XY-PHOC Symbol Location Embeddings for Math Formula Retrieval and Autocompletion

Robin Avenoso, Behrooz Mansouri and Richard Zanibbi

Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY, USA 14623

Abstract

We describe the participation of the XY-PHOC team from The Document and Pattern Recognition Lab (DPRL) from the Rochester Institute of Technology (RIT, USA) in ARQMath 2021 Task 2 (Formula Retrieval). We generalize a one dimensional spatial encoding for word spotting in handwritten document images, the Pyramidal Histogram of Characters or PHOC, to obtain the two-dimensional XY-PHOC, which provides robust spatial embeddings of symbols with modest storage requirements. XY-PHOC symbol location embeddings capture the relative position of symbols without the need to generate or store explicit edges between symbols. For ARQMath 2021, the XY-PHOC model obtains competitive results in formula similarity search. We also present new results using XY-PHOC for the related task of formula autocompletion from visual queries, in which target formula symbols may be added to the query in an any order.

Keywords

Character Embeddings, Spatial Embeddings, Formula Retrieval, Formula Autocomplete, Mathematical Information Retrieval (MIR)

1. Introduction

The XY-PHOC team from The Document and Pattern Recognition Lab (DPRL) from the Rochester Institute of Technology (RIT, USA) participated in ARQMath 2021 Task 2. In this task, the participants are given a formula from questions in Task 1, and are asked to return the top-1000 relevant formulas for each topic [1, 2, 3]. These retrieved formulas are from the ARQMath Math Stack Exchange corpus, containing posted answers and questions from 2010-2018. Formula topic queries are then taken from posts in 2019 (ARQMath 2020) and 2020 (ARQMath 2021).

We provided one run for Task 2, using the XY-PHOC system [4]. XY-PHOC uses a very simple relative spatial encoding to capture the locations of symbols in a formula at different levels of granularity. Embeddings for the location of individual symbols are stored in an inverted index, which can then be compared with symbol location embeddings for the query formula using cosine similarity.

Current state-of-the-art systems for formula similarity search rely on graph-structured data, such as in Tangent-S [5] and Tangent-CFT [6]. These systems require special indexing strategies in order to build inverted indexes on non-traditional keys. The scoring of these systems also requires several operations (e.g., a subtree alignment step in Tangent-S) or trained embedding models for tree paths (e.g., in Tangent-CFT).

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
rxa8448@rit.edu (R. Avenoso); bm3302@rit.edu (B. Mansouri); rxzvcs@rit.edu (R. Zanibbi)

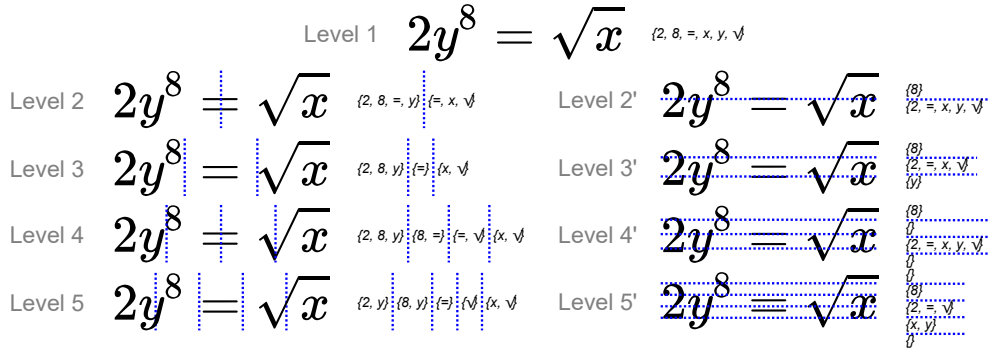
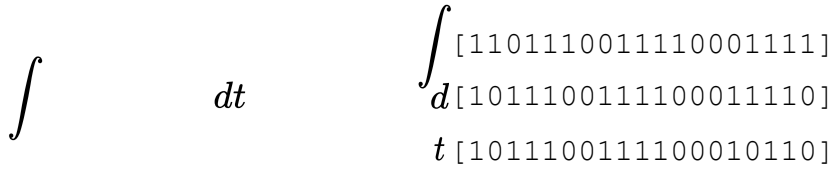


Figure 1: Example XY-PHOC Embedding. Each level numbering indicates the number of regions in the horizontal and vertical (′) directions. A symbol belongs to regions intersected by a horizontal line placed at the vertical center of its bounding box, where the horizontal line spans the width of the bounding box (vertical-center). For space, symbol region memberships are shown using sets.

The XY-PHOC representation requires less storage than does a graph-based model: in the XY-PHOC symbol location embeddings index keys are symbols, and postings consist of 29-bit binary vectors stored in 32-bit words, accompanied by an integer identifier for the associated formula (Section 2 provides details). Storing standard types in the index will allow for standard information retrieval tools and techniques to be utilized. The representation for XY-PHOC can be put into a standard inverted index used for document retrieval with only small modifications for scoring. Scoring matches for XY-PHOC embeddings is performed with an optimized rank-equivalent cosine similarity.

Interestingly, the XY-PHOC retrieval model was designed with formula autocompletion rather than formula retrieval in mind, as initially we believed that formula search using symbol positions alone would be over-constraining and produce many false negatives/misses. The ARQMath results obtained indicate that this is not the case, possibly because the presence of a symbol anywhere in a formula is represented. The XY-PHOC symbol location retrieval model can be applied as an autocompletion model or a similarity search model simply by switching from conjunctive queries (requiring all query symbols) to disjunctive queries (requiring at least one query symbol).

Currently there are only two parameters in our model, controlling the number of regions and the function used to identify which regions a symbol belongs to. Scoring is performed using cosine similarity over binary vectors defined for individual symbols. Indeed, our symbol location vectors may be understood as a form of ‘term location frequency’ vectors. Our retrieval model is nearly identical to the TF portion of a standard TF-IDF retrieval model over words, but where the presence of symbols in multiple overlapping regions are captured, rather than the number of symbol occurrences.



(a) Formula Query (b) XY-PHOC Symbol Location Embeddings

Figure 2: Example 4-level XY-PHOC Embedding, with 19-bit Vectors for each Symbol. Levels are encoded by increasing numbers of regions: the first bit represents Level 1 (whole formula), the next two bits Level 2 (horizontal), followed by Level $2'$ (vertical), and so on. Levels are illustrated in Figure 1.

2. XY-PHOC: Bidirectional Pyramidal Histogram of Characters

In this section we describe the XY-PHOC model, which generalizes Pyramidal Histogram of Character (PHOC) bit vectors representing character locations in words [7] to capture horizontal and vertical symbol positions within formulas. An illustration of XY-PHOC symbol locations and their vector representation can be found in Figure 1, and Figure 2 illustrates how the bit vector per symbol looks using the first four levels.

The Pyramidal Histogram of Characters (PHOC) embedding model generalized for XY-PHOC comes from the field of word spotting [7]. In word spotting, the goal is to retrieve word images matching a query word given as either a character string (e.g., in UTF-8) or raster image. PHOC is a binary vector indicating the presence of a character within each horizontal region. For each level n there are n regions beginning with one word-level region (level 1), left and right regions after splitting the formula in half (level 2), followed by additional identically-sized splits up to n regions. The presence of a symbol in region(s) at each level gives the represention of locations where the symbol appears. After region embeddings are concatenated, a total of $n(n+1)-1$ regions are represented. So for example, using up to 5 uniform splits in the horizontal and vertical directions, we obtain $5 \cdot 6 - 1 = 29$ regions, which will be represented as a 29 bit vector.

This embedding was inspired by how the redundant spatial information in the character embeddings make it suitable for capturing both coarse and precise locations for symbols in space, making it useful for *visual* formula autocompletion. As Level 1 captures all symbols present in a formula, this along with the redundant spatial information are well suited to finding formulas similar to a query, starting from the symbols they share.

The original PHOC embedding is very sparse, as it uses a vector the length of the alphabet per region to indicate which symbols are in that region. For math the symbol vocabulary is much larger than the Latin alphabet, making the vector even longer. To use XY-PHOC efficiently, we use an inverted index over symbols with each posting as a pair $(id,\ v)$ containing a formula identifier and a bit vector representing the XY-PHOC regions where that symbol appears.

For spatial regions, we use five levels in both the horizontal (X) and vertical (Y) directions, making a symbol-specific XY-PHOC vector 29 bits long, which we store in a 32-bit integer. Formulas tend to be much wider than tall; we found in experiments that we could better distinguish the vertical location of symbols using a single point, and that using a line to represent the horizontal extent of symbols provided helpful redundancy [4].

An example of a simple 4-level embedding is shown in Figure 2. When encoding the binary XY-PHOC vectors, as seen in Figure 1, symbols are represented by a horizontal line that spans the width of the symbol's bounding box, with the line positioned at the vertical center of the bounding box. If any point on this line is included in a horizontal region, the bit for that region is set to 1. Vertical region memberships are identified by the vertical center of the box (a single point). This encoding represents the presence of the symbol in each region, and if a symbol appears in a region more than once it is still represented by a 1. For each horizontal level n greater than 1, there will be a level n' which represents the vertical splits at that level.

We index each formula identified as visually distinct in the ARQMath corpus once using a single instance (e.g., aiming to have just a single entry for x^2). XY-PHOC symbol location vectors are computed starting from formulas given in LaTeX using the MathJax javascript library[1]. Once formulas have been converted to SVG images, we convert these to lists of symbols with bounding boxes, which are then used to compute our XY-PHOC symbol embeddings. In order to index the large ARQMath collection, several tools are employed. We perform indexing using Apache Spark[2], using a distributed map-reduce implementation that ultimately produces the index in a text file. The index is then loaded into a Redis[3] database for use in retrieval. A second index maps formula ids to their original file, the normalization constant for each XY-PHOC vector as described in Section 2.1, and the number of symbols in the formula.

An important advantage of the reduced representation of XY-PHOC is that standard information retrieval techniques and tools can be used to generate an efficient and robust system. Other math formula retrieval systems that work on graph representations need to use custom solutions in order to index the paths that make up formula graphs, whereas the XY-PHOC embedding easily fits into standard tools for text-based search engines.

2.1. Scoring using XY-PHOC Symbol Location Vectors

We are able to retrieve and score formulas in a manner similar to conventional TF-IDF retrieval over words - each query symbol is looked up in the index, XY-PHOC vectors from formulas containing the symbol are retrieved, and then compared to the query symbol XY-PHOC vector. Symbol location match counts are then accumulated across query symbols, and finally scaled by candidate formula size, giving preference to smaller formulas when two or more candidate formulas have the same number of matching symbol locations.

More concretely, to score candidate formulas, we follow the work of Sudholt et al. [8] who demonstrated that in the word-spotting context, cosine similarity works well for scoring words represented as one-dimensional PHOCs.

For query vector **a** and candidate formula vector **b** the cosine similarity is:

$$cos = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||\ ||\boldsymbol{b}||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}. \quad (1)$$

A faster rank-equivalent similarity metric $bcos$ is defined as:

[1] https://www.mathjax.org
[2] https://spark.apache.org
[3] https://redis.io

$$cos(\mathbf{a}, \mathbf{b}) \stackrel{rank}{=} bcos(\mathbf{a}, \mathbf{b}) = \frac{1}{\sqrt{|\mathbf{b}|_1}} |\mathbf{a} \wedge \mathbf{b}|_1 \quad (2)$$

The dot product of two binary vectors is the Hamming weight (number of 1s) in the logical AND of the vectors, which is equivalent to the L_1 norm ($||_1$). The normalization factor for query $\mathbf{a}$ is constant across candidate formulas, and so can be removed without altering the rank ordering. To accelerate computation, the normalization factor for $\mathbf{b}$ is pre-computed and stored in the formula (secondary) index for lookup at retrieval time.

Our first retrieval implementation is very simple. We perform term-at-a-time scoring, and do not make use of skip lists, score bounds (e.g., from MaxScore [9]) or alternative indexing strategies (e.g., block-max [10]). For ARQMath-1 (2020) topics, this resulted in an average query time of 566 seconds (i.e., roughly 9.5 minutes) per query using a Python implementation running on a desktop Linux system with an Intel i7-8700K CPU (3.7GHz) and 32GB RAM, using Redis [4]. For a rapid prototype this was workable, but in the future, we hope to produce a much faster implementation.

3. Formula Retrieval Results

In this section we will present the results for the XY-PHOC retrieval model on the ARQMath formula retrieval task [11]. We show results for both queries from ARQMath 2020 and ARQMath 2021. Note that our model has only two parameters (number of regions and the symbol region membership function), and only a small amount of training/fitting on the ARQMath 2020 collection was performed [4].

For formula similarity search, we use disjunctive queries (weak AND) over symbol location vectors. This requires that at least one symbol present in the query is present in a candidate for retrieval. This assumes that relevant candidates share at least one symbol with the query formula.

There are 45 ARQMath 2020 topics. For ARQMath-2, the top results from all teams are shown in Table 1. Our model obtained the third-highest nDCG′ score. In the table we see that XY-PHOC has a higher P′@10 than the baseline system and is within 1.7% in MAP′ and 8% in nDCG′ scores relative to the baseline system, Tangent-S [5]. Tangent-S is a much more complex model, making use of path-based retrieval on both Symbol Layout Trees (SLTs) and Operator Trees (OPTs), followed by aligning the query formula with candidates in both representations before producing a final score. The best-performing system (LtRall) is obtained by re-ranking Tangent-S results after including additional OPT and SLT tree embeddings and tree edit distances within a learning-to-rank framework, adding complexity and computational expense.

ARQMath 2021 has 60 topics. The top results from all teams is shown in Table 1. Our model obtained the third-highest nDCG′, MAP′, and P′@10. It is interesting to see both the Tangent-S system and LtRall systems performed less well then the XY-PHOC model, while two systems that scored lower than XY-PHOC for ARQMath 2020 topics (MathDowsers and approach0) obtained the highest metrics for ARQMath 2021 topics. Across the ARQMath 2020 and ARQMath 2021 topics, the performance of XY-PHOC model remains relatively stable. Among the top-3 runs for ARQMath 2021, all the systems had very similar scores, with the nDCG′ having a

Table 1
ARQMath Task 2 results. The run with the highest nDCG′ (in ARQMath-1) is shown for each team. For TU_DBS team, we considered the run available on both ARQMath-1 and -2.

		Evaluation Measures					
		2020			2021		
Team	Run	nDCG′	MAP′	P′@10	nDCG′	MAP′	P′@10
approach0	P300	0.507	0.342	0.441	**0.555**	**0.361**	**0.488**
MathDowsers	formulaBase	0.562	0.370	0.447	0.552	0.333	0.450
XY-PHOC-DPRL	XY-PHOC	0.611	0.423	0.478	0.548	0.323	0.433
Baseline	Tangent-S	0.691	0.446	0.453	0.492	0.272	0.419
DPRL	ltrall	**0.738**	**0.525**	**0.542**	0.454	0.221	0.317
NLP_NITS	FormulaEmbedding_P	0.233	0.140	0.271	0.161	0.059	0.197
TU_DBS	TU_DBS_A2	0.157	0.085	0.122	0.154	0.071	0.217

Table 2
Examples of ARQMath 2020 queries (B.27 and B.29) for which the XY-PHOC had nDCG′@5 of 1. All the retrieved formulas have relevance score of High.

Rank	Query B.27 $e^{3i\pi/2}$	Query B.29 $i=\sqrt{-1}$
1	$e^{3i\pi/2}$	$i=\sqrt{-1}$
2	$e^{3\pi i/2}$	$i=\sqrt{-1}.$
3	$3e^{3i\pi/2}$	$i=\pm\sqrt{-1}$
4	$3e^{3\pi i/2}$	$\pm i=\sqrt{-1}$
5	$e^{2i\pi/3}$	$i=\sqrt{-1},$

difference of 0.7%, MAP′ having a difference of 3.8% and P′@10 having a difference of 5.5%. It is interesting that all three of these models use one formula representation for retrieval: OPT for approach0, SLT for MathDowsers, and XY-PHOC for our system. It would be interesting to see how different retrieved formulas are between these systems, and whether an ensemble would produce stronger results.

To understand the types of queries the XY-PHOC system performs well and poorly on, we present some queries with their top-5 retrieval results from the 2020 ARQMath queries, after removing the formulas that are not assessed and so not used in computing the prime (′) evaluation metrics. In Table 2 topics B.27 and B.29 are shown. For both of these queries, the top-5 results have an nDCG′@5 of 1. The strong performance is due to the types of formulas marked with a relevance of High (3) - formulas with the same symbols, and roughly the same placement as the queries. For both queries, the top result is the query formula. Formulas in ranks 2-5 also contain the query symbols, with only slight changes in placement or additional punctuation such as a '.' or a ','.

Next we consider two queries that the system did not perform well on. XY-PHOC's preference for candidates with similar structure and symbols can be seen in the results for topic B.2 and B.32 presented in Table 3. Both Topics have an nDCG′@5 of 0, meaning no formulas in the collection annotated as relevant were returned in the top 5 after removing unassessed formulas.

Table 3
Examples of ARQMath 2020 queries (B.2 and B.32) for which the XY-PHOC had nDCG′@5 of 0. Results shown after deduplication and removing unassessed hits.

Rank	Query B.2 $\frac{df}{dx} = f(x+1)$	Query B.32 $Empty(x) \iff \nexists y(y \in x)$
1	$\frac{df(x)}{dx} = e^{x+1}(x+1)$	$(\forall x)(\exists y)(\forall z)(z \in y \iff (\exists t)(z \in t \;\&\; t \in x))$.
2	$\frac{dx}{dt} = f(x) \quad (1)$	$\psi(x) = \exists y(y \in x)$
3	$\frac{df}{dx} = f(x, y)$	$\phi(x) = \exists y\,(y \in x)$
4	$\frac{dx}{dt} = x(1+x)$	$(\nexists x)x \in E \wedge A \subseteq E \rightarrow (\nexists y)y \in A$
5	$\frac{dy}{dx} = x^x(\log x + 1)$	-

Table 4
XY-PHOC Results on ARQMath 2020 Queries without Removing Unrated Formulas (nDCG′@5 = 0).

Rank	Query B.2 $\frac{df}{dx} = f(x+1)$	Query B.32 $Empty(x) \iff \nexists y(y \in x)$
1	$\frac{df}{dx} = (f(x) + x)x$	$[\neg\forall x \in y\ (p(x)] \iff \exists x \in y\ (\neg p(x))$.
2	$\frac{d}{dx}f(x) = c \cdot f(x+1)$	$\forall x\exists y(p(x) \vee q(y)) \iff \exists y\forall x(p(x) \vee q(y))$
3	$\frac{d}{dx}f(x) = x^x(\ln(x) + 1)$	$x\ E(\mathscr{Q})\ y \iff \exists Q \in \mathscr{Q}\big(x, y \in Q\big)$
4	$\frac{d}{dx}f(x) = f(x)\frac{d}{dx} + \frac{df}{dx}$	$y \in (x) \iff (y) \subseteq (x)$
5	$\frac{dF}{dx} = f(xy)(x^2 + xy + 1)$	$\forall x\exists y p(x, y) \iff \exists x\forall y\exists z R(x, y, z)$

For topic B.32, after removing unrated formulas only 4 formulas remained in the results. To get a better understanding of the the formulas returned, we present results for these topics without removing unrated formulas in Table 4. Based solely on shared symbols and structure, it would appear these filtered formulas may be more similar to the query, and it is possible that results would be stronger if these formulas were annotated.

Having shown the extremes for topics that obtained nDCG′ scores of 1.0 and 0, we now present two query topics with an nDCG′@5 > 0.7 . These results are presented in Table 5 using topics B.12 and B.14. For both of these topics, the top retrieval result matches the query formula, with a relevance of High (3). The relevance scores then decrease with rank. In these two queries, relevant results are retrieved at a high rank based on the symbols locations.

As originally hoped, the XY-PHOC model generally works well for exact item retrieval, as often the embedding with the highest score will be the query formula. We apply our model to the problem of formula autocompletion in the next Section.

4. Formula Autocompletion Results

In this section we briefly present the results from formula autocompletion experiments reported by Avenoso [4]. The XY-PHOC system was originally designed with the goal of autocompletion in mind, and is the first math formula autocomplete system that allows symbols to be input in any order. For example, a query such as shown in Figure 2 can be used to retrieve formulas

Table 5
Examples of ARQMath 2020 queries (B.12 and B.14) for which the XY-PHOC with nDCG′ > 0.7. Ratings of 3 are high, 2 are medium, 1 are low.

	Query B.12		Query B.14	
Rank	$(1+i\sqrt{3})^{1/2}$	**Rating**	$y = xy' + \frac{1}{2}(y')^2$	**Rating**
1	$(1+\sqrt{3}i)^{1/2}$	3	$y = xy' + \frac{1}{2}(y')^2$	3
2	$4(1+\sqrt{3}i)^{1/2}$	2	$y = xy' + \frac{1}{2}y'^2$	3
3	$(1+i\sqrt{3})/2$	1	$yy' = \frac{1}{2}(y^2)'$	0
4	$(35+18i\sqrt{3})^{1/3}$	0	$x = \frac{1}{2}\,y + \frac{1}{2}\,(-y)$	0
5	$(1 \pm i\sqrt{3})/2$	1	$x^4 + y^4 = \frac{1}{2}(x^2+y^2)^2 + \frac{1}{2}(x^2-y^2)^2$	0

containing many different integrands between the integral and 'dx.' Further, the query symbols can be either entered, or selected from an existing formula in any order. Query symbols do not need to comprise a well-formed subexpression (e.g., an SLT), or to be entered as a text encoding from left-to-right (e.g., for LaTeX). Matching is purely spatial.

For autocompletion, we make some small but important changes from our formula similarity search: (1) we use conjunctive queries, requiring all symbols in the query to be present in all candidates, and (2) the number of symbols in each candidate formula must be no smaller than that for the query formula. This reflects that our autocompletion is intended to exactly match a portion of the target formula, and so formulas that do not meet these requirements are pruned during retrieval.

To study the effect of inputting symbols in different orders on XY-PHOC autocompletion performance, we tested four possible symbol input orders: Left-to-Right (a roughly 'standard' entry order), Right-to-Left, Outside-in alternating adding symbols from the left and right ends, and Middle-out starting at the middle symbol and alternating between adding a symbol on the left and right side of the query. The formulas used for autocompletion queries were taken from the set of visually distinct ARQMath formulas that have been assessed [4]. Results for the first 102 queries are shown in this section. In Figure 3 we show the Mean Reciprocal Rank (MRR) grouped by the percentage of symbols input from the target formula, using groupings of 10 percent. MRR reflects how close target formulas are to the top rank. In an autocomplete system we generally want our target formula to obtain as high a rank as possible using as few symbols as possible, so that the user can easily find it. Note that an MRR of 50% reflects that the harmonic mean of the rank for the 102 query formulas is rank 2, 25% is the same for rank 4, etc.

Looking at Figure 3, we see that the inside-outside order obtains substantially higher MRR scores throughout the input size range. Interestingly, the left-right order obtains the lowest score across the input size range, which might be caused by formulas being less unique at their left end (particularly given the stronger performance for entering symbols right-left), but we have not had time to confirm this. The improved performance for outside-in also reflects that anchoring the left and right ends of the XY-PHOC symbol location vectors after the first two symbols leads to more stable location vectors.

In terms of how these different input orders for formula autocompletion might be used, we see two possibilities. One is users placing symbols on a canvas, e.g., by recognizing a formula

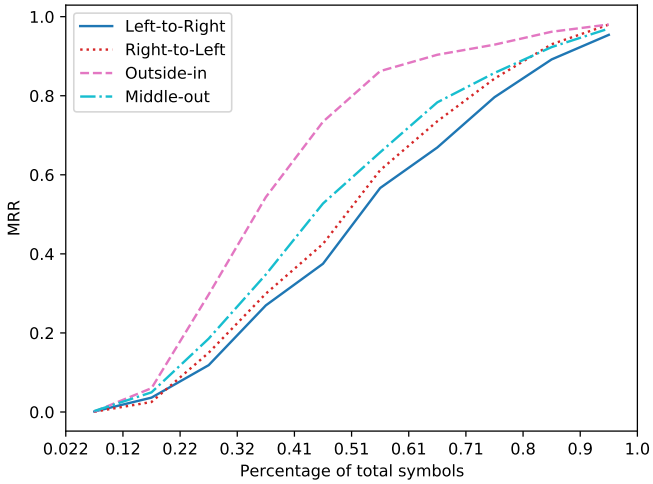


Figure 3: Autocompletion Results for Different Symbol Input Orders. Shown is the percentage of target formula symbols entered vs the Mean Reciprocal Rank (MRR) across target formulas.

image, using handwriting, or manually entering/placing symbols. Another is interactively selecting symbols in displayed formulas to define queries such as shown in Figure 2. To achieve this, we plan to integrate our XY-PHOC based autocompletion into the multimodal MathDeck search interface [12].[4]

5. Conclusion

We have presented a bidirectional PHOC embedding [4] that has been applied to the ARQMath 2021 Task 2 (Formula Retrieval). The system uses a simple embedding for relative symbol locations, in order to capture appearance without knowing the underlying writing or mathematical content for formulas. This simplified embedding can easily be used in standard search engines designed for text. For ARQMath-2, the XY-PHOC system ranked a close third based on highest nDCG′, MAP′ and P′@10 metrics, using what is perhaps the most naive formula representation, based only on the spatial arrangement of symbols. We also showed how XY-PHOC may be used for autocompletion by switching from disjunctive (weak AND) queries to conjunctive queries. The results from our autocompletion experiment are promising, and illustrate how a system could be built using XY-PHOC for both formula retrieval and autocompletion.

For similarity search, an obvious limitation of our approach is that formulas sharing no symbols (e.g., x^y and a^b) cannot be used to retrieve one another. We believe unified matches can be implemented through additional symbols in the index that encode mathematical types and symbol alphabets at different levels of granularity (e.g., variable, greek letter, set operation). We also have not incorporated inverse term frequencies into our scoring model, but believe that weighting rarer symbols higher in a TF-IDF or BM25 [13] manner may improve results substantially.

[4] https://mathdeck.org

Our current naïve implementation is very slow for the whole ARQMath collection, but we believe this can be improved with a more sophisticated implementation along with the use of rank-safe (e.g., skip lists, MaxScore [9]) and non-rank-safe (e.g., thresholded traversal of score-based sorted posting lists) optimizations.

The XY-PHOC symbol location retrieval model requires only symbol positions for indexing and search, without any representation of formula structure (visual, operation, or otherwise). This suggests that our approach might be applied to retrieving other two-dimensional notations and graphics (e.g., tables, plots, figures, chemical diagrams, etc.). It would also be worth exploring how additional levels of spatial partitioning beyond the currently used five vertical and horizontal splits improve or adversely impact retrieval effectiveness.

Acknowledgments

Our thanks to Wei Zhong for notes from his early investigations into formula autocompletion, and to Yancarlos Diaz and the RIT DPRL lab for helpful discussions. This material is based upon work supported by the Alfred P. Sloan Foundation under Grant No. G-2017-9827 and the National Science Foundation (USA) under Grant No. IIS-1717997. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation or the Alfred P. Sloan Foundation.

References

[1] B. Mansouri, A. Agarwal, D. Oard, R. Zanibbi, Advancing math-aware search: The arqmath-2 lab at clef 2021., in: European Conference on Information Retrieval, Springer, 2021.

[2] Experimental ir meets multilinguality, multimodality, and interaction., Proceedings of the Twelfth International Conference of the CLEF Association (CLEF 2021) (????).

[3] Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum (????).

[4] R. Avenoso, Spatial vs. Graph-Based Formula Retrieval, Master's thesis, Rochester Institute of Technology, 2021. URL: https://scholarworks.rit.edu/theses/10784.

[5] K. Davila, R. Zanibbi, Layout and Semantics: Combining Representations for Mathematical Formula Search, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Shinjuku Tokyo Japan, 2017, pp. 1165–1168. URL: https://dl.acm.org/doi/10.1145/3077136.3080748. doi:10.1145/3077136.3080748.

[6] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, R. Zanibbi, Tangent-CFT: An Embedding Model for Mathematical Formulas, in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 11–18. URL: http://doi.org/10.1145/3341981.3344235. doi:10.1145/3341981.3344235.

[7] S. Sudholt, G. A. Fink, PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents, in: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, Shenzhen, China, 2016, pp. 277–282. URL: http://ieeexplore.ieee.org/document/7814076/. doi:10.1109/ICFHR.2016.0060.

[8] S. Sudholt, G. A. Fink, Evaluating Word String Embeddings and Loss Functions for CNN-Based Word Spotting, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01, 2017, pp. 493–498. doi:10.1109/ICDAR.2017.87, iSSN: 2379-2140.

[9] H. Turtle, J. Flood, Query evaluation: Strategies and optimizations, Information Processing & Management 31 (1995) 831–850. URL: https://www.sciencedirect.com/science/article/pii/030645739500020H. doi:https://doi.org/10.1016/0306-4573(95)00020-H.

[10] S. Ding, T. Suel, Faster top-k document retrieval using block-max indexes, in: W. Ma, J. Nie, R. Baeza-Yates, T. Chua, W. B. Croft (Eds.), Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011, ACM, 2011, pp. 993–1002. URL: https://doi.org/10.1145/2009916.2010048. doi:10.1145/2009916.2010048.

[11] R. Zanibbi, D. W. Oard, A. Agarwal, B. Mansouri, Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math, in: A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, H. Joho, C. Lioma, C. Eickhoff, A. Névéol, L. Cappellato, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction, volume 12260, Springer International Publishing, Cham, 2020, pp. 169–193. URL: http://link.springer.com/10.1007/978-3-030-58219-7_15. doi:10.1007/978-3-030-58219-7_15, series Title: Lecture Notes in Computer Science.

[12] Y. Diaz, G. Nishizawa, B. Mansouri, K. Davila, R. Zanibbi, The mathdeck formula editor: Interactive formula entry combining latex , structure editing, and search, in: Y. Kitamura, A. Quigley, K. Isbister, T. Igarashi (Eds.), CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama Japan, May 8-13, 2021, Extended Abstracts, ACM, 2021, pp. 192:1–192:5. URL: https://doi.org/10.1145/3411763.3451564. doi:10.1145/3411763.3451564.

[13] B. Croft, D. Metzler, T. Strohman, Search Engines: Information Retrieval in Practice, 1st ed., Addison-Wesley Publishing Company, USA, 2009.