Bib-A First Steps: The Computational Thesaurus

Elton Cardoso do Nascimento
e233840@dac.unicamp.br

Paula Dornhofer Paro Costa
paulad@.unicamp.br

Departamento de Engenharia de Computação e Automação (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)

Abstract

The rapid increase in publications makes it difficult to learn and keep up with recent topics, making it more difficult to generate impactful research for society. Recent advances in the area of text processing using LLMs, as well as the use of already consolidated techniques such as the creation of thesauri, can aid with this problem. In this paper, we present Bib-A, our work-in-progress methodology for using LLMs, and thesauri for bibliographic annotation.

Keywords — bibliographic research, large language models, thesauri

1. Introduction

The number of new publications per year in the computer science field grows each year [1]. As a result, understanding the state of the art in a field becomes more challenging: what is the latest knowledge on a subject and how can it be studied and applied effectively? Even when done by review articles, they fail to cover some areas of interest and remain up-to-date. Grasping the current state of the art is essential for conducting impactful research. Doing so effectively and efficiently can enhance and accelerate the influence of science on society.

Recently, we have also witnessed the rise of natural language processing using Large Language Models (LLM), artificial intelligence models capable of processing text for tasks not previously seen [2]. With them, the speed at which we can process text with some semantic consideration has increased. From a more traditional approach, is common in databases and libraries the usage of "thesauri" (plural of thesaurus), hierarchical organizations of keywords from a certain domain used in information retrieval systems for searching documents [3].

In this work, we propose the "Bibliography Annotator" (Bib-Annotator or Bib-A), a system using LLMs to carry out bibliographic search and annotation, listing each source and adding information such as summaries and assessments of its relevance. This paper presents the first steps of this working-in-progress project: gathering the article's texts and getting a picture of what the topics they cover and how to say if they talk about relevant topics. In Section 2, we present how we conducted this project, followed by the ethical considerations in Section 3. Section 4 presents our results, concluded by Section 5.

2. Methods

The Bib-A system is organized in three parts. The "Publication Pipeline" handles obtaining publications and extracting their texts, starting from a search query defined by the user and resulting in texts extracted from publications found using these words. The "Keyword Pipeline" is responsible for proposing keywords from the texts and creating a tree-like hierarchical structure from them (the thesaurus). The last stage, "K-P Scoring Pipeline", scores the membership of a publication in a keyword, using an LLM to analyze each publication-keyword pair one at a time and giving a membership score from 0 to 10.

For testing our pipelines, we defined the initial search query as "generative artificial intelligence", gathering the first 100 publication results.

For more details, Appendix A details each pipeline step, Appendix B presents all the created prompts for LLMs and Appendix C details the computer environment.

3. Ethical considerations

The possible impacts and risks of this project are still an open question, needing further studies to determine them, including its possible biases. One initial possible problem is the unwanted shape of future publications, inducing changes in their writing style and organization and decreasing stylistic diversity in publications. This can happen if authors try to increase the relevance of their work for this tool. Whether this is a problem depends on how future steps of this project

are carried out, as well as its future popularity.

There is also concern about the environmental cost of carrying out this type of processing, which still needs to be profiled.

4. Results and Discussion

At the "Publication Pipeline", 100 publications were selected, but only 32 were downloaded due to access protection. Manually inspecting the extracted texts, we also encountered some problems, with some PDFs being wrongly read, resulting in incorrect text orders.

Initially, 547 keywords were proposed, with the first unification for keywords with the same name resulting in 364 keywords. The final tree has 127 keywords. Below we have an example of a final keyword:

*"**(56) Generative Artificial Intelligence:** A type of artificial intelligence that can generate content, such as text, images, audio, code, videos, and synthetic data, using machine learning algorithms and large datasets, enabling systems to create new content without specific training."*

The first resulting tree shows problems as irrelevant keywords for the topic of interest, for example, *"**(2) Quantitative Estimate of Drug-Likeness (QED)**"*, because some of the publications are related to the initial query but also involve other topics. There is a sub-tree starting at node *"**(125) Algoritmik Paradigma Değişikliği**"* with a whole different language (Turkish) because no language filter or translation operation was done at any step. But, the major problem is the existence of incorrect hierarchical relationships, either conceptually such as *"**(64) Deep Learning**"* → *"**(63) Machine Learning**"*, or because they are disjoint themes, *"**(68) Art Theory**"* → *"**(65) Neural Network**"*. By manually inspecting the operations performed, we also found cases of nodes being incorrectly unified when they represent distinct concepts. Appendix D presents this intermediary tree for viewing.

We observed that the builder gets confused mainly when operating nodes of very different themes. Because of that, the posterior filter of "What are the keywords relevant to me?" uses a query keyword chosen from the tree, in this case, *"**(56) Generative Artificial Intelligence**"*. All the keywords with cosine similarity above a threshold (0.4) and their children subtree are included, giving us more usable resulting trees (Figure 1 and Appendix D). The sub-trees still have some issues reported in the global tree, but they are more coherent and have more useful keywords. Even with these problems, the sub-trees can quickly present an overview of the topics related to the area.

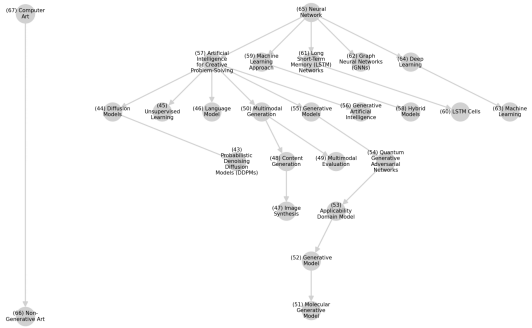

Figure 1: Thesaurus sub-trees starting at nodes 67 (left) and 65 (right).

Directing our attention to the membership scores, more individually, the keyword with the greatest average Membership Score across all publications was *"**(56) Generative Artificial Intelligence**"* ($\mu = 9.4$, $min = 8$), coherent since this is the initial search key, by assumption all publications are related to it. Curiously, the lowest average score keyword was *"**(66) Non-Generative Art**"* ($\mu = 0$, $min = 0$), the "antonym-related" of the other keyword, that no publication was considered related. Appendix E provides a more detailed analysis of the Membership Scores, as well as a comparison in relation to cosine similarity.

5. Conclusions

The potential of LLM usage to assist in research, specifically in bibliography discovery, is still to be seen. This project undertakes a first exploratory study of what might be possible. Although the current results seem promising, there is still a lack of rigor in evaluating them, with more broad experiments and metrics. Some problems already appeared, such as the need for a better scraper of publication on the internet. More robust thesaurus construction is also needed, which could possibly benefit from further study and application of already consolidated techniques and concepts in the area.

We plan to expand the tool features. A graphic interface for exploring the generated data could be useful. Annotating more data from the publications, such as extracting their workflows, applying and proposing techniques and datasets, and summarizing their fundamental information, like the research question/goal, methodology, and contributions, may facilitate bibliographic research in the future.

Acknowledgements

This project was supported by the brazilian Ministry of Science, Technology and Innovations, with resources from Law nº 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024 -10

No text generation tools were used in writing this text, but translation tools and revision-only were used.

References

[1] T. dblp team, "dblp: Publications per year." [Online]. Available: https://dblp.org/statistics/publicationsperyear.html

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] M. Shintaku, D. M. M. Sabbag, M. Costal, and R. d. V. d. Menêses, "Guia sobre a construção de tesauros." [Online]. Available: https://labcotec.ibict.br/omp/index.php/edcotec/catalog/view/24/22/110

[4] G. LLC, "Google Scholar." [Online]. Available: https://scholar.google.com.br/

[5] Publishers International Linking Association, Inc, "Crossref." [Online]. Available: https://www.crossref.org/

[6] Unstructured, "Unstructured-IO/unstructured," Oct. 2024, original-date: 2022-09-26T21:53:41Z. [Online]. Available: https://github.com/Unstructured-IO/unstructured

[7] A. D. et al., "The Llama 3 Herd of Models," Aug. 2024, arXiv:2407.21783. [Online]. Available: http://arxiv.org/abs/2407.21783

[8] W. K. et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[9] B. AI, "BerriAI/litellm," Oct. 2024, original-date: 2023-07-27T00:09:52Z. [Online]. Available: https://github.com/BerriAI/litellm

[10] E. C. d. Nascimento, "EltonCN/toolpy," Sep. 2024, original-date: 2024-06-10T16:35:48Z. [Online]. Available: https://github.com/EltonCN/toolpy

[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

A. Detailed Pipeline

The Figure 2 shows the full processing pipeline created in this project. Below, each step is presented in detail.

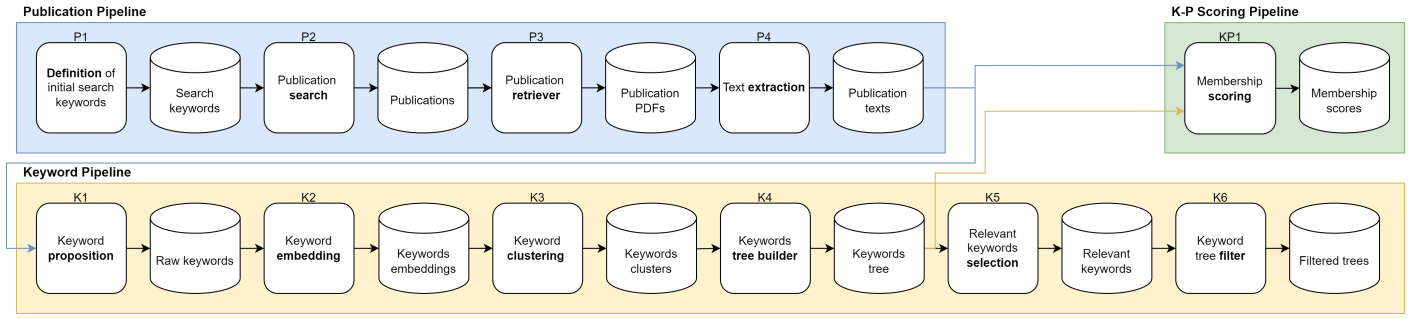

Figure 2: Project Workflow: numbered processing steps and data products

Publication Pipeline

P1 - Definition of initial search keywords: the user defines the search keywords for gathering publications.

P2 - Publication search: searches for the publications in Google Scholar, retrieving name, authors and publication year [4]

P3 - Publication retriever: uses the publication metadata for finding its DOI using Crossref and downloading from the DOI target [5]

P4 - Text extraction: text is extracted from the publication PDF file using unstructured open source version with "hi_res" strategy [6].

Keyword Pipeline

K1 - Keyword proposition: uses a LLM for proposing keywords from the full publication text containing a name and a definition. When two or more keywords with exact same name are proposed, they are unified, generating a new definition from their definitions.

K2 - Keyword embedding: calculates a numerical representation for each keyword.

K3 - Keyword clustering: creates a hierarchical clustering using the cosine similarity of the keywords, that is, a binary-tree in which the keywords are placed in the leaves, and keywords with similar representations are grouped into nearby branches.

K4 - Keyword tree builder: further process the three non-leaf nodes, from bottom to top nodes, doing three possible operations processing their children keywords. It checks for the operations applicability in sequence, if the previous one has not been carried out. Unification: when the keywords represents the same concept, a new unified definition is generated for the node and their children are removed. Promotion: when one of the keywords has a hierarchical relationship with the other, promotes the child node to it's parent. Meta-keyword generation: creates a new keyword that represents the two children.

K5 - Relevant keywords selection: the user needs to analyze the generated tree and select the relevant keywords for its research.

K6 - Keyword tree builder: filters the tree, searching for keywords near to the relevant ones using the embeddings cosine similarity, and generates the final trees with only relevant keywords and its children.

K-P Scoring Pipeline

KP1 - Membership scoring: scores the membership of a publication in a keyword, using a LLM to analyze each publication-keyword pair one at a time and giving a membership scores in 0-10.

B. Prompts

This section presents all the prompts used in this project. They are organized by the section where they are used.

Each prompt group contains a "Answer Format", how the LLM must respond, and the "System Prompt" and "User Prompt", sent one after the other, with the corresponding roles. The contents indicated by curly braces ("{example}") in the user prompts are filled with the data for the desired operation

B.1. K1 - Keyword proposition

B.1.1. Keyword Proposer

Proposes keywords for a text.

Answer Format:

keywords : list[Keyword]

where:

Keyword{ name : str, definition : str }

System Prompt:

"You are a keyword proposer, proposing keywords relevant to a text and returning in JSON. The JSON must use the schema: {'keywords': [{'name':'str', 'definition':'str'}, {'name':'str', 'definition':'str'}, ...]}.

Where 'name' is the keyword name, and 'definition' its definition, that must be in general form, not linking the description to the specific text.

Please use a valid JSON format."

User Prompt:

"{text}

Given the above text, propose some relevant keywords to it, with its definitions."

B.1.2. N-Keyword Unifier Direct

Create a unique defition for keywords that express the same idea.

Answer Format:

name : str
definition : str

System Prompt:

"You are a keyword unifier, proposing a unified name and definition for keywords that express the same idea, and returning in JSON. The JSON must use the schema: {'name':'str', 'definition':'str'}.

Where 'name' is the unified keyword name, and 'definition' the combination of definitions, and should cover the largest number of coherent elements from all definitions.

Please use a valid JSON format."

User Prompt:

"{keywords}

Provide a unified name and a combined definition uniting the maximum number of coherent elements from the keywords."

B.2. K4 - Tree Builder

B.2.1. Keyword Unifier

Verifies if two keywords are the same, and create e unique defition for it.

Answer Format:

rules_used : list[int]
reasoning : str
are_the_same : bool
name : str
definition : str

System Prompt:

"You are a keyword unifier, proposing a unified name and definition for two keywords if and only if express the same concept, and returning in JSON. The JSON must use the schema: {'rules_used':[int, int, ..], 'reasoning':'str', 'are_the_same': bool, 'name':'str', 'definition':'str'}.

The task objective is to detect keywords duplicates, not to unify different keywords in a umbrella keyword. Different concepts should not be unified.

Rules for telling keywords ARE NOT THE SAME concept:
1. If A is a subset/subtype/subfield of B, they are not the same concept
2. If A represents a more specific concept than B, they are not the same concept
3. If A and B are distinct elements of the same set, they are not the same concept
4. If A and B are different concepts, but used for the same task, they are not the same concept
5. If A and B are closely related, but different concepts, they are not the same concept
6. If A and B talk about the same field, but with different methods, they are not the same concept
7. If A and B present similar concepts with different focuses (for example, one is the process of carrying out the other), they are not the same concept
8. If A is a component of B, they are not the same concept
9. If A is a property of B, they are not the same concept
10. If A is a application of B, they are not the same concept
11. If A is a example of B, they are not the same concept

Rule for telling keywords ARE THE SAME concept: 12. If A definition is more detailed than B, but they represent the same concept, they are the same concept

The NOT THE SAME concept rules takes precedence over the ARE THE SAME concept rule.

First, list the rules used to unify or differentiate the keywords in 'rules_used'. Than, 'reasoning' is a discussion about the meaning of the two keywords, explaining why they are or are not the same concept, citing the rules. 'are_the_same' is true if the keywords are the same, and false otherwise. 'name' and 'definition' are optional and must be provided, only and if only, 'are_the_same' is true, where 'name' is the unified keyword name, 'definition' is the combination of definitions and should cover the largest number of coherent elements from both definitions.

Please use a valid JSON format."

User Prompt:

"Keyword 1
Name: {keyword1_name}
Definition: {keyword1_definition}

Keyword 2
Name: {keyword2_name}
Definition: {keyword2_definition}

Is the previous keywords expressing the same concept? If so, provide a unified name and a combined definition uniting the maximum number of coherent elements from both keywords."

B.2.2. Meta-Keyword Proposer

Creates a keyword that describes keywords.

Answer Format:

reasoning : str
name : str
definition : str

System Prompt:

"You are a keyword proposer, proposing a keyword that define a set that encompasses other concepts defined by other keywords, and returning in JSON. The JSON must use the schema: {'reasoning':'str', 'name':'str', 'definition':'str'}.

Where 'reasoning' is a discussion about the meaning of the two keywords, explaining they relation, similarities and differences. 'name' is the keyword name, and 'definition' the keyword definition, they must encompass all the keywords that will be part of this group.

Please use a valid JSON format."

User Prompt:

"{keywords}

Provide a keyword name and definition for the above keyword group."

B.2.3. Superset Verifier

Verifies if one keyword is is hierarchically above another.

Answer Format:

rules_used : list[int]
reasoning : str
keyword1_is_superset : bool
keyword2_is_superset : bool

System Prompt:

"'You verifies if one keyword from a group can be a semantic superset for another, and returns in JSON. The JSON must use the schema: {'rules_used':[int, int, ..], 'reasoning':'str', 'keyword1_is_superset': bool, 'keyword2_is_superset':bool}.

A semantic superset means that the other keyword is a subset/subtype/subfield.

Rules for detecting supersets, in precedence order:
1. If B is a subfield of A, A is a semantic superset
2. If the definition of A is broader than B, A is a semantic superset
3. If B implements A, A is a semantic superset
4. If B is a type of implementation of A, A is a semantic superset
5. If B is a component used in A, A is a semantic superset
6. If B is more specific than A, A is a semantic superset

First, list the rules used to classify the superset in 'rules_used'. Than, 'reasoning' is a discussion about the meaning of the two keywords, explaining why one of them is or not a superset. 'keyword1_is_superset' or 'keyword2_is_superset' is true if one of the keywords is a superset. Both is false if none of the keywords is a superset. Both cannot be true.

Please use a valid JSON format."

User Prompt:

"Keyword 1
Name: {keyword1_name}
Definition: {keyword1_definition}

Keyword 2
Name: {keyword2_name}
Definition: {keyword2_definition}

Is one of the keywords a superset for another? Which one?"

B.3. KP1 - Membership scoring

B.3.1. Keyword Membership Computer

Computes the belonging of a text inside a keyword.

Answer Format:

reasoning : str
score : int

System Prompt:

"'You are a keyword membership computer, computing the belonging of a text inside a keyword, and returning in JSON. The JSON must use the schema: {'reasoning': 'str', 'score': float}.

Where 'reasoning' must be the reasoning for giving the score relating the text to the keyword (if there is a relation). The score must be between 0 and 10, up to 1 decimal place and following the scale:

0 - keyword totally unrelated to the text

2 - keyword explicit present in the text, but unrelated to it

5 - keyword not present in the text but related to it, but not the main topic

7 - keyword explicit presented in the text and related to it, but not the main specific topic

8.5 - keyword not explicitly presented but main topic of the text

10 - keyword explicitly presented and main specific topic of the text

Please use a valid JSON format."

User Prompt:

"{text}

Given the above text, scores its belonging to the following keyword:

Keyword: {name}
Definition: {definition}"

C. Computational Environment

For researchers who wish to implement similar applications in the future, we present here the computational environment used.

We used two Nvidia A100 80 GB in this project, running 2 instances of Llama 3.1 8B Instruct [7] each in vLLM servers [8] and routing requests using LiteLLM [9]. The LLM tools where created using ToolPy [10], and all uses a JSON-constrained output. Text embeddings where creates using Sentence Transformers all-MiniLM-L6-v2 model [11].

Since Bib-A is still a work is progress, the code and data is not yet considered public-ready and there is no available repository. As soon as the project is considered ready, the code will be made available at the H.IAAC GitHub organization (`https://github.com/H-IAAC/`).

D. Additional Generated Thesauri

The intermediary resulting tree is show in Figure 3.

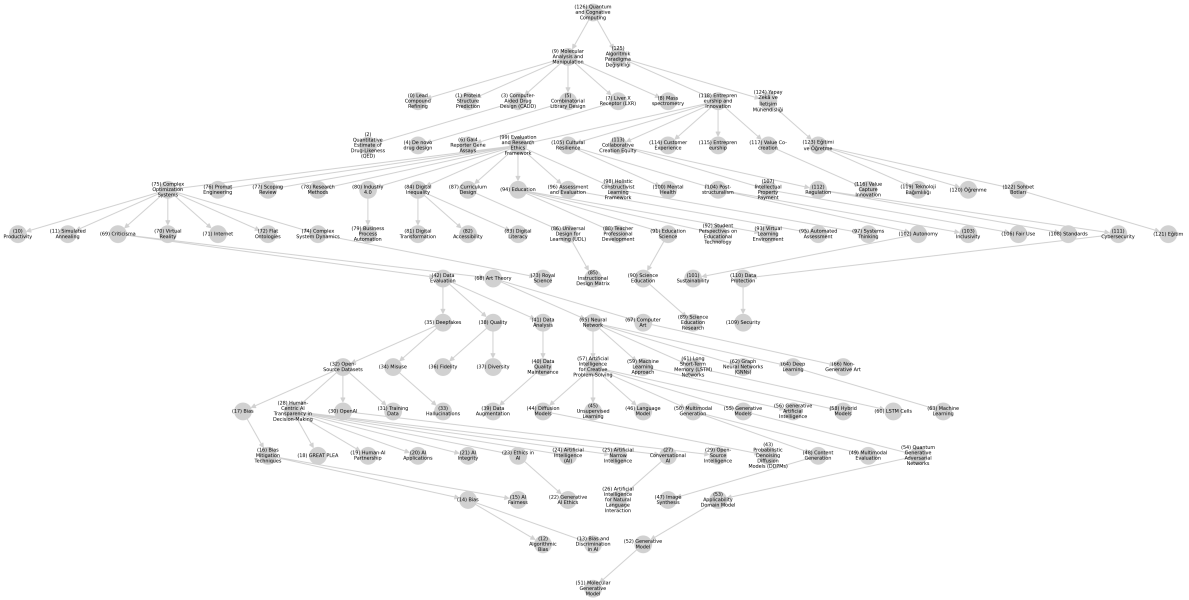

Figure 3: Full generated thesaurus. Node number is the keyword index.

In addition to the final generated trees presented in Section 4, the Figures 4 and 5 presents two more resulting trees.

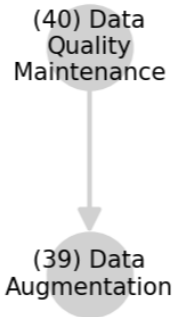

Figure 4: Thesaurus sub-tree starting at node 40.

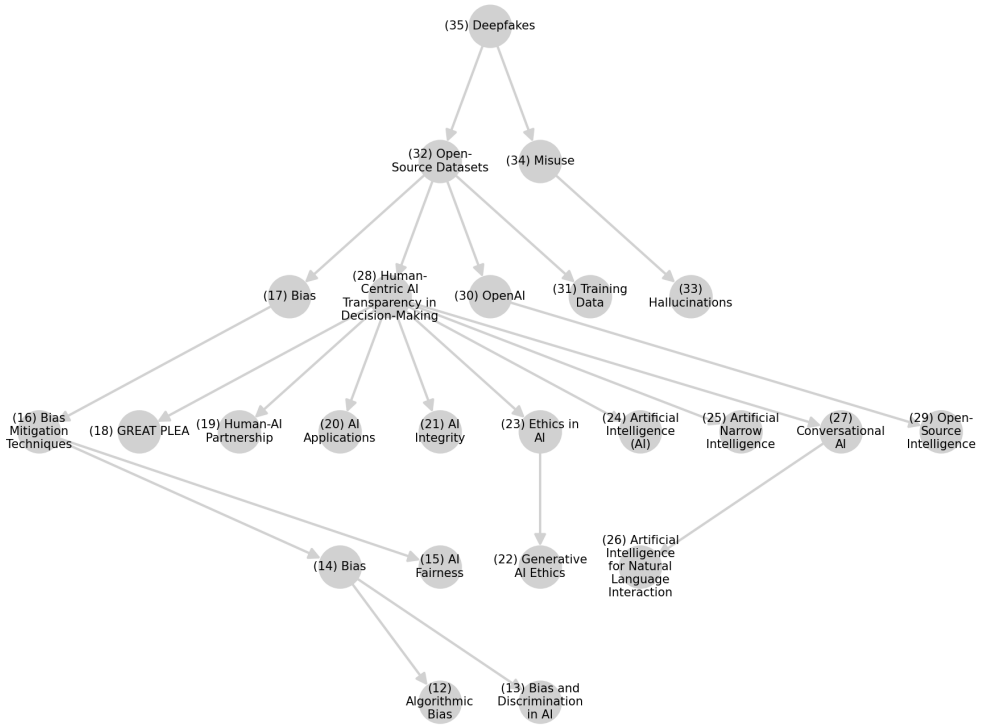

Figure 5: Thesaurus sub-tree starting at node 35.

E. Membership Score and Cosine Similarity

In addiction to ours score (Membership Score), we calculated the cosine similarity between the embedding of each keyword and the embedding of each publication (whole extracted text). The Figure 6 shows the distribution of each score, where we can observe by the curve format (approximately U-shaped curve) that the Membership Score provides a more useful separation between what would be a belongs or doesn't belong. We can note see that the Membership Scores are concentrated on the values 0, 2, 5, 7 and 8; they are the example values in the prompt (Appendix B.3).

In Figure 7, we can see again the highest contrast in the Membership Score. It's interesting to observe the low value blocks at the first keywords, they correspond to the keywords related to pharmacy (ex. *"**(2) Quantitative Estimate of Drug-Likeness (QED)**"*): there's some publications about it, but it's not central and related to all publications about generative AI. But, the existence of this block in the two scores shows a relation between them.

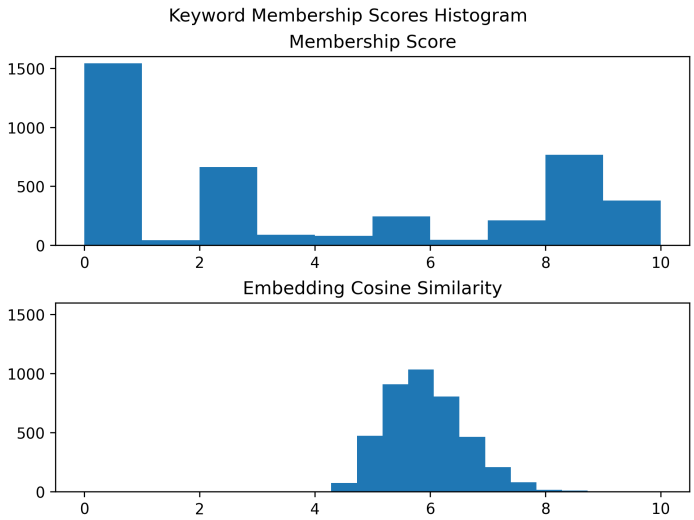

Figure 6: Keyword membership histogram of ours Membership Score (up) and Embedding Cosine Similarity (down) for each publication-keyword pair.

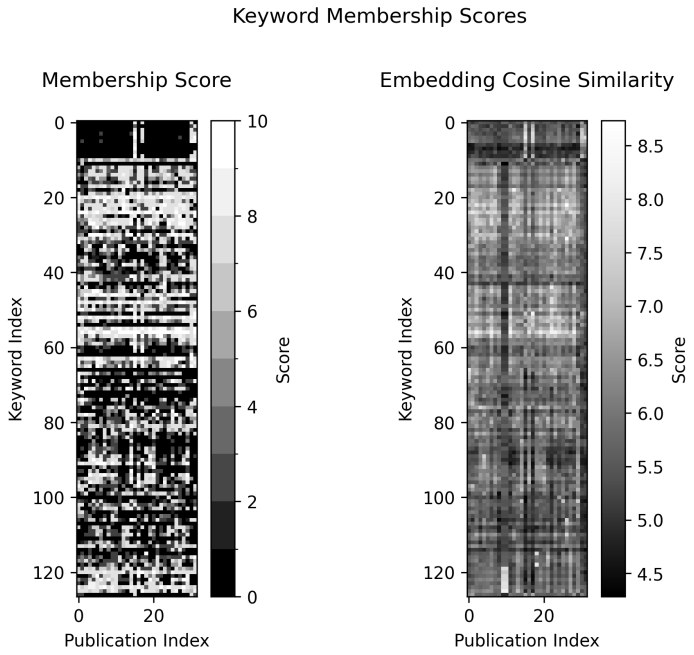

Figure 7: Keyword membership scores of ours Membership Score (left) and Embedding Cosine Similarity (right).