The Search for Equations – Learning to Identify Similarities between Mathematical Expressions

Lukas Pfahler (✉), Jonathan Schill, and Katharina Morik

TU Dortmund University
Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. On your search for scientific articles relevant to your research question, you judge the relevance of a mathematical expression that you stumble upon using extensive background knowledge about the domain, its problems and its notations. We wonder if machine learning can support this process and work toward implementing a search engine for mathematical expressions in scientific publications. Thousands of scientific publication with millions of mathematical expressions or equations are accessible at arXiv.org. We want to use this data to learn about equations, their distribution and their relations in order to find similar equations. To this end we propose an embedding model based on convolutional neural networks that maps bitmap images of equations into a low-dimensional vector-space where similarity is evaluated via dot-product. However, no annotated similarity data is available to train this mapping. We mitigate this by proposing a number of different unsupervised proxy tasks that use available features as weak labels. We evaluate our system using a number of metrics, including results on a small hand-labeled subset of equations. In addition, we show and discuss a number of result-sets for some sample queries. The results show that we are able to automatically identify related mathematical expressions. Our dataset is published at `https://whadup.github.io/EquationLearning/` and we invite the community to use it.

Keywords: Applied Data Science Track · Data Science · Preference learning and ranking · Deep learning.

1 Motivation

Finding relevant scientific publications to your own research question with the tools currently available is a tedious and at times frustrating endeavor. Every day, huge amounts of new scientific manuscripts are published. On the pre-print service arXiv.org alone, more than 140,000 papers where uploaded in 2018[1]. This flood of scientific manuscripts is impossible to filter, index and organize manually. Hence scientists rely heavily on existing search engines like Google Scholar or Mendeley to find relevant content, mainly by using keyword search.

[1] `https://arxiv.org/stats/monthly_submissions`

Keyword search has limitations, because similar concepts are referred to using different terminology between disciplines or even between subfields. For instance in machine learning we talk about features and labels, whereas statisticians might refer to the same concepts as independent and dependent variables.

However, once an interesting candidate paper is identified, it is usually not immediately read thoroughly from beginning to end. Instead, the scientist might briefly scan the pages for clues that indicate if the paper is indeed relevant. The most useful clues for disciplines like computer science or physics are arguably the equations used to describe the main ideas. A trained reader can easily match patterns of concepts in equations, generalize between different notations and use extensive background-knowledge to estimate the relevance of a paper to their own research questions. We wonder whether machine learning can help automatize this process.

The service arXiv.org grants access to thousands of publications, many containing mathematical expressions. We process a sample of publications and derive a dataset of equations in a standardized format. While this dataset is not manually annotated in any form – particularly similarities are not labeled – it still contains valuable information on equations and their relations. Ultimately the goal is to exploit this information and provide a search-engine that allows users to find equations related to their own research.

To this end we train embedding models based on convolutional neural networks. These networks take an equation represented as a fixed-size bitmap image and embed it in a low-dimensional vector space. Similarities can then be evaluated by dot-product in a low-dimensional space. Since ground-truth similarities are not available, we propose a number of unsupervised proxy tasks that take the available information and use it to define weak labels that guide the training of the embedding model. We evaluate their usefulness in an empirical study. For evaluation purposes we have hand-labeled a small set of equation into categories. This way we can show that our embedding model is capable of grouping related equations into clusters.

The rest of this paper is structured as follows: After discussing related work and orthogonal approaches, we present the dataset we created in greater detail. In Section 4 we present the unsupervised learning tasks that we design to train embedding models as well as the convolutional neural networks we use to tackle the tasks. Particular emphasis is put into choosing suitable loss functions. Section 5 presents our empirical study; after presenting our validation dataset and the metrics used to measure our success, we do a quantitative and qualitative analysis of our models. Finally we summarize our contribution and present directions of future work.

2 Related Work

The idea to implement a search engine for mathematical expressions is not new, see for instance [21,20,28,6,33,34,8]. Most of these systems work on the level of symbols and use inverted index structures designed for text to retrieve formulas.

Zannibi et al. distinguish between text-based and tree-based approaches [34] and propose a similarity measure based on the number of matching subtrees in a symbol layout tree derived from the LaTex source. Kamali and Tompa apply tree-edit-distance to retrieve similar equations [8]. NGuyen et al. use a tree-representation of the equation and derive features that are used in a tf-idf vector space[20].

To the best of our knowledge we propose the first system that represents mathematical expressions as bitmap images and uses machine learning to detect similarities. This way we can build on a rich set of works on similarity learning between images. The current boom of deep networks, and in particular convolutional neural networks, has heavily impacted computer vision. One important task is detecting similar images or similar objects in images or videos and a popular architecture for learning these similarities are Siamese networks, i.e. convolutional neural networks that are used to encode two images into a shared vector space where the similarity is measured. Examples of this application include identifying products in different product photos on e-commerce websites[27], person re-identification in photos[4,35], detection of models of vehicles [13]

Because we do not have labels for learning similarities, we proceed to view the problem as an embedding learning problem. Since Mikolov et al. proposed word2vec[17], these models have become increasingly popular for unsupervised learning of similarities. The underlying hypothesis of word embeddings is that words that appear together frequently share semantic similarities. This principle has been extended to other domains, including textual documents [12] or users in social networks [22]. We can view a whole publication as the context and compute embeddings of the equations that capture regularities in their distribution. We can construct a similar analogy to collaborative filtering [9,29], where we want to learn to recommend items to users based on the logged interactions of all users and items. Two items that are consumed by the same user may be similar. This problem is often formulated as a matrix factorization problem where items and users are mapped to low-dimensional embeddings such that their dot-product models the logged interactions. Now instead of the dichotomy of users and items, we have equations and documents, still we want to learn to identify similarities between equations. The major difference between our approach and both word embeddings and collaborative filtering is that we do not view equations as atomic units, which would be counterproductive as almost no equation appears identically in two documents, but as examples with features, in our case the grayscale information of a bitmap image.

Many machine learning tasks evolve around the use of LaTeX documents. There is plenty of research on converting images of equations back to valid LaTeX source [11,2]. We believe our data set might be a welcome addition to the available datasets of this branch of research. Even more challenging than parsing to LaTeX is parsing to a representation of the equation in a symbolic math description [23]. This could allow us to apply powerful solvers like Maple or Mathematica to judge similarities.

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\top x_i) + \lambda\|w\|^2$$

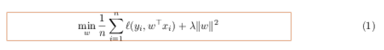

Fig. 1. Example equation (left) with corresponding bitmap graphic (right), the orange box marks the center rectangle used for learning. Note the decrease in visual clarity due to converting to a low-resolution bitmap.

3 The Dataset

The popular pre-print service arXiv.org hosts thousands of scientific publications; for the majority of these publications LaTex sources are available. This allows us to easily extract the mathematical expressions and represent them as images. Each publication has a unique arXiv-id. For this study, we only work on a small subset of publications, a combination of two publicly available crawls of arXiv-ids at Kaggle.com[2] and at Andrej Karpathy's page[3]. The publications in these crawls are mainly from machine-learning related subdomains of computer science. For each publication in these crawls, we try to download and extract the source files, obtaining more than 44k publications.

From these sources we extract snippets that describe mathematical equations as well as snippets that define new commands or macros. With these snippets, we are able to compile more than 600,000 equations; the major cause for failed compilation being more complicated user-defined LaTeX macros or the use of non-standard LaTeX packages. We have decided to work with bitmap images of equations rather than working with a text based representation like LaTex. This representation is invariant to the authors' LaTeX programming style, use of macros, etc. Additionally the LaTeX sources of a scientific publication are often unavailable, particularly on platforms other than arXiv, but images of equations may be identified and cropped from digital documents automatically in future projects. The images we create are 531×106 pixels in size, however for more efficient learning we only use the center rectangle of size 333×32 pixels for training. This decrease in height does not affect single-line equations, the decrease in width does not affect equations that fit into a single column in a two-column layout, longer equations will miss beginning and end. See Figure 1 for an example equation and the corresponding bitmap.

In addition to the image data, for each equation we know the arXiv-id of the corresponding publication. For each publication we know the title as well as the abstract, additional meta-data could be obtained via this identifier at a later point.

We have applied only rudimentary data-cleaning on the images. Most notably, we omit images that are mostly white background and have only few black foreground pixels. The majority of these images are artifacts of failed LaTeX compilation.

[2] `https://www.kaggle.com/neelshah18/arxivdataset`

[3] `https://cs.stanford.edu/people/karpathy/arxiv_10K_export.zip`

Our dataset is published at `https://whadup.github.io/EquationLearning/` and we invite the community to use it.

4 Unsupervised Training

Usually similarities between objects are learned in a supervised manner, where a dataset of labeled pairs is available. Since we do not have labeled data, we propose a number of unsupervised proxy training tasks for learning embeddings. All our proposed embedding models are based on the same encoder architecture that embeds a bitmap equation x into a low-dimensional vector space. We denote this operation by $f(x) : \mathbb{R}^{333\times 32} \to \mathbb{R}^d$. These embeddings can then be used for computing similarities between equations. We compute the similarity between two equations by taking the dot-product of the embeddings

$$s(x_1, x_2) = \langle f(x_1), f(x_2)\rangle.$$

This allows us to store all embeddings in an efficient index structure, e.g. a navigable small-network graph [15], and efficiently perform similarity search by embedding the query. Depending on the particular proxy task, we try various loss functions for optimizing our models.

4.1 Equation-Encoder

Our equation-encoder is a standard convolutional neural network. We try two different networks, a small and a large one. The small model is comprised of three convolution layers with 32 channels, each followed by a MaxPooling layer, followed by a final fully connected linear layer, as depicted in Figure 2. The larger model uses 6 convolution layers with 64 channels each, every other layer followed by a MaxPooling layer. Then the embedding is computed by a two fully connected layers. We apply batch normalization between the fully connected layers for more stable optimization, but freeze that layer in later epochs of training. Both models map into a space of dimensionality $d = 64$. All non-linearities are rectifying linear units (ReLU). In total, the small model has 45,504 parameters, while the number of parameters in the large model is 255,040.

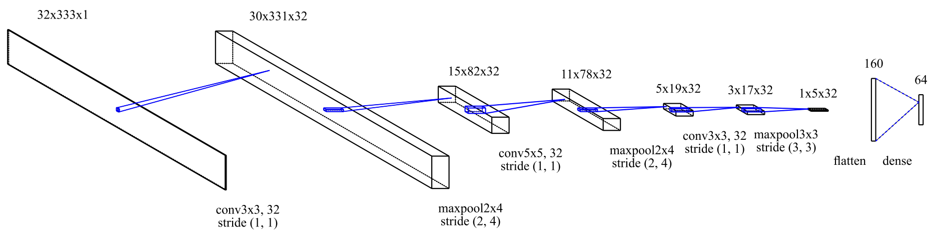

Fig. 2. Equation Encoder Architecture (small)

4.2 Learning Tasks and Loss Functions

We want to learn an encoder that embeds two equations into a low dimensional vector space, such that the dot product between two embeddings is large if and only if the corresponding equations are similar. Unfortunately, in our application we do not have gold-standard labels for similar and dissimilar pairs available. We design a number of unsupervised proxy tasks that use the available features as weak labels and argue why they are related to the real task.

Latex-Symbols as Labels The first proxy task we propose is to treat expressions in the available LaTeX-code as labels. We train a network based on our equation encoder, that has to decide if a given equation contains a selected symbol. This way the convolutional equation encoder has to learn to identify letters, numbers or symbols in an equation, a task loosely related to optical character recognition.

Abstract-Keywords as Labels The second task uses the abstract that is available for each publication to derive labels. We automatically extract keywords from those abstracts using RAKE [24] and label equations with those keywords. This task tries to link equations to background knowledge about research domains, particularly the objects of investigation. Prominent keywords include 'deep learning', 'optimization', 'classification', etc. This task is very similar to a weak-label task proposed by Denton et al. who train to label images on social media platforms by the hashtags used to describe the images [3].

Both of these tasks are multi-label classification tasks with a large number of classes. We choose a negative-sampling approach for training [16]: For each possible label y we have a weight vector $w_y \in \mathbb{R}^d$. We jointly learn the encoder and these weights, such that the dot-product of $\langle f(x), w_y\rangle$ is large when y is in the set of labels of x and small otherwise. To train, we randomly generate pairs of equations and labels; with probability 0.5 we create a pair with a correct label, otherwise we choose an incorrect label uniformly at random. This process generates data (x, y, z) where $z \in \{0, 1\}$ describes whether the label y fits the data x or not. Treating this as a binary classification problem with two inputs, we train by minimizing the cross entropy loss of the logistic function $\sigma(a) = 1/(1+\exp(-a))$ applied to the scalar product

$$\ell_{log}(x, y, z) = -z \ln(\sigma(\langle f(x), w_y\rangle)) - (1-z)\ln(1-\sigma(\langle f(x), w_y\rangle)). \qquad (1)$$

Document-Context as Labels Finally, we propose a proxy similarity learning task by choosing weak labels for pairs of examples. We say that two equations are similar if they appear in the same publication. This task is reminiscent of the word2vec embedding model by Mikolov et al. [17] popular in natural language processing, where two words are similar if they appear in the same context in a document. While it appears plausible that two equations in a publication are somehow related, the inverse is not necessarily true: Two equations can be related if they are from different documents. In fact it is our goal to identify those related documents. Like Mikolov et al., we use negative sampling to address this: We create artificial pairs of dissimilar equations by sampling equations from our collection uniformly and proclaiming them as dissimilar. We conjecture that the probability of accidentally sampling a similar pair is sufficiently small. Hence the labels should have a level of noise that still allows learning [10].

We explore two different types loss functions for this task. The first one works on annotated pairs of equations. Mobahi et al. [19] propose the *Siamese loss*: Given a margin $\varDelta$, the dot product of the embeddings should be larger than $\varDelta$ for a similar pair and less than $-\varDelta$ otherwise. We use $y = \pm 1$ to encode the similarity and define

$$\ell_{siam}(x_1, x_2, y) = \max(0, \varDelta - y\langle f(x_1), f(x_2)\rangle). \tag{2}$$

The equation encoder is used twice to calculate the loss, motivating the term 'Siamese' network, because we can also view this as two convolutional networks that share all their weights (twins) and that are connected by the loss function.

The second loss works on triples of equations, so that the encoder is used three times per training example: Given an equation x, the so-called anchor, a similar equation x_+ and a dissimilar equation x_-, we want the encoder to judge the similar pair as more similar then the dissimilar pair. Balntas et al. propose to formulate this as a margin loss as well [1]. Furthermore they propose to swap the roles of anchor example and the positive example if this yields a larger loss value to obtain a more strict loss:

$$\ell_{tri}(x, x_+, x_-) = \max \begin{cases} 0 \\ \varDelta - \langle f(x), f(x_+)\rangle + \langle f(x), & f(x_-)\rangle \\ \varDelta - \langle f(x_+), f(x)\rangle + \langle f(x_+), & f(x_-)\rangle \end{cases} \tag{3}$$

Margin losses require that the magnitude of embeddings is controlled either by applying regularization or normalization. Otherwise arbitrary absolute margins can be satisfied by scaling the output by a large enough values. We choose to normalize our embeddings to unit length $\bar{f}(x) = f(x)/(||f(x)||_2 + \epsilon)$ where a small, constant $\epsilon > 0$ is added for numerical stability. This choice avoids the introduction of yet another hyperparameter that controls regularization.

Following Janocha and Czarnecki [7] we use squared hinge losses ℓ^2_{siam} and ℓ^2_{tri} instead of hinge losses for faster convergence.

4.3 Training Procedure

We train our models using the ADAM optimizer, a variant of stochastic gradient descent. We initialize the learning rate at 0.00025 and reduce it by a factor of 0.1 every 10 epochs of training. We train using minibatches of size 100 and run training for 30 epochs.

All of our training tasks involve negative sampling; the triples task sample an additional positive point. For every gradient step, we sample new negative/positive samples. This way we hope to avoid overfitting to the particular choices of pairs/triples used for training.

The margin losses use a margin $\Delta = 1$, the constant used for normalizing the encoder output is $\epsilon = 10^{-5}$. The filters of the convolutional neural layers are initialized uniformly in $(-k^{-1/2}, k^{-1/2})$ where k is the number of weights per filter. The weights of the linear layers are initialized uniformly in $(-m^{-1/2}, m^{-1/2})$ where m is the input dimensionality of the layer.

5 Evaluation

In this section we experimentally evaluate the models we proposed in the previous section. First we try to establish quantitative metrics for measuring the quality of our embeddings. To this end, we curate a small evaluation dataset with hand-labeled equations. Second, we do a qualitative evaluation and inspect a few queries and discuss the results.

5.1 Gold-Label Evaluation Data

Quantitative evaluation is difficult whenever we cannot compare to a ground truth. This problem arises in many unsupervised learning applications, including clustering [14] and embedding learning[18,26,5]. Word Embedding models are often evaluated by computing measures using a small, manually labeled set of word pairs. Following their experimental protocol, we hand-labeled about 100 equations into 13 categories of equations that we have identified as prominent in machine learning literature. These include equations describing the empirical risk minimization principle, the k-means objective, the dual formulation of the kernelized support vector machine, but also concentration inequalities and Rademacher averages used in the theoretical analysis of machine learning algorithms. Under our embedding model, equations that belong to the same category should have higher similarity than inter-category pairs. Thus quantitative measure of embedding quality are possible. Note that this labeled dataset is used exclusively for evaluation purposes and never for learning.

5.2 Metrics

To get a detailed picture of the quality of our embeddings, we evaluate a number of different metrics.

Margin Losses on Eval Data We can evaluate the Siamese ℓ_{siam} and Triplet losses ℓ_{tri} on all triples in the evaluation data. In contrast to the training phase, now the similarity is measured with respect to the hand-annotated labels and not with respect to the weak-label based on the arXiv-id.

Hard Ranking on Eval Data In addition to the triplet margin loss, we can also evaluate a discrete ranking loss, that is 0, if the anchor example is more similar to to the positive example than the negative example and 1 otherwise. We can evaluate this over all triples derived from the evaluation dataset and measure how many triples are ranked wrong.

Nearest-Neighbor Accuracy on Eval Data We compute leave-one-out estimates of the accuracy for 1-nearest-neighbor classification for the hand-labeled evaluation data. This metric measures how many nearest-neighbor pairs share the same category.

Loss on Hold-Out Data We evaluate the margin losses and the hard ranking loss on hold-out data. This allows us to assess whether we are overfitting to the training data. For comparability between models we use the weak label based on the document context for all models.

5.3 Quantitative Results

We investigate a total of 12 different models, using 6 small architectures and 6 large architectures. The first four models of both architectures are trained using the different proxy tasks, namely predicting latex symbols (Symbols), predicting keywords in the abstracts (Abstract), training using the document context with triples loss (Triples) and Siamese loss (Siamese). In addition we try a transfer learning approach where we initialize the network with the weights trained using the symbol-task (Symbols+Triples) or the abstract-task (Abstract+Triples) and then proceed to train the network on the triples-task. This way we hope to combine the benefits of both weak labels.

Table 1. Performance Metrics for Small Models. Best scores in bold letters.

	Symbols	Abstract	Triples	Siamese	Symbols +Triples	Abstract +Triples
Eval Accuracy	0.4712	0.5096	**0.6827**	0.5385	0.5385	0.6058
Eval Ranking	0.3762	**0.2355**	0.3675	0.4637	0.4408	0.3259
Eval Triples	0.8885	0.9033	0.8091	0.9208	0.8918	**0.7633**
Eval Siamese	0.9088	0.9437	0.8403	0.8777	0.8789	**0.8132**
Hold-Out Ranking	0.5520	0.4540	0.3341	0.3642	0.3353	**0.3331**
Hold-Out Triples	1.0342	0.9819	0.8015	0.8120	0.8040	**0.8010**
Hold-Out Siamese	0.9578	0.9739	0.8458	**0.8392**	0.8469	0.8439

We have summarized our results for the small architecture in Table 1 and for the large architecture in Table 2. First of all, we note that a larger model size is beneficial for the problem at hand. It allows us to obtain larger margins on both hold-out and evaluation data. Furthermore we note that pretraining has a negative effect on the nearest neighbor accuracy, but has advantages for the Siamese and Triples loss. Overall, the pure Triples-model or the Triples model with abstract pretraining seem to perform best. A notable exception is that the Abstract model has the best ranking performance on the eval dataset. This is probably due to the manual labeling process for curating the evaluation dataset. We used keyword search in the abstract to identify candidate papers for a given

Table 2. Performance Metrics for Large Models. Best scores in bold letters.

	Symbols	Abstract	Triples	Siamese	Symbols +Triples	Abstract +Triples
Eval Accuracy	0.4135	0.4808	**0.6827**	**0.6827**	0.5385	0.5673
Eval Ranking	0.3568	**0.2485**	0.2812	0.3598	0.3409	0.2848
Eval Triples	0.8736	0.7936	0.6921	0.7867	0.7555	**0.6715**
Eval Siamese	0.8886	0.8852	0.7680	0.8168	0.7861	**0.7530**
Hold-Out Ranking	0.5474	0.5186	**0.2942**	0.3048	0.3181	0.3098
Hold-Out Triples	1.0382	0.4851	**0.7245**	0.7314	0.7861	0.7397
Hold-Out Siamese	0.9480	0.9592	0.7972	0.7970	0.7916	**0.7897**

category of equations, the same information was used to train the model, while all other models did not have access to the abstract information.

We visually inspect the performance of the large Triples-embedding in a hierarchical clustering of our labeled evaluation data. The clustering is done using agglomerative clustering with the complete-linkage strategy and dot-product similarity. In Figure 3 we see that on the lower levels of the hierarchy many clusters belong to a unique category of equations and that the computed similarities are high. On the right side of the plot, we see that categories mix, but do so in a plausible manner: the connection from sgd (stochastic gradient descent) and convex (equations describing properties like convexity or Lipschitz continuity) is apparent, as articles analyzing the convergence of optimization algorithms rely heavily on these definitions. The connection between Rademacher complexities and concentration inequalities cannot be denied as well.

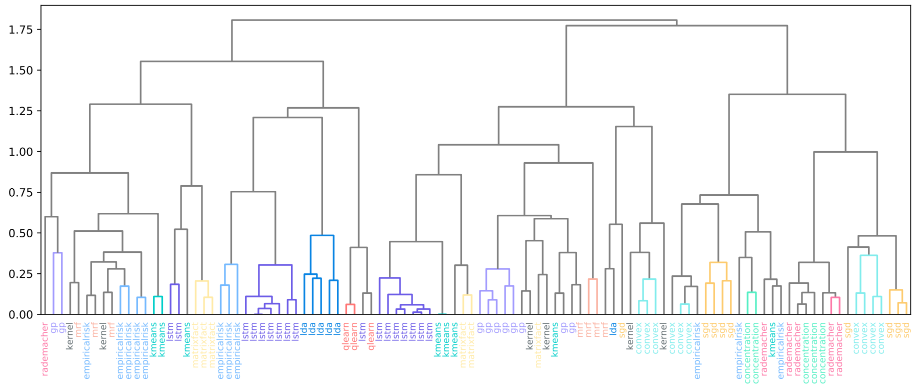

Fig. 3. A dendrogram showing a hierarchical clustering of our evaluation dataset according to our embedding model, colored according to hand-annotated labels, gray for non-pure clusters.

(q)	$c_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}} + W_c cov_t)$	$\mathcal{B}[p_\sigma] = \frac{T(\sigma, w_1^n) + 1}{n + \lvert\Sigma\rvert}$
(1)	$h_0 = f(W_{hh} h_{-1} + W_{he} e_0 + W_{hs} s + W_{hv} v)$	$\mathcal{B}[p_\sigma] = \frac{T(\sigma, w_1^n) + \alpha m_\sigma}{n + \alpha\lvert\Sigma\rvert}$
(2)	$h_0 = f(W_{hh} h_{-1} + W_{he} e_0 + W_{hs} s)$	$A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$
(3)	$y_t = \text{softmax}(W_o f(W_{dec} s_t + V_{dec} d_t))$	$C_M'(\kappa) = -C_M(\kappa) \cdot \frac{I_{s+1}(\kappa)}{I_s(\kappa)}$
(4)	$\beta_{tj} = v^\top \tanh(W s_{t-1} + U x_j + V h_j),$	$R(\pi) = \frac{\lvert G(\pi) \cap S_{pair}\rvert}{\lvert G(\pi)\rvert},$
(5)	$h_0 = f(W_{hh} h_{-1} + W_{he} e_0 + W_{vh} v)$	$Q(k, \alpha) = C_k \cdot \prod^{a} \frac{C_{i\alpha_i k}}{C_k}$
	(a)	(b)

Fig. 4. Sample Queries and Results: The first line shows queries, the remaining lines show the top-5 results. Shows center rectangle only.

(q)	$V^\pi(s) = \sum_{s' \in S} \mu(\pi(s), s, s')\Big(r(\pi(s), s, s') + V^\pi(s')\Big)$	$\mathcal{L}_r = \frac{1}{T} \sum^{L} \log p(Y, z^{(l)}) - D[q(z\lvert X)\lvert\lvert p(z)]$
(1)	$\hat{Q}(s, a) \leftarrow \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma \mathcal{P}(s, a, s') \bigotimes_{a'} \hat{Q}(s', a').$	$(\boldsymbol{\theta}_x^*, \boldsymbol{\theta}_y^*) = \arg \max_{(\theta_x, \theta_y)} \text{corr}(f_x(X; \theta_x), f_y(Y; \theta_y)).$
(2)	$b_{t+1}(\theta') = \eta \Pr(s', r\lvert s, a, \theta') \int_{\mathcal{S}, \Theta} \Pr(s', \theta'\lvert s, a, \theta) b_t(\theta) ds d\boldsymbol{\theta} \ ,$	$\hat{w}\,(Y, \boldsymbol{\theta}) \approx p\,(Y\lvert\boldsymbol{\theta}) = \int p\,(X, Y\lvert\boldsymbol{\theta})\, dX.$
(3)	$\frac{1}{q(\bar{a}\lvert s)} (\bar{Q}_w^\pi(s, a) - \bar{r} - \gamma \bar{Q}_w^\pi(\bar{s}', \mu(\bar{s}'))^2 \ .$	$p(z_t^k; X_t, L_{y_t^k}) \sim \mathcal{N}(L_{z_t^k} \ominus X_t, R)$
(4)	$g_\pi\,(s'\lvert s) = \sum_{a^1, a^2} \pi^1\,(a^1\lvert s)\, \pi^2\,(a^2\lvert s)\, p\,(s'\lvert s, a^1, a^2)\ .$	$\boldsymbol{\theta} = \underset{\theta}{\text{argmin}}\, \mathbb{E}_{P_{trg}(x)P(y\lvert x)}[-\log \hat{P}(Y\lvert X)] + \lambda\lvert\lvert\theta\rvert\rvert_2^2,$
(5)	$Q^A(s, a) = Q^A(s, a) + \alpha_t(s, a)(y - Q^A(s, a))$ $Q^B(s, a) = Q^B(s, a) + \alpha_t(s, a)(y - Q^B(s, a)).$	$\frac{\partial}{\partial\omega} \log p(y\lvert\hat{X}(\omega), \theta)$
	(a)	(b)

Fig. 5. More Sample Queries and Results: The first line shows queries, the remaining lines show the top-5 results. Shows center rectangle only. The results for these queries are more diverse.

5.4 A Qualitative Analysis of Example Queries

We choose to evaluate the large model without pretraining, because it beats the model pretrained with abstract keywords on 4 of 7 metrics. In Figure 4 we want to show some example queries with their respective top results[4]. In the first column we query an equation that describes a modified variant of an LSTM recurrent neural network. The system returns equations that seem to describe other recurrent network architectures or parts of recurrent units like the output part in row (3). In the second column the system does not work as well: The first result is relevant, however it is from the same publication as the query. That is a) not very useful for the user, and b) not difficult for the system to identify, as the query-answer-pairs were used during training. The remaining

[4] In the supplementary material you can find a table of the corresponding works

answers seem to have no semantic similarity. However they are at least visually similar: Short equation where an uppercase character function equals a term with a fraction. This difference in quality between the two columns is also expressed in the computed similarities. While the most relevant equation on the left has a similarity of 0.99, the most relevant equation on the right that is not from the same paper, has a similarity of only 0.91.

In Figure 5 we see more examples: On the left we query a definition of the value function, an expression related to reinforcement learning. And indeed all results are related to reinforcement learning. Even (2), which looks the most dissimilar, is taken from a survey on reinforcement learning[5]. On the right, the equations appear more diverse, but all involve probabilistic modeling. The computed similarities range between 0.93 and 0.90, which again is substantially lower than on the left.

Overall the quality of the results is convincing, the results contain useful and related equations in most cases. It seems that our system has learned to group equations that belong to a subfield of machine learning together in a latent space. Using a visual representation helps comparing the coarse structure of expressions.

6 Conclusion and Outlook

This work has worked towards the development of a search engine for mathematical expressions in scientific publications. We have created a dataset of equations that we used to learn to detect similarities between equations based on unsupervised training tasks. Our models work on bitmap image representations of equations and we use standard convolutional neural networks to compute low-dimensional embeddings of those equations[6]. In order to evaluate our system, we have hand-labeled real equations into categories. This way we tested how informative our embeddings are for grouping equations into categories and retrieving equations of the same category. A qualitative analysis revealed that or system is indeed capable of retrieving related equations.

We see many opportunities to further improve our system. First we believe it is crucial to refine the process we use to generate weak labels. Using the whole document context may be to broad and does not recognize that scientific publications have a sequential nature. We want to look into generating better weak labels by including the text in the paragraphs surrounding an equation. Furthermore we want to use the citation network to generate positive labels between two different documents. Generating better labels for learning also includes generating better negative labels. Currently our system generates negative examples by sampling uniformly at random, but more advanced sampling routines are possible.

[5] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar. *Bayesian Reinforcement Learning: A Survey*, 2016.

[6] We made both our code for computing embeddings and our dataset available at `https://github.com/Whadup/EquationLearning/`

In the future we want to generate training data not only based on weak labels, but also on simulations and symbolic math-solvers like SymPy or Maple. This way we hope to generate pairs of equations that not only share a semantic context, but that are actually equivalent pairs. We can also think about other forms of relations other than equivalence, e.g. lower- or upper-bounds. Preliminary experiments with a very simple grammar that allowed us to randomly generate pairs of equivalent equations showed a stark mismatch between the space of equations we encounter in real scientific publications and the space of synthetically generated equations. More advanced methods for synthetic generation of labeled data are necessary, maybe using advances in generative models or generative adversarial models [32].

Another point of attack for further improvements is the family of models used for embedding the bitmap equations into a vector space. A plain feed-forward convolutional neural network may very well be inferior to a more task-specifically designed model. Currently a lot of research investigates models that can solve simple mathematical problems [25,30]. Innovation from that area of research could inspire new models that can detect more advanced connections between pairs of equations.

Ultimately the goal is to have a running search system, which offers new exciting research possibilities in learning from logged interactions and hopefully provides the community a new tool for efficient literature search.

Acknowledgment

Parts of this work are based on the results of an undergraduate group project, we thank its participants Abdullah, Jan, Jonas, Lars, Marc, Marcel, Omar, Oskar, Sven and Xiazhe for contributing to building a prototype. Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A1. This research has been funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01|S18038A).

References

1. Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: Wilson, R.C., Hancock, E.R., Smith, W.A.P. (eds.) Proceedings of the British Machine Vision Conference (BMVC). pp. 119.1–119.11. BMVA Press (2016)
2. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-Markup Generation with Coarse-to-Fine Attention (2017)
3. Denton, E., Weston, J., Fergus, R.: User Conditional Hashtag Prediction for Images. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1731–1740 (2015)
4. Ding, S., Lin, L., Wang, G., Chao, H.: Deep Feature Learning with Relative Distance Comparison for Person Re-identification. Tech. rep. (2015)

5. Gladkova, A., Drozd, A.: Intrinsic Evaluations of Word Embeddings: What Can We Do Better? In: Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP. pp. 36–42 (2016)
6. Hu, X., Gao, L., Lin, X., Tang, Z., Lin, X., Baker, J.B.: WikiMirs: A Mathematical Information Retrieval System for Wikipedia. In: Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 11–20. JCDL '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2467696.2467699, `http://doi.acm.org/10.1145/2467696.2467699`
7. Janocha, K., Czarnecki, W.M.: On Loss Functions for Deep Neural Networks in Classification. Schedae Informaticae **25**, 1–10 (2017). https://doi.org/10.4467/20838476SI.16.004.6185, `http://arxiv.org/abs/1702.05659`
8. Kamali, S., Tompa, F.W.: Retrieving Documents With Mathematical Content. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 353–362. ACM
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009). https://doi.org/10.1109/MC.2009.263, `http://www2.research.att.com/{~}volinsky/papers/ieeecomputer.pdf`
10. Krishna, A., Brendan, M., Natarajan, N.: Learning from binary labels with instance-dependent noise. Machine Learning **107**(8), 1561–1595 (2018). https://doi.org/10.1007/s10994-018-5715-3, `https://doi.org/10.1007/s10994-018-5715-3`
11. Le, A.D., Nakagawa, M.: Training an end-to-end system for handwritten mathematical expression recognition by generated patterns. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 1056–1061. IEEE (2017)
12. Le, Q., Mikolov, T.: Distributed Representations of Sentences and Documents. In: International Conference on Machine Learning - ICML 2014. vol. 32, pp. 1188–1196 (2014). https://doi.org/10.1145/2740908.2742760
13. Liu, H., Tian, Y., Wang, Y., Pang, L., Huang, T.: Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2167–2175 (2016)
14. von Luxburg, U., Williamson, R.C., Guyon, I.: Clustering: Science or Art? In: JMLR Workshop and Conference Proceedings 27. pp. 65–79 (2012)
15. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems **45**, 61–68 (2014). https://doi.org/https://doi.org/10.1016/j.is.2013.10.006, `http://www.sciencedirect.com/science/article/pii/S0306437913001300`
16. Mikolov, T., Chen, K., Corrado, G., Dean, J., Chen, K., Dean, J.: Efficient Estimation of Word Representations in Vector Space. CoRR **abs/1301.3**, 1–12 (2013), `http://arxiv.org/pdf/1301.3781.pdf`
17. Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Advances in Neural Information Processing Systems. vol. abs/1310.4, pp. 1–9 (2013)
18. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, {USA}. pp. 746–751 (2013)

19. Mobahi, H., Collobert, R., Weston, J.: Deep Learning from Temporal Coherence in Video. Proceedings of the 26th Annual International Conference on Machine Learning (ICML) pp. 737–744 (2009)
20. Nguyen, T.T., Chang, K., Hui, S.C.: A Math-Aware Search Engine for Math Question Answering System (August), 724–733 (2012)
21. Nguyen, T.T., Hui, S.C., Chang, K.: A lattice-based approach for mathematical search using Formal Concept Analysis. Expert Systems with Applications **39**(5), 5820–5828 (2012). https://doi.org/https://doi.org/10.1016/j.eswa.2011.11.085, http://www.sciencedirect.com/science/article/pii/S0957417411016319
22. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. KDD pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
23. Raja, A., Rayner, M., Sexton, A., Sorge, V.: Towards a Parser for Mathematical Formula Recognition
24. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic Keyword Extraction from Individual Documents. In: Text Mining: Applications and Theory, pp. 1–20 (2010). https://doi.org/10.1002/9780470689646.ch1
25. Saxton, D., Grefenstette, E., Hill, F., Kohli, P.: Analysing Mathematical Reasoning Abilities of Neural Models pp. 1–17 (2019)
26. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (September), 298–307 (2015)
27. Shankar, D., Narumanchi, S., Ananya, H.A., Kompalli, P., Chaudhury, K.: Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce (2017)
28. Stalnaker, D., Zanibbi, R.: Math expression retrieval using an inverted index over symbol pairs. In: Document Recognition and Retrieval XXII. vol. 9402 (feb 2015), https://doi.org/10.1117/12.2074084
29. Verstrepen, K.: Collaborative Filtering with Binary , Positive-only Data. Ph.D. thesis, Universiteit Antwerpen (2015)
30. Wang, L., Wang, Y., Cai, D., Zhang, D., Liu, X.: Translating Math Word Problem to Expression Tree. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018. pp. 1064–1069 (2018), https://aclanthology.info/papers/D18-1132/d18-1132
31. Wu, F., Wang, Z., Zhang, Z., Yang, Y., Luo, J.: Weakly Semi-Supervised Deep Learning for Multi-Label Image Annotation. IEEE Transactions on Big Data **1**(3), 109–122 (2015). https://doi.org/10.1109/TBDATA.2015.2497270
32. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
33. Zanibbi, R., Davila, K., Kane, A., Tompa, F.W.: Multi-Stage Math Formula Search: Using Appearance-Based Similarity Metrics at Scale. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 145–154. SIGIR '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2911451.2911512, http://doi.acm.org/10.1145/2911451.2911512
34. Zanibbi, R., Tompa, F.W.: The Tangent Search Engine : Improved Similarity Metrics and Scalability for Math Formula Search (Section 6)
35. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-

Identification. IEEE Transactions on Image Processing **24**(12), 4766–4779 (2015). https://doi.org/10.1109/TIP.2015.2467315