntax. Since the number of papers containing pseudocode before the year 2010 is negligible, as indicated in Figure 1, we exclusively utilized text references from the year 2010 onward. Additionally, common English stop words and non-instructive words for topic modeling, such as ’use’, ’employ’, and ’indicate’ are filtered out. To consolidate different variations of the same word, such as ’decode,’ ’decoding,’ and ’decoded,’ into a single representation, each word undergoes stemming. The Term Frequency-Inverse Document Frequency Vectorizer (TF-IDF) is utilized to obtain vector representations for each text snippet. Terms that appear in more than 85 percent and less than 0.02 percent of documents are disregarded.

A. Reference Detection Algorithm

The reference detection mechanism operates as follows: Whenever pseudocode includes a label, our mechanism attempts to locate that label. Given that labels can be represented in various ways in LaTeX, we employ regular expressions (regex) to accommodate the diverse representations of labels, which also account for special characters. Once a label is found, the mechanism dynamically generates the associated reference tag. These reference tags are tailored to different types of reference words (e.g., equations, algorithms, theorems, etc.), which typically have an identifying segment followed by ’ref’ (e.g., algref, eqref). To handle the variations specific to each type of reference, we employ regular expressions (regex) for pattern matching. The generated references are then searched within the entire directory, as these references could potentially be located in other LaTeX files. Finally, when a reference tag is identified, we mark a span of 1200 characters before and after that tag. If the 300-character span around these marked locations contains a character indicating the end of a sentence, we extract the text up to that point. However, if no such characters are found, we retain the original span of 1200 characters. Our mechanism successfully extracted the references of the majority of pseudocodes. The ones that were not extracted either lack associated references or possess a reference tag that is too complex for regular expressions to handle.

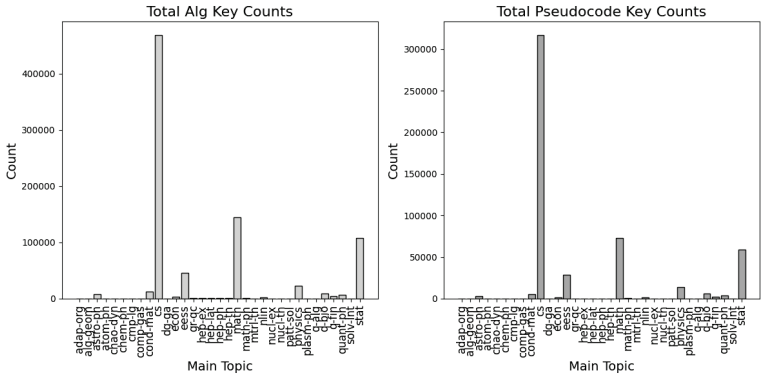

Fig. 3. Category Distribution of Papers with Keywords (See Table V for keyword details)

B. Cleaning Extracted Text

The extracted reference texts for each pseudocode are presented in LaTeX format, containing LaTeX keywords, special symbols, comments, and mathematical symbols. To clean the extracted reference texts, removal processes are implemented for different LaTeX elements: LaTeX keywords by detecting \, comments via %, mathematical symbols through $, and special words containing a single character on either side of _ (or both sides or none).

C. Topic Modelling

To investigate how the topics of pseudocodes evolve over time, we employed Latent Dirichlet Allocation (LDA) to perform soft clustering of the data, allowing each pseudocode reference to have multiple topic representations. To obtain consistency across years, the number of topics is set to 10, which might not be optimal for certain years given the significant variation in the number of pseudocodes across years as shown in figure VII. To understand the representation of each topic, we provide the top 5 most representative words for each topic, listed in descending order of relevance. The clustering results are displayed in Table VII. From Table VII, highlighted in bold, starting in 2010 and continuing thereafter, there is a noticeable emphasis on pseudocodes related to graph algorithms. We can also observe a diminishing prominence of physics-related pseudocode. This transition can be observed from Table VII, marked in red. Initially, physics-related words begin to be superseded by probability-related terms within the same cluster as years progress from 2010 to 2014. Furthermore, after the year 2014, physics-related terms cease to appear. Starting from 2014 and continuing up to 2023, we observe machine learning as one of the prevailing topics for pseudocode, highlighted in blue in Table VII.

TABLE VII
PSEUDOCODE TOPIC CLUSTERS FOR SELECTED YEARS

Years	Cluster Number	Top 5 words in stemmed format
2010	0	polici, stage, edg, algorithms, channel
2010	1	messag, node, root, max, tree
2010	2	particl, posterior, filter, sampl, likelihood
2010	3	protocol, request, receiv, node, neighbor
2010	4	bit, specif, read, string, regular
2010	5	matrix, optim, iter, vector, estim
2010	6	**node, edg, graph, polynomi, vertex**
2010	7	threshold, delet, reconstruct, spars, recurs
2010	8	insid, bit, residu, embed, exist
2010	9	thread, processor, reconstruct, queue, termin
...	...	...
2016	0	polynomi, satisfi, state, return, theorem
2016	1	optim, iter, converg, solv, gradient
2016	2	**node, edg, graph, vertex, vertic**
2016	3	block, array, item, parallel, cell
2016	4	cluster, frame, assign, merg, distanc
2016	5	agent, action, state, transit, flow
2016	6	matrix, probabl, rank, sampl, column
2016	7	estim, sampl, matrix, distribut, approxim
2016	8	layer, pixel, bit, encod, code
2016	9	train, learn, featur, user, network
...	...	...
2023	0	attack, adversari, game, perturb, box
2023	1	client, train, server, featur, dataset
2023	2	iter, converg, solv, optim, matrix
2023	3	**edg, node, graph, vertex, vertic**
2023	4	train, loss, sampl, learn, network
2023	5	sampl, estim, distribut, simul, probabl
2023	6	path, robot, search, state, optim
2023	7	regret, bound, proof, theorem, reward
2023	8	polici, agent, action, reward, state
2023	9	node, cluster, matrix, oper, block

VIII. CONCLUSIONS

We developed a pseudocode collection pipeline that we utilized to create a large collection of pseudocode examples, totaling approximately 320,000. Due to the significant heterogeneity of the collection, we have built a sampling-based validation mechanism to ensure the reliability of our pipeline. Additionally, 1000 papers are manually inspected and labeled based on whether they contain pseudocode, along with their corresponding lines and supplementary information. Moreover, clustering techniques, that are specifically LDA reveal thematic structures in a large collection of pseudocodes. Significantly, our findings reveal an exponential growth in the utilization of pseudocodes over time, with a particular focus

on pseudocodes related to graph algorithms emerging as a predominant theme.

IX. Future Work

Our dataset has the potential to serve as a valuable resource for a wide range of applications such as empowering NLP or computer vision based models and enhancing algorithmic understanding, opening up avenues for diverse future research.

In this dataset, each pseudocode is linked to its respective arXiv identifier. Leveraging these identifiers, connections between LaTeX-formatted pseudocodes and their corresponding PDFs can be established. By using these pairs as training and testing data for bimodal machine learning models, we can partially automate the extraction of information, including text and figures, from the PDFs. Additionally, despite the prevalence of LaTeX files in the arXiv, we have also identified around $25,000$ papers without LaTeX files. These papers are potential candidates that may contain pseudocode and could greatly benefit from such an automated process. Moreover, we manually inspected 1000 papers and labeled them based on whether they contain pseudocode, along with their supplementary information.

There are other valuable uses of our dataset. If we convert our LaTeX-formatted pseudocodes into a more text-like format, they could serve as testing and benchmarking data for automated code generation tasks. This can be achieved to some extent using tools such as LaTeXML Project (2022) [5]. Such tasks involve converting pseudocode to actual code or utilizing pseudocode as an intermediary for natural language to code or code to natural language translation.

Other future research would to be to build a focused search that permits the indexing and searching of pseudocode in order to facilitate the use and discovery of related pseudocode.

X. Acknowledgements

The arXiv is gratefully acknowledged for providing access to documents with pseudocode and their latex versions.

References

[1] S. Kulal, P. Pasupat, K. Chandra, M. Lee, O. Padon, A. Aiken, P. Liang, "SPoC: Search-based Pseudocode to Code," *CoRR*, vol. abs/1906.04908, 2019. http://arxiv.org/abs/1906.04908

[2] R. Zhong, M. Stern, D. Klein, "Semantic Scaffolds for Pseudocode-to-Code Generation," *CoRR*, vol. abs/2005.05927, 2020. https://arxiv.org/abs/2005.05927

[3] Y. Oda, H. Fudaba, G. Neubig, H. Hata, S. Sakti, T. Toda, S. Nakamura, "Learning to Generate Pseudo-Code from Source Code Using Statistical Machine Translation," *2015 IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 574-584, 2015. https://api.semanticscholar.org/CorpusID:15979705

[4] M. Zavershynskyi, A. Skidanov, I. Polosukhin, "NAPS: Natural Program Synthesis Dataset," *arXiv preprint arXiv:1807.03168*, 2018.

[5] "LaTeXML Project", "LaTeXML: A LaTeX to XML/HTML/MathML Converter," https://math.nist.gov/~BMiller/LaTeXML/, 2022.

[6] S. Peltsverger, S. Debnath, "Instructional Pseudocode Guide to Teach Problem-Solving," *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, https://doi.org/10.1145/3304221.3325581, ACM, New York, NY, USA, 2019.

[7] O. Odisho, M. Aziz, N. Giacaman, "Teaching and learning data structure concepts via Visual Kinesthetic Pseudocode with the aid of a constructively aligned app," *Computer Applications in Engineering Education*, vol. 24, no. 6, pp. 926–933, Wiley Online Library, 2016.

[8] W. Zhang, Z. Xu, Y. Xiao, Y. Xue, "Unleashing the power of pseudo-code for binary code similarity analysis," *Cybersecurity*, vol. 5, 2022, https://doi.org/10.1186/s42400-022-00121-0.

[9] M. Mishra, P. Kumar, R. Bhat, R. Murthy, D. Contractor, S. Tamilselvam, "Prompting with Pseudo-Code Instructions," *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15178–15197, Association for Computational Linguistics, 2023, https://aclanthology.org/2023.emnlp-main.939.

[10] M. Yasunaga, P. Liang, "Graph-Based, Self-Supervised Program Repair from Diagnostic Feedback," *Proceedings of the 37th International Conference on Machine Learning*, JMLR.org, 2020.

[11] S. M. Xie, T. Ma, P. Liang, "Composed Fine-Tuning: Freezing Pre-Trained Denoising Autoencoders for Improved Generalization," *Proceedings of the 38th International Conference on Machine Learning*, pp. 11424–11435, PMLR, 2021, https://proceedings.mlr.press/v139/xie21f.html.

[12] K. Shi, D. Bieber, C. Sutton, "Incremental Sampling Without Replacement for Sequence Models," *Proceedings of the 37th International Conference on Machine Learning*, pp. 8785–8795, PMLR, 2020, https://proceedings.mlr.press/v119/shi20a.html.

[13] L. Blecher, G. Cucurull, T. Scialom, R. Stojnic, "Nougat: Neural Optical Understanding for Academic Documents," *arXiv preprint arXiv:2308.13418*, 2023.

[14] A. Nassar, N. Livathinos, M. Lysak, P. Staar, "TableFormer: Table Structure Understanding with Transformers," *arXiv preprint arXiv:2203.01017*, 2022.

[15] Y. Hou, C. Jochim, M. Gleize, F. Bonin, D. Ganguly, "TDMSci: A Specialized Corpus for Scientific Literature Entity Tagging of Tasks Datasets and Metrics," *Proceedings of the 16th conference of the European Chapter of the Association for Computational Linguistics*, Online, 2021.

[16] "ArXiv", "ArXiv Monthly Submissions Statistics," https://arxiv.org/stats/monthly_submissions

[17] Y. Hu, H. Li, Y. Cao, D. Meyerzon, Q. Zheng, "Automatic Extraction of Titles from General Documents Using Machine Learning," *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 145–154, ACM, New York, NY, USA, 2005, https://doi.org/10.1145/1065385.1065418.

[18] "The Apache Software Foundation", "Apache PDFBox Projekt," http://pdfbox.apache.org/, 2012.

[19] C. Rane, S. M. Subramanya, D. S. Endluri, J. Wu, C. L. Giles, "ChartReader: Automatic Parsing of Bar-Plots," *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pp. 318-325, 2021, https://doi.org/10.1109/IRI51335.2021.00050.

[20] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, M. Zhou, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," *arXiv preprint arXiv:2002.08155*, 2020, https://arxiv.org/abs/2002.08155.

[21] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, M. Tufano, S.K. Deng, C. Clement, D. Drain, N. Sundaresan, J. Yin, D. Jiang, M. Zhou, "GraphCodeBERT: Pre-training Code Representations with Data Flow," *arXiv preprint arXiv:2009.08366*, 2020, https://arxiv.org/abs/2009.08366.

[22] F.F. Xu, U. Alon, G. Neubig, V.J. Hellendoorn, "A Systematic Evaluation of Large Language Models of Code," *arXiv preprint arXiv:2202.13169*, 2022, https://arxiv.org/abs/2202.13169.

[23] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih, L. Zettlemoyer, M. Lewis, "InCoder: A Generative Model for Code Infilling and Synthesis," *arXiv preprint arXiv:2204.05999*, 2023, https://arxiv.org/abs/2204.05999.

[24] B. Rozière, J. Zhang, F. Charton, M. Harman, G. Synnaeve, G. Lample, "Leveraging Automated Unit Tests for Unsupervised Code Translation," *ArXiv preprint ArXiv:2110.06773*, 2021, https://arxiv.org/abs/2110.06773.

[25] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tufano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S.K. Deng, S. Fu, S. Liu, "CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation," *arXiv preprint arXiv:2102.04664*, 2021, https://arxiv.org/abs/2102.04664.