Improving Histogram-Based Mathematical Formula Search with Local Matching and Learned Embeddings

Quinn Tucker
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
qt2393@rit.edu

***Abstract*—The Pyramidal Histogram of Characters (PHOC) is a spatial vector representation that has recently been applied to the task of mathematical formula search. It is attractive for being computationally lightweight, but unfortunately struggles more than other formula ranking methods in cases involving (a) a small query match embedded within a larger formula or (b) a mismatch in variable names or notation between otherwise equivalent formulas.**

In this work we aim to address these two limitations via two independent modifications: matching local windows and replacing sparse binary vectors with learned dense embeddings. We evaluate these on the ARQMath-3 formula search benchmark and find that local matching does indeed improve retrieval effectiveness, but our integration of dense embeddings did not.

***Index Terms*—Formula retrieval; Spatial embeddings**

I. Introduction

A. Mathematical formula search

The ability to search collections of information with mathematical notation has diverse applications spanning education and research, and the construction of software systems to address this need has been a topic of investigation for multiple decades [2]. One specific task falling under this umbrella is *formula search*, in which a user issues a query consisting of mathematical notation and the system must retrieve relevant formulas from some collection.

The Answer Retrieval for Questions on Math lab, most recently part of CLEF 2022 (ARQMath-3 [3]), includes a formula search task with an associated formula collection (derived from Math Stack Exchange) and datasets of relevance judgements assessed by human annotators. In this work we extend an existing system submitted to ARQMath ([1]) with the aim of improving its effectiveness on this mathematical information retrieval benchmark.

B. Pyramidal Histogram of Characters (PHOC)

The Pyramidal Histogram of Characters (PHOC) is a vector representation of sequentially or spatially distributed letters or symbols that was originally proposed for word spotting in images [4]. Avenoso et al. [1] describe an extension to two dimensions, which they dub XY-PHOC, and apply it to the formula search task.

First let us precisely define some terminology. A **symbol** refers to a particular instance of a glyph within a formula. Every symbol has a bounding box and a **label**, which specifies what "kind" of symbol it is (e.g. x or $+$). The formula's overall bounding box is divided into various spatial **regions**; in XY-PHOC these are vertical and horizontal subdivisions with varying resolution, and other work has explored different region shapes (we refer the reader to [5] for more details and an explanation of the $\texttt{xy}n$ notation for region configurations).

For a given configuration of regions, a **PHOC vector** (or simply a **PHOC**) refers to a binary vector indexed by (symbol label, PHOC region) pairs, with a dimensionality of

$$(\text{\# of symbol labels}) \times (\text{\# of PHOC regions}). \tag{1}$$

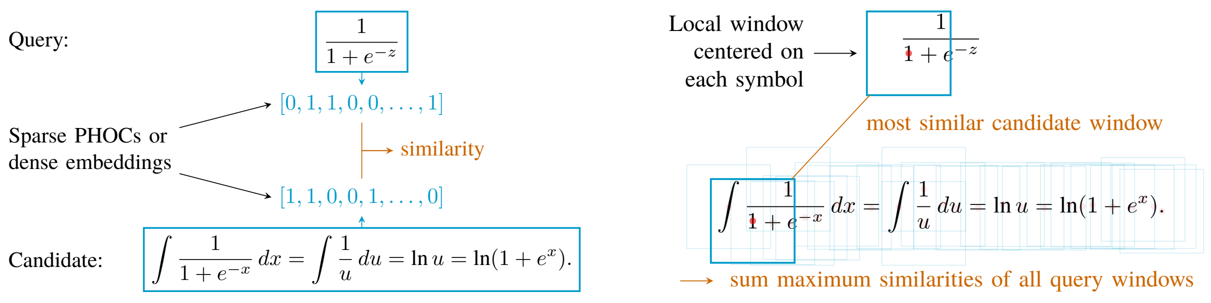
If ℓ is a symbol label and r is a PHOC region within the window, the component for the pair (ℓ, r) is defined as

$$v_{\ell,r} = \begin{cases} 1 & \begin{array}{l}\text{if there is a symbol with label } \ell \\ \text{whose bounding box intersects } r\end{array} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In practice, the set of labels ℓ with non-zero components in $\mathbf{v}$ is sparse, so $\mathbf{v}$ can be stored as a short list of labels, each accompanied with a fixed-length bit string representing which regions contain a symbol with that label. For example, if there are 6 PHOC regions, a PHOC might be stored like this:

$$\begin{aligned} \mathbf{v} \equiv [\ &(x,\ \overbrace{100101}^{\text{6 regions = 6 bits}}) \\ &(2,\ 001000) \\ &(+,\ 110100)\ ] \end{aligned} \tag{3}$$

For the formula search task, candidate formulas are ranked by the cosine similarity between the candidate PHOC vector and the query PHOC vector (see Figure 1a), and the top k results are returned to the user [1].


Query:
$\frac{1}{1+e^{-z}}$
Sparse PHOCs or dense embeddings
$[0,1,1,0,0,\ldots,1]$
similarity
$[1,1,0,0,1,\ldots,0]$
Candidate:
$\int \frac{1}{1+e^{-x}}\,dx = \int \frac{1}{u}\,du = \ln u = \ln(1+e^x).$
Local window centered on each symbol
most similar candidate window
sum maximum similarities of all query windows

(a) Global matching. (b) Greedy local matching.

Fig. 1. Illustration of (a) global matching, which has been used in prior work on PHOCs [1], versus (b) our proposed greedy local matching.

C. This work

While prior work has demonstrated that formula-level (global) PHOC vectors are a useful and lightweight representation for measuring formula similarity [1] [5] [6], they struggle in certain situations.

One pain point is a small query match embedded within a larger candidate formula. Since the query and candidate differ in their global structure, their formula-level PHOC representations are likely to be dissimilar. To address this, we propose to compute PHOC vectors for *local* windows in both the query and candidate. Then, drawing inspiration from the ColBERT [7] passage ranking model, we can align these local representations to produce a final score for ranking.

Previous systems also underperform when a query and candidate use different notation or variable names to express similar mathematical concepts – effectively a *vocabulary mismatch* in the domain of symbol labels. For this we propose to learn dense embeddings for PHOC region representations, with the aim to capture gradations of cosine similarity between non-identical symbol labels.

Our two research questions are therefore:

RQ1. Does the use of local (as opposed to global) PHOC matching improve retrieval effectiveness?

RQ2. Does learning dense embeddings for PHOC region representations improve retrieval effectiveness?

II. Methodology

We now detail our two modifications to PHOC-based formula search: local matching (Section II-A) and learned dense embeddings (Section II-B). We then describe the experimental setup we use to assess their effectiveness (Section II-C).

A. Greedy local matching

In the *global matching* paradigm from prior PHOC systems (such as [1] and [5]), each formula (query or candidate) is represented by a single PHOC vector computed from the contents of the formula's entire bounding box [1] (Figure 1a).

We instead propose to represent a formula as a *set* of PHOC vectors computed from *local windows*. As a simple way to distribute windows across the formula, we compute one PHOC vector for each symbol using the contents of a square window centered on that symbol. (For example, the formula $x + x^2$ would have four local PHOCs corresponding to the four symbols.) The computation of each local PHOC differs only in the translation of the window; in particular, the central symbol receives no special treatment. The width of the window is a hyperparameter that we set relative to the x-height of the surrounding font.

Given their sets of local PHOC vectors, we use a method analogous to the *MaxSim* token alignment mechanism from ColBERT [7] to compute the similarity between a query and a candidate formula (Figure 1b). First, we match each local window in the query to the most similar local window in the candidate. Then we sum these "maximum similarity" values to get the overall query-candidate similarity score

$$\text{score}(Q,C) = \sum_{\mathbf{v}_i^{(q)} \in Q} \max_{\mathbf{v}_j^{(c)} \in C} \cos(\mathbf{v}_i^{(q)}, \mathbf{v}_j^{(c)}) \tag{4}$$

where Q and C are the sets of local PHOC vectors from the query and candidate (respectively) and $\cos(\cdot,\cdot)$ denotes the cosine similarity between two vectors.

Candidates are then ranked in descending order by score.

B. Learning dense embeddings

The PHOC representation for the contents of a (global or local) window is effectively a concatenation of bag-of-symbol-labels representations from each of the individual regions within the window.

Consider some window in some formula, such as any of the blue windows in Figure 1. The representation vector $\mathbf{v}_r$ for each region r in this window can be described as a sum of "embedding" vectors $\mathbf{e}_\ell$ for each symbol label ℓ present in that region:

$$\mathbf{v}_r = \sum_{\ell \in r} \mathbf{e}_\ell \tag{5}$$

In the original sparse PHOC representation, each $\mathbf{e}_\ell$ is simply a $|V|$-dimensional 1-hot vector encoding the identity of the symbol ℓ (where V is the vocabulary of symbol labels). Any two non-identical symbol labels are therefore completely orthogonal, even if they are semantically equivalent in context (such as R versus $\mathbb{R}$ to denote the set of real numbers).

Replacing these high-dimensional sparse vectors with lower-dimensional dense vectors could in theory capture a more nuanced measure of similarity between non-identical symbols. To

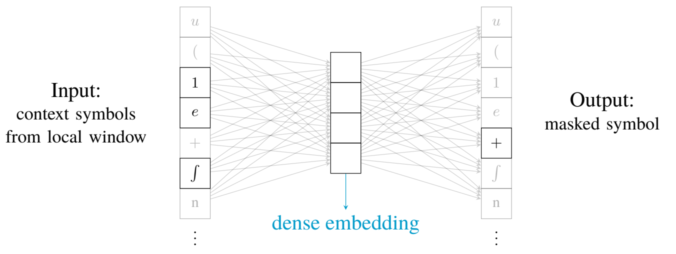


Fig. 2. The model architecture for learning dense bag-of-symbols embeddings.

test this hypothesis we use an unsupervised masked-prediction task similar to word2vec [8] (and in the same vein as more sophisticated models like BERT [9]) to learn dense embedding vectors.

Figure 2 illustrates the prediction task and model structure. First we select a random local window from a random formula in the collection and get the set of symbol labels it contains. A random label in the window is "masked out", and the remaining labels are passed as input to the embedding network. The network must then compress this information into a low-dimensional embedding in the middle which is finally used to predict the identity of the masked label. Intuitively, this encourages sets of symbols that appear in similar contexts to be given similar dense embeddings.

We use a simple model architecture with binary inputs, two linear layers without biases, and a cross-entropy loss. We initialize the first layer to random unit vectors and the second layer to zeros, then train using AdamW with $\gamma = 0.001$ and $\lambda = 0.0001$ for 20,000 iterations with a batch size of 1,000.

Once the network has been trained, we use the first layer of the network to compute dense embeddings for PHOC regions. This is a matrix multiplication between the first layer's weight matrix and the binary representation of the region's symbol labels, which is equivalent to summing the columns of the weight matrix that correspond to the labels present in the region. Therefore, we can interpret this as replacing the sparse 1-hot symbol embeddings in Eq. 5 ($\mathbf{e}_\ell$) with dense symbol embeddings extracted from the trained network parameters.

It is worth noting that our setup has a couple differences between training and inference. The first is that labels are only masked from the input during training; at query time, the input to the network is the entire set of labels for a region. The second is that during training, the input contexts are entire local windows, while at query time the input "contexts" are PHOC regions, which are subsets of the current local or global window.

C. Experimental design

To measure the impact of our two proposed modifications, both when combined and when applied independently, we compare the effectiveness of four candidate scoring methods:

- Global+Sparse (the baseline without our modifications)
- Global+Dense
- Local+Sparse
- Local+Dense

For each scoring method we independently tune the *PHOC region configuration* and *window size* hyperparameters using a grid search over the following ranges:

- region configuration $\in \{\texttt{x5}, \texttt{y5}, \texttt{r5}, \texttt{xy5}, \texttt{xr5}, \texttt{yr5}\}$
- window size $\in \{2\text{ex}, 3\text{ex}, 4\text{ex}, \ldots, 12\text{ex}\}$

where the "ex" unit denotes multiples of the surrounding font's x-height. All scoring methods except Global+Sparse are affected by the window size; the embeddings for Global+Dense were trained using local window contexts. For each scoring method we select the hyperparameter configuration that achieves the best nDCG$'$@1000 on a set of queries from ARQMath-1 and 2 for use in our main evaluation.

Our main experiment then evaluates the four scoring methods on 76 queries from Task 2 of ARQMath-3. We use a collection of 31,206 formulas, consisting of formulas for which there are relevance judgements from any of the ARQMath competitions. We compute nDCG$'$, mAP$'$, and P$'$@10, @5, and @1 by exhaustively scoring all formulas in the collection, ranking them by score (without any retrieval stage or cutoff), and ignoring results that do not have a relevance judgement for the current query.

In all Dense experiments we use a fixed embedding dimensionality of 32 and retrain the embeddings for every window size. We train the embeddings on the same collection of 31,206 formulas used for evaluation; this would be a realistic practice for a deployed system, since the unsupervised training procedure does not use relevance labels and only requires the formulas themselves.

III. Results and Discussion

A. Hyperparameter tuning

Table I shows the best hyperparameters identified for each of the four scoring methods as a result of the grid search on queries from ARQMath-1 and 2.

For all four scoring methods, PHOC region configurations with x splits tended to perform better (in terms of nDCG$'$ on the tuning set) than those without. xy5 in particular performed well across the board, especially for the two Global scoring methods.

For scoring methods that use local windows, window sizes greater than 7ex generally performed the best. Very small window sizes (2-3ex) harmed effectiveness.

B. Evaluation on ARQMath-3

Table II shows the results of our evaluation on ARQMath-3 for each of the four scoring methods considered.

Both systems using **local matching** performed significantly better in nDCG$'$, mAP$'$, and P$'$@10 compared to the baseline, and did not perform worse in P$'$@5 and P$'$@1 ($p < 0.05$). This validates our initial hypothesis that fine-grained, local alignment results in a better similarity metric for the formula search task.

Dense embeddings, however, did not provide much benefit in our experiments. The use of that modification on its own (Global+Dense) did not produce statistically different results from the baseline in any metric examined. When adding dense

TABLE I
Hyperparameter tuning results. For each of the four scoring methods, the parameters selected are those with the highest nDCG′ on the training set (rightmost column).

	Best parameters		Effectiveness on ARQMath-1 and 2
Scoring method	Regions	Window size	nDCG′@1000
Global + Sparse	xy5	—	0.6892
Global + Dense	xy5	9ex	0.6818
Local + Sparse	xr5	11ex	0.7505
Local + Dense	x5	9ex	**0.7660**

TABLE II
Results from our main experiment evaluating the impact of our two scoring modifications. Significant differences from Global+Sparse are indicated with an asterisk ($p < 0.05$; two-sided t-tests with Bonferroni correction).

	Parameters		Effectiveness on ARQMath-3				
Scoring method	Regions	Window size	nDCG′	mAP′	P′@10	P′@5	P′@1
Global + Sparse	xy5	—	0.8014	0.5233	0.5632	0.6211	0.8026
Global + Dense	xy5	9ex	0.8051	0.5322	0.5592	0.6474	0.7895
Local + Sparse	xr5	11ex	**0.8302***	0.5806*	**0.6605***	**0.7263***	**0.8684**
Local + Dense	x5	9ex	**0.8302***	**0.5834***	0.6487*	0.7053	0.8289

embeddings on top of local matching (Local+Dense, relative to Local+Sparse), nDCG′ and mAP′ were minimally affected and small *decreases* were observed in P′@10, P′@5, and P′@1.

We interpret this result with the following hypothesis. For some queries, the use of dense embeddings does indeed surface relevant results that sparse scoring would miss due to differences in notation or variable names. But for other queries, it also surfaces *irrelevant* results involving symbols that happen to be nearby in the embedding space. In other words, the structure of our embedding space is not entirely aligned with what is important for relevance.

Since our evaluations use exhaustive comparison on a reduced collection without a retrieval stage, these results are not directly comparable to previously reported results for ARQMath-3. Despite that, our results do provide an informal upper bound on the effectiveness of the algorithms under consideration, and the relative improvements we observe with local matching suggest that it is a promising direction for future study.

IV. Conclusion

Our results suggest that greedy alignment over local windows improves the effectiveness of PHOC representations for the task of mathematical formula search. We hypothesize that this is because local alignment prevents scoring from penalizing relevant candidates that nonetheless contain non-matching content or have a different spatial layout than the query. Local matching also instills a degree of invariance to global translation, which relates to invariance priors in other models such as convolutional neural networks. Ultimately, local matching exploits the modular and compositional nature of mathematical notation to create a better relevance heuristic.

In contrast, our use of dense embeddings did not benefit retrieval effectiveness. This is somewhat surprising given that dense vector representations have seen great success in many tasks, including representations trained with unsupervised context-prediction objectives [9]. We therefore offer a few hypothetical explanations for our experimental results:

- The discrepancy between the network's input distribution during training (entire local window contexts) versus at query time (sub-region contexts) may harm performance.
- Although we did not observe improvements when increasing the embedding dimension during informal experimentation, it is possible that our choice of 32 dimensions is too small and lacks representational capacity.
- The unsupervised context modeling objective may simply be a suboptimal proxy for the formula relevance task; fine-tuning the embeddings on relevance judgements with a suitable learning-to-rank loss may improve effectiveness.

Testing these hypotheses is left to future work.

Despite the current popularity of deep neural networks in information retrieval, the PHOC representation is attractive for its conceptual simplicity, light computational footprint, and lack of reliance on large training datasets. Our work shows that a straightforward modification — aligning PHOCs computed from local windows — can improve effectiveness. We leave it to future work to evaluate this approach on larger benchmarks, to explore alternative methods for contextualizing symbols, and to extend these ideas for efficient comparison of spatial structures in other domains.

Acknowledgment

The author would like to thank Dr. Richard Zanibbi for his guidance throughout this project; Bryan Amador for his assistance with the existing PHOC codebase and ARQMath datasets; and the Document and Pattern Recognition Lab at RIT for providing essential computational resources.

References

[1] R. Avenoso, B. Mansouri, and R. Zanibbi, "XY-PHOC symbol location embeddings for math formula retrieval and autocompletion," 2021. [Online]. Available: https://ceur-ws.org/Vol-2936/paper-02.pdf

[2] Richard Zanibbi and Dorothea Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 4, pp. 331–357, Dec. 2012. [Online]. Available: http://link.springer.com/10.1007/s10032-011-0174-4

[3] B. Mansouri, V. Novotný, A. Agarwal, D. W. Oard, and R. Zanibbi, "Overview of ARQMath-3 (2022): Third CLEF lab on answer retrieval for questions on math," in *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Cham: Springer International Publishing, 2022, vol. 13390, pp. 286–310. [Online]. Available: https://link.springer.com/10.1007/978-3-031-13643-6_20

[4] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[5] M. Langsenkamp, B. Mansouri, and R. Zanibbi, "Expanding spatial regions and incorporating IDF for PHOC-based math formula retrieval at ARQMath-3," 2022.

[6] M. Langsenkamp, "SpatialEntity2Vec: Creating dense vector representations for entities within a 2D space."

[7] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 39–48. [Online]. Available: https://doi.org/10.1145/3397271.3401075

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NeurIPS'13, 2013, pp. 3111–3119.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423