Comparing Power Models for circuits design with Mathematical Language Processing

Adam-Gabriel Desormiere, Lilia Gzara, Jean Bigeon, Luc Nguyen-The

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry Sciences and Computer Science Innovation

Comparing Power Models for circuits design with Mathematical Language Processing

Adam Desormiere [a,b,*], Lilia Gzara [a], Jean Bigeon [c], Luc Nguyen-thê [b]

[a] *UDL, INSA Lyon, UCBL, Université Lumière Lyon 2, DISP, EA4570, Villeurbanne, FRANCE*
[b] *Intel France, 166 Rue du Rocher de Lorzier, 38430 Moirans, FRANCE*
[c] *Nantes LS2N UMR CNRS 6004, 2 Chemin de la Houssinière, FRANCE*

Abstract

The work presented in this paper is part of a project that focuses on sorting simulation models used at Intel to calculate power consumption of electronic devices. Hundreds of thousands of analytical power models have already been accumulated in a directory of files and folders, for the simulation of the consumption of thousands of products. As a first step, this work focuses on comparing and grouping together models of smaller granularity - the formulas - which consist of power equations, the definition of the parameters involved in these equations, and a description.

To manage this large data problem, we follow the principles of Model Management and Analytics, to decompose the problem into the analysis of constituent fields. We believe that the equations describing power consumption are the most highly information-rich field of our models. The aim of this work is to group together the formulas models, using Mathematical Language Processing (MLP), to take advantage of the business information contained in electrical equations. We calculate the embeddings of our formulas, to enable comparisons between them, to cluster together similar equations (and therefore models).

Keywords: Power Modeling, Models Management, Machine Learning, Clustering

1. Introduction

We work with analytical power models that are built hierarchically from simple electrical equation models to complex models in order to simulate the power consumption of products, such as CPUs. Hundreds of thousands of

Corresponding author. Tel.: +33-6-69-78-59-82
E-mail address: adam.desormiere@intel.com.

power models have already been accumulated in our databases, for thousands of simulations, and we want to take advantage of this knowledge.

We have two goals: to sort the existing models in our databases, and to help future users in their reuse of existing models [1]. We focus on models of smaller granularity, the formulas models, which can be of three types (static, dynamic, supply) dedicated to different simulations. This allows us to pre-treat the models by separating them into three subsets. Each resulting formula model contains equations for calculating electrical power, the definition of parameters involved in these equations and a description field.

Model Management and Analysis (MMA) [2]–[4] proposes to tackle this big data problem by analyzing all the fields available in the models and processing them with machine learning [5], [6]. Among these fields, we believe that the richest information is the equation that expresses the power. So, we use Mathematical Language Processing (MLP) [7]–[10], which is based on architectures close to Natural Language Processing (NLP) [11]–[14] to encode the semantics of the equation contained in each power model. Once this information is contained in a vector, we can compare the equations with each other by comparing the vectors [15] (Figure 1), to obtain similarity scores.

This paper is organized as follows. Section 2 presents the tools and protocol used for mathematical data extraction and encoding, as well as model clustering based on similarity scores. Section 3 evaluates performance of the clustering algorithm applied to the similarity scores obtained between equations. Section 4 enumerates the conclusions from our work and introduces future perspectives.

2. Equations Embedding and Clustering

a. Need for Mathematical Language Processing (MLP)

As time goes by, advances in natural language processing continue to improve information retrieval capabilities. Today's machine translators are no longer just word-by-word processors, but are capable of capturing context. Search engines no longer simply offer a list of results, but also the ability to understand users' questions and propose answers. The progress of OpenAI with successive versions of GPT is revolutionizing the scientific community [16], [17]. However, it is precisely scientific content that is the most complex to reference. NLP algorithms are designed to process words in natural language, but do not allow encoding the mathematical portions of scientific content. Whether in scientific papers, training courses, tutorials, or encyclopedias, scientific content often includes mathematical equations. As a result, an immense amount of scientific content contains equations that current algorithms do not utilize for analysis and classification [18]. Also, the use of mathematical notation is dialectical. Different communities use different conventions for naming variables and defining operators (for example, using a horizontal line above an "x" to represent Boolean negation, versus the average of a set of values). Each author redefines and adapts the notation to his or her immediate needs. This flexibility benefits authors and readers alike, but makes automatic interpretation extremely difficult. Mathematical Language Processing is concerned with capturing mathematical language, and including it in the content to be encoded, in order to convey the meaning of a document. We want to use an algorithm based on this approach to compare our models. We only consider, for each of our models, an equation describing the amount of power consumed, as input material for an MLP algorithm.

b. Algorithm choice

The key to MLP is to choose the right way to analyze and decompose the formula, to capture its semantics. Two types of representations can be found in the literature: Operator Trees (OPTs) and Symbol Layout Trees (SLTs). OPTs contain the mathematical meaning, and SLTs the layout of the formula (the positions of the symbols with respect to each other). OPTs capture formula semantics while SLTs capture visual structure, as illustrated in figure 1.

In order to evaluate algorithms, NTCIR-12 MathIR [19], [20] is an open task for retrieving mathematical information in documents from queries. The candidate algorithms enable users to search for a particular math concept using math formulae. Queries can be keywords, formulae, or a combination. Candidate algorithms need to return a ranked list of retrieval documents (document excerpts) matching the input query. This task enables Mathematical Language Processing algorithms to be evaluated with a single, clear evaluation procedure. Algorithms proposed in literature can be compared with the formula retrieval task, on the NTCIR-12 dataset (cf. Table 1).

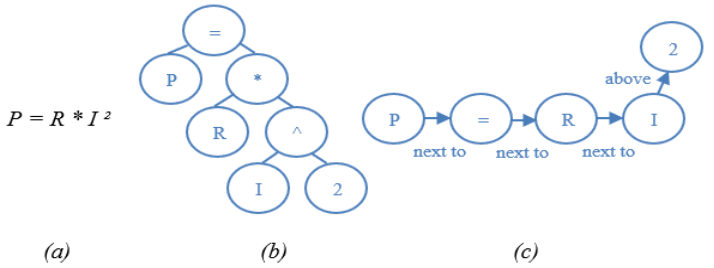

Fig. 1: Formula (a) with its OPT (b) and SLT (c) representations

Algorithm	Bpref
TangentCFT	65
TangentCFT (OPT only)	63
TangentCFT (SLT only)	61
TangentS	61
MCAT	57
Approach0	63

Table 1: Bpref [21] scores comparison between literature algorithms

When reviewing the different approaches made by the scientific community for the NTCIR task, we observe that algorithms using combined representations (OPT + SLT) tend to have better results [22]–[24]. Mathematical formulas are often unique: It is rare for the same equation to be found identically in another theorem or another field of science. Hence, FastText embedding model will learn more abstract formula representations when compared with classic tree-based approaches, especially thanks to the n-gram embedding which allows to capture local semantic of the formula.

We've chosen Tangent-CFT [22] for its high performance on this task, which demonstrates that the features included in the embeddings produced contain significant information about the formulas.

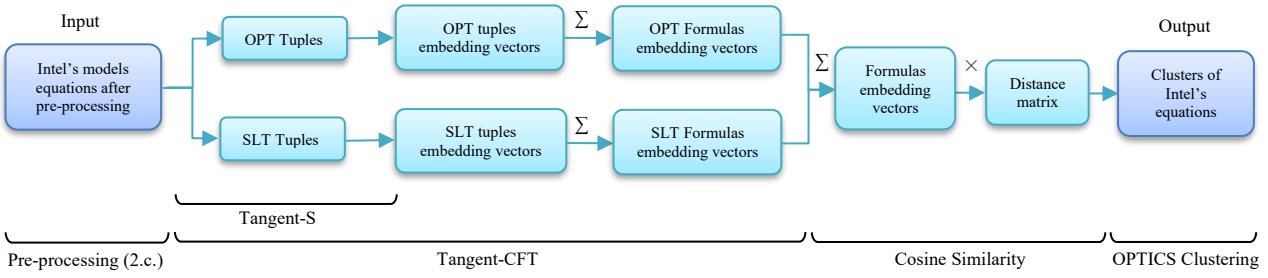

Fig.2: Overview of the workflow applied to Intel's simulation models in this article.

Figure 2 summarizes our complete workflow. We use Tangent-CFT to produce the embeddings of our formulas. Then, unlike the original algorithm which operates as a search engine, we compare all pairs of formula embeddings with a cosine similarity, to obtain a distance matrix ready to be used by a clustering algorithm.

c. Data pre-processing

We extract the equations describing the power for each model. We first sort the formulas into their 3 different types: static, dynamic and supply. These 3 types of equations cover different use cases and will be treated separately. These equations are written in a computational language from Intel, used to compute the power value. We convert these equations into LaTeX, then into MathML (content + presentation), which is suitable for Tangent-CFT. Figure 3 illustrates this process with an example. The algorithm uses these two MathML forms respectively to obtain the OPT and SLT expressions of the formula. We also perform a pre-processing step by eliminating strictly identical models, to save computation time by eliminating all duplicates.

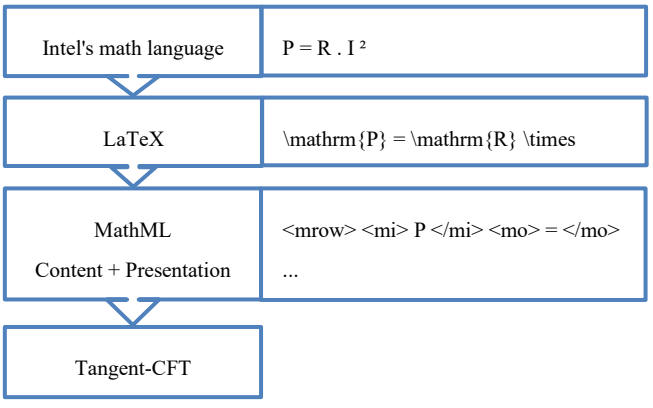

Fig. 3: Pre-processing

d. Inference

For each equation, we compute a combined OPT and SLT embedding vector. We use the cosine similarity between each pair of embedding vectors to obtain a similarity score, then compute the cosine distance from the cosine similarity. Given two embedding vectors **A** and **B** of dimension n:

$$\text{Cosine Similarity} = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{||A||\ ||B||} = \frac{\sum_{i=0}^{n} \boldsymbol{Ai}\ \boldsymbol{B}\text{i}}{\sqrt{\sum_{i=0}^{n} \boldsymbol{Ai}^2} \cdot \sqrt{\sum_{i=0}^{n} \boldsymbol{Bi}^2}}$$

Cosine Distance = 1 – Cosine Similarity

Finally, we store all those cosine distance scores in a matrix, for all pairs of equations of the database. This format is ready to be used as input by the clustering algorithm.

e. Clustering from scores

We wanted to avoid classical methods such as K-means, which require knowing the number of clusters in advance, as we don't have this information. We tried to cluster with DBSCAN, but the optimal value for the neighborhood was not always the same, depending on formula type and length.

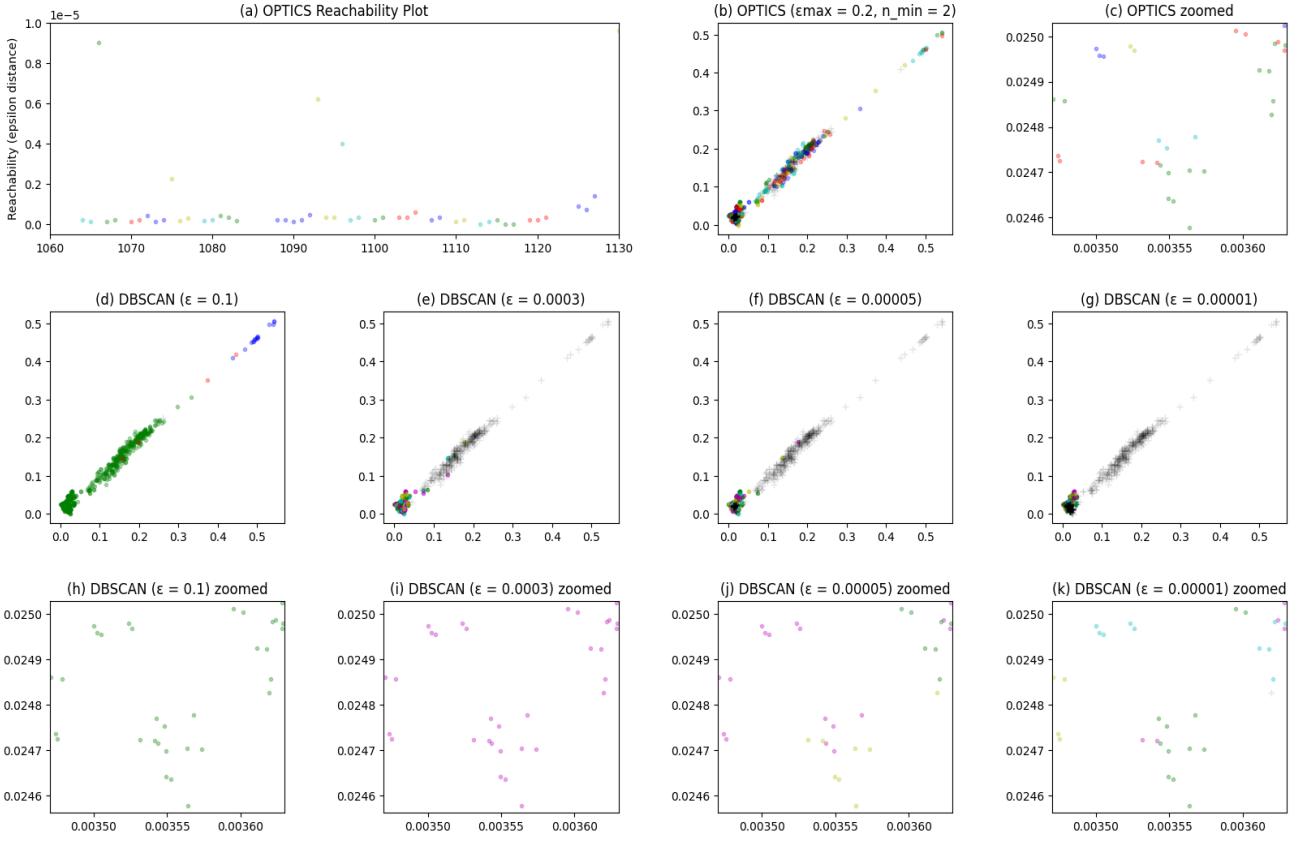

Fig. 4: Comparison between DBSCAN (with different neighborhood settings) and OPTICS.

(a) OPTICS reachability plot.
(b) OPTICS clustering proposal.
(c) Zoom on (b) on a random area.
(d), (e), (f), (g), DBSCAN clustering proposal with different neighborhood settings.
(h), (i), (j), (k), same as (d), (e), (f), (g) but zoomed on the same area as (c).

Where DBSCAN (Figure 4) fails to capture clusters in areas of varying density, OPTICS succeeds:
- On figure (d), DBSCAN is set with a very high epsilon value, and clusters all the dataset in the same cluster. Figure (h), zoomed in on a high-density area, shows that no discrimination is performed.
- As we reduce the epsilon value, we see in figures (i), (j) and (k) that clusters in high-density areas are better and better captured. However, the number of outliers increases disproportionately (>60% of the dataset for (g)) and we can see from figures (e), (f) and (g) that low-density areas are no longer considered by DBSCAN.
- Fortunately, (b) and (c) show that OPTICS manages to capture clusters in areas of varying density: rather than a neighborhood value, it takes into account abrupt variations of reachability in successive points.

As a consequence, we choose to cluster with OPTICS, which allows variable neighborhood densities [25]. OPTICS [26] is a density-based clustering algorithm similar to DBSCAN, but that can extract clusters of varying densities and shapes. It is useful for identifying clusters of different densities in large, high-dimensional datasets.

Taking a small set of four equations as an example, Figures 5 & 6 respectively show the scores calculated between pairs of equations, and the Multidimensional Scaling representation of the samples. Indexes start at 0.

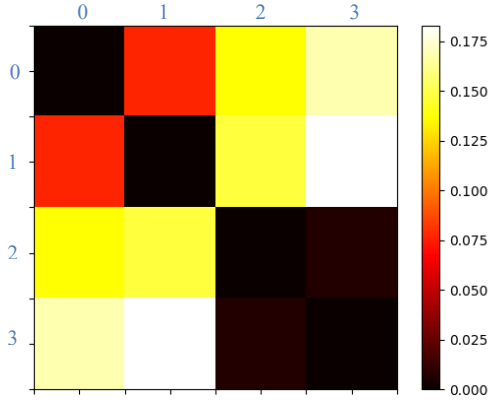

Fig. 5: Distance matrix between pairs of samples

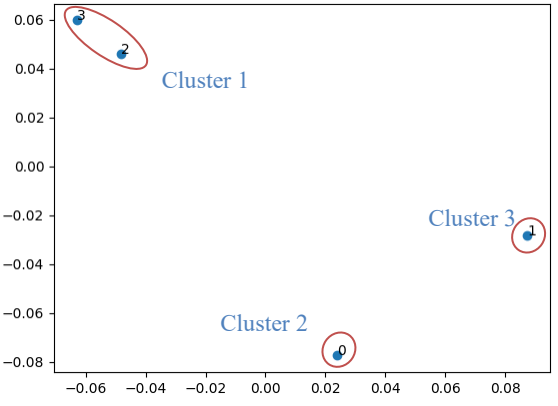

Fig. 6: Multidimensional Scaling to visualize distances between four samples

Hence, we compared mathematical formulas with each other, and used the resulting similarity scores to cluster the corresponding models. In the following, we evaluate the quality of our algorithm's proposals.

3. Validation and results

a. Validation sets

Unfortunately, once a clustering proposal has been made by our algorithm, we can't evaluate it. We have no ground truth for the models in our database, so, even if we verified that the clustering proposals made by our algorithm are satisfactory and bring together similar patterns, we can't quantify this without a validation set. To overcome this problem, we asked three Intel engineers to cluster by hand all the models belonging to three different teams. We chose three teams working on different and diversified products to get a representative sample of all power engineers. There are three diverse types of equations, so each engineer will have to manually cluster nine subsets of equations, giving us 27 manual proposals on representative samples from our database. Each proposal made by the engineers is valid, their manual clustering is subjective and based on their business knowledge, so the proposals can vary.

On the other hand, we can evaluate (cf. Table 2) the similarity of different proposals made by each engineer on each cluster using the Rand Index [27], which represents the frequency of occurrence of agreements over the total pairs. Given two partitions A and B of a set Ω, we call:

- u the number of pairs of elements of Ω that are in the same subset in A and in the same subset in B

- u' the number of pairs of elements of Ω that are in different subsets in A and in different subsets in B
- v the number of pairs of elements of Ω that are in the same subset in A and in different subsets in B
- v' the number of pairs of elements of Ω that are in different subsets in A and in the same subset in B

Then the Rand Index is defined by $RI = \frac{u + u'}{u + u' + v + v'}$

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6	Subset 7	Subset 8	Subset 9
Engineer 1 / Engineer 2	0.90	1.0	0	0.23	1.0	0.37	1.0	0.36	0.41
Engineer 1 / Engineer 3	0.37	1.0	0.57	0.18	1.0	0.41	0.18	0.36	0.41
Engineer 2 / Engineer 3	0.42	1.0	0	0.88	1.0	0.55	0.18	1.0	1.0

Table 2: Comparison (Rand Index) of clustering proposals from three engineers for two subsets of models.

b. Algorithm evaluation

It's important to remember that clustering proposals are subjective, and result from the engineer's interpretation. There is no one proposition that can prevail over another, and we keep all these propositions as valid ground truths for evaluating our algorithms. Nevertheless, we note that these propositions are often extremely close. Thanks to these validation sets, we can now evaluate our algorithm's clustering proposals. We use the 9 validation subsets described above, and the 3 ground truths from the engineers who manually labelled the models.

We use the classic evaluation measures [28], we recall that a clustering result satisfies:

- Homogeneity - if all of its clusters contain only data points that are members of a single class.
- Completeness - if all the data points that are members of a given class are elements of the same cluster.

Both scores (cf. Table 3) have positive values between 0.0 and 1.0; larger values being desirable. V-measure [29] is the harmonic mean of the first two.

	Adjusted Rand Index			Homogeneity			Completeness			V-measure		
Ground truth n° →	1	2	3	1	2	3	1	2	3	1	2	3
Subset 1 (team A)	0.37	0.48	0.11	0.84	1.0	0.70	0.52	0.61	0.23	0.65	0.76	0.35
Subset 2 (team A)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Subset 3 (team A)	1.0	0	0.57	1.0	0.75	1.0	1.0	1.0	0.67	1.0	0.86	0.80
Subset 4 (team B)	0.27	0.51	0.43	0.69	0.80	0.78	0.69	0.78	0.67	0.69	0.79	0.72
Subset 5 (team B)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Subset 6 (team B)	0.42	0.03	0.06	0.74	0.56	0.59	0.74	0.61	0.46	0.74	0.59	0.52
Subset 7 (team C)	0.36	0.36	0.29	0.66	0.66	0.67	0.88	0.88	0.61	0.75	0.75	0.64
Subset 8 (team C)	0.50	0.40	0.40	0.72	0.76	0.76	1.0	0.75	0.75	0.84	0.76	0.76
Subset 9 (team C)	0.16	0.71	0.71	0.37	0.69	0.69	0.46	1.0	1.0	0.41	0.81	0.81

Table 3: Evaluation of our algorithm with classic measures, when compared to three different ground truths

In most cases, the algorithm proposes the same thing as the engineers. In the few cases where it proposes a different clustering, the proposal is logical and makes sense, and we understand its purpose when examining it. As a result, we are satisfied with its ability to cluster similar power models.

4. Conclusion & Future research

In this paper, we have shown that we can take advantage of the semantic information contained in mathematical equations to compare models. Using MLP, we have encoded our formulas in a space that expresses their mathematical characteristics, and we can then compare the formulas with each other thanks to the cosine similarity between the vectors. We can use these similarity scores to sort our databases, by grouping the formulas into subsets using a clustering algorithm, whose proposals are quite close to those that an expert would have obtained manually. We can also use these comparison scores to suggest to the user, like a search algorithm (Tangent-CFT's initial use case), equations like the one they are currently typing. In this way, the tool is an excellent assistant to an application administrator for tidying and sorting equations.

Unfortunately, this clustering does not allow us to automatically label subsets once the models have been grouped. A manual interpretation from an expert is required to finish exploiting these results and arranging the models.

Our prospects for future work target the creation of a search engine, which redirects the user of our application to similar models (based on the MLP as described in this paper), in real time as he types in the mathematical formula for the power consumption of his model.

We're also considering changing the embedding algorithm, as some recent algorithms offer different embedding methods, potentially more efficient than FastText, which is an area worth exploring.

Acknowledgments

The authors would like to thank the *Association Nationale Recherche et Technologie* (ANRT) for its financial contribution. We also thank the members of the Intel® Docea™ team.

References

[1] A. Desormiere, L. Gzara, J. Bigeon, and L. Nguyen-Thê, “Hierarchical Clustering of Power Models for circuits design,” *Procedia Comput Sci*, vol. 204, pp. 566–572, 2022, doi: 10.1016/j.procs.2022.08.069.

[2] B. Tekinerdogan, Ö. Babur, L. Cleophas, M. van den Brand, and M. Akşit, “Introduction to model management and analytics,” in *Model Management and Analytics for Large Scale Systems*, Elsevier, 2020, pp. 3–11. doi: 10.1016/b978-0-12-816649-9.00009-0.

[3] Ö. Babur, “Model Analytics and Management.”

[4] Ö. Babur, L. Cleophas, M. van den Brand, B. Tekinerdogan, and M. Aksit, “Models, More Models, and Then a Lot More,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2018, pp. 129–135. doi: 10.1007/978-3-319-74730-9_10.

[5] J. Heaton, “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning,” *Genet Program Evolvable Mach*, vol. 19, no. 1–2, pp. 305–307, 2018, doi: 10.1007/s10710-017-9314-z.

[6] S. Haque, Z. Eberhart, A. Bansal, and C. McMillan, “Semantic Similarity Metrics for Evaluating Source Code Summarization,” in *IEEE International Conference on Program Comprehension*, IEEE Computer Society, 2022, pp. 36–47. doi: 10.1145/nnnnnnn.nnnnnnn.

[7] K. Cobbe *et al.*, “Training Verifiers to Solve Math Word Problems,” pp. 1–22, 2021.

[8] G. Lample and F. Charton, “Deep Learning for Symbolic Mathematics,” pp. 1–24, 2019, [Online]. Available: http://arxiv.org/abs/1912.01412

[9] Z. Wang, A. Lan, and R. Baraniuk, “Mathematical formula representation via tree embeddings,” *CEUR Workshop Proc*, vol. 2895, pp. 121–133, 2021.

[10] J. Meadows and A. Freitas, “A Survey in Mathematical Language Processing,” 2022, [Online]. Available: http://arxiv.org/abs/2205.15231

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12, 2013.

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” Jul. 2016, [Online]. Available: http://arxiv.org/abs/1607.04606

[13] A. Vaswani *et al.*, “Attention is all you need,” *Adv Neural Inf Process Syst*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[14] P. Dadure, P. Pakray, and S. Bandyopadhyay, “BERT-based embedding model for formula retrieval,” *CEUR Workshop Proc*, vol. 2936, pp. 36–46, 2021.

[15] K. Krstovski and D. M. Blei, “Equation Embeddings,” 2018, [Online]. Available: http://arxiv.org/abs/1803.09123

[16] OpenAI, “GPT-4 Technical Report,” vol. 4, pp. 1–100, 2023.

[17] T. B. Brown *et al.*, “Language models are few-shot learners,” *Adv Neural Inf Process Syst*, vol. 2020-Decem, 2020.

[18] D. Hendrycks *et al.*, “Measuring Mathematical Problem Solving With the MATH Dataset,” no. NeurIPS, pp. 1–22, 2021.

[19] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila, “NTCIR-12 MathIR Task Overview,” *Proc. 12th NTCIR conference*, pp. 299–308, 2016.

[20] H. Joho, “Overview of NTCIR-11,” pp. 1–7, 2014.

[21] T. J. Green, “Bag Semantics,” *Encyclopedia of Database Systems*, pp. 201–206, 2009, doi: 10.1007/978-0-387-39940-9_979.

[22] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi, “Tangent-CFT: An embedding model for mathematical formulas,” *ICTIR 2019 - Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 11–18, 2019, doi: 10.1145/3341981.3344235.

[23] S. Peng, K. Yuan, L. Gao, and Z. Tang, “MathBERT: A Pre-Trained Model for Mathematical Formula Understanding”.

[24] K. Davila and R. Zanibbi, “Layout and semantics: Combining representations for mathematical formula search,” in *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, Inc, Aug. 2017, pp. 1165–1168. doi: 10.1145/3077136.3080748.

[25] P. Dubey and A. Rajavat, “- Dbscan and Optics,” no. 12, pp. 34–37, 2016.

[26] M. Ankerst, M. M. Breunig, and H. Kriegel, “OPTICS : Ordering Points To Identify the Clustering Structure,” pp. 49–60, 1999, doi: 10.1145/304181.304187.

[27] S. Zhang, H. S. Wong, and Y. Shen, “Generalized adjusted rand indices for cluster ensembles,” *Pattern Recognit*, vol. 45, no. 6, pp. 2214–2226, 2012, doi: 10.1016/j.patcog.2011.11.017.

[28] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, “A comparison of extrinsic clustering evaluation metrics based on formal constraints,” *Inf Retr Boston*, vol. 12, no. 4, pp. 461–486, 2009, doi: 10.1007/s10791-008-9066-8.

[29] A. Rosenberg and J. Hirschberg, “V-Measure: A conditional entropy-based external cluster evaluation measure,” *EMNLP-CoNLL 2007 - Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, no. June, pp. 410–420, 2007.