Distributed Representations for Arithmetic Word Problems

Sowmya S Sundaram,[1] **Deepak P,**[2] **Savitha Sam Abraham**[1]
[1]Indian Institute of Technology, Madras, India
[2]Queen's University Belfast, UK
{sowmya, savithas}@cse.iitm.ac.in, deepaksp@acm.org

Abstract

We consider the task of learning distributed representations for arithmetic word problems. We outline the characteristics of the domain of arithmetic word problems that make generic text embedding methods inadequate, necessitating a specialized representation learning method to facilitate the task of retrieval across a wide range of use cases within online learning platforms. Our contribution is two-fold; first, we propose several 'operators' that distil knowledge of the domain of arithmetic word problems and schemas into word problem transformations. Second, we propose a novel neural architecture that combines LSTMs with graph convolutional networks to leverage word problems and their operator-transformed versions to learn distributed representations for word problems. While our target is to ensure that the distributed representations are schema-aligned, we do not make use of schema labels in the learning process, thus yielding an unsupervised representation learning method. Through an evaluation on retrieval over a publicly available corpus of word problems, we illustrate that our framework is able to consistently improve upon contemporary generic text embeddings in terms of schema-alignment.

Introduction

A word problem is a narrative involving a few sentences describing a real-life scenario expressing a task that needs to be solved by way of a mathematical calculation[1]. Central to any system that processes word problems electronically is a method to represent them in a meaningful way.

Notwithstanding the effectiveness of current embedding methodologies such as (Mikolov et al. 2013) in general NLP scenarios, there have been a number of efforts in improving text embeddings for specific tasks. As an example, SSWE (Tang et al. 2014) is a sentiment-specific word embedding that builds on the observation that generic word embedding models such as word2vec, being sentiment-agnostic, tend to put words of opposite sentiment polarity, such as *good* and *bad*, to neighboring locations in the vector space. Such characteristics render them inappropriate for sentiment-specific tasks, necessitating a sentiment-specific word embedding that SSWE provides. A similar motivation was leveraged in developing a methodology for emotion-enriched word representations (Agrawal, An, and Papagelis 2018). These indicate that bespoke techniques to learn specialized text embeddings are needed in order to make sure that the resultant representations are useful for specific tasks.

Coming to word problems, *schemas* (Marshall 1996) are classes of word problems meant to be indicative of the kind of solution methodology that is to be adopted to solve the problem. Table 1 lists three example word problems along with their *schemas*. A schema may be thought of as a categorization of word problems that is aligned with the learning task. For example, *combine* and *change* word problems can be solved effectively by students who have understood the intricacies of the mathematical operations involved in the respective domains. Given this mapping between schemas and learning tasks, schema-aligned retrieval of word problems is fundamental to automation in student-centric learning frameworks such as supportive scaffolding (Saxena 2010).

Lucy went to the grocery store. She bought 12 packs of cookies and 5 packs of noodles. How many packs of groceries did she buy in all? [**combine**]
Lucy was counting the two gifts she received on her fifth birthday. The first gift was a set of two toy cars. The second gift was a set of two rattle toys. How many toys did she receive altogether? [**combine**]
Bobby went to the grocery store. He bought 12 packs of noodles. He gave 6 packs to his friend, Lucy. How many packs does Bobby have? [**change**]

Table 1: Example Word Problems and Schemas

In Table 1, the first and second problems involve the *combine* schema, whereas the third problem involves a *change* (i.e., transfer). It may be observed that the first and the third problems relate to a grocery store context and have a reasonable amount of word overlap, and that is liable

[1]https://www.theschoolrun.com/what-is-a-word-problem

to force a generic text representation learner (e.g.,doc2vec (Le and Mikolov 2014)) to put them close to each other. However, a representation that puts the first and second problems closer to each other (as against the first and third) would be more appropriate for incorporation in educational support frameworks that seek to ensure that student learning is achieved through gaining mastery over learning tasks in a systematic fashion.

Our Contributions: In this paper, we present a specialized representation learning method for word problems, the first such method to our best knowledge. Our intent is to develop representations that are well aligned with word problem schemas, given that schemas correlate well with learning tasks. We propose a set of *operators* that fall into one of two categories; those that yield transformed word problems that fall within the same schema, and those that do not necessarily preserve schemas. We then propose a *novel neural architecture* that combines LSTMs and GCNs within a Siamese-inspired architecture designed to leverage both the text sequence as well as linguistic structures to accomplish learning vector embeddings for word problems. We then illustrate, through experiments on retrieval tasks over word problem datasets, that our representations are superior to contemporary generic text embedding methods in being schema-aligned, and thus are much more suited for usage in automated learning support scenarios.

Related Work

We briefly cover background and related literature across four different directions, viz., (i) methods for automatically solving word problems, (ii) ML techniques for representation learning, (iii) neural architectures targeted at text data, and (iv) positioning our task within e-learning.

Automatic Word Problem Solving: Arithmetic word problem solving has attracted interest starting from the 1960s (Bobrow 1964). The notion of *schemas* accelerated progress in this area as can be seen in (Fletcher 1985), (Bakman 2007) and (Sundaram and Khemani 2015). An extensive survey (Mukherjee and Garain 2008) offers perspectives on the literature. Since the 2010s, there has been a resurgence of interest ((Kushman et al. 2014), (Roy and Roth 2016), (Koncel-Kedziorski et al. 2015), etc.) in solving word problems automatically using empirical methods. There has been recent work in using neural architectures over word problems. An upcoming trend [2] is to model it as a neural machine translation task of 'translating' the word problem to the underlying equation (Robaidek, Koncel-Kedziorski, and Hajishirzi 2018).

Representation Learning: Dimensionality reduction offers a method of representing text in a compact fashion. Principal Component Analysis (F.R.S. 1901) is a popular method for general dimensionality reduction. Similarly, Latent Semantic Indexing (LSI) (Deerwester et al. 1990) is a text-oriented dimensionality reduction method using singular value decomposition. Newer methods for text documents include GloVe (Pennington, Socher, and Manning 2014), FastText[3] and doc2vec (Le and Mikolov 2014). Recently, contextualised word representations (Peters et al. 2018) have also been developed. Within deep learning approaches, auto-encoders (Baldi 2012) are most popular for representation learning. A popular variant, denoising autoencoders, (Vincent et al. 2008) learn noise-robust representations.

Neural Architectures for Text: Long-short term memory units (LSTMs (Hochreiter and Schmidhuber 1997)) are popular within NLP due to their ability to discern long-range sequential dependencies. Bidirectional LSTMs (Bi-LSTMs) are a variant of LSTMs that trace the document both in the forward and reverse directions. Many NLP tasks such as semantic parsing (Yang and Mitchell 2017), dependency parsing (Wang et al. 2018), etc. have achieved state-of-the-art results using LSTMs and Bi-LSTMs. Often, text in specific domains may have grammatical regularities that could be captured using dependency parsing (Nivre et al. 2007), a classical NLP task that generates a structured representation for the text. Such structured representations may be exploited by neural frameworks such as graph convolutional networks (GCNs) (Kipf and Welling 2016). Graph neural networks have been recently popular for tasks such as classification (Yao, Mao, and Luo 2019), sequence labelling (Liu et al. 2019) and so on. We use both LSTMs and GCNs in devising our methodology.

Positioning within E-Learning: Learning may be thought of as the process of encoding new knowledge in the learner's brain; effective *retrieval* of such encoded knowledge from the learner is a natural test of learning effectiveness. Recent research (Karpicke 2012) suggests that embedding the retrieval process more centrally within learning promotes effective and longer-term learning. While word problems have been an oft-used tool in mathematics teaching, they have been criticized for being presented in stereotyped fashion (Greer 1997). Online learning platforms have an opportunity to address both these issues by embedding contextual retrieval of word problems to ensure lexical/scenario diversity as well as effective long-term learning through invoking topical retrieval on the part of the learner. A representation learner that can take unlabelled word problems and produce a representation that is aligned with learning tasks would be a core building block in using word problem question banks to enrich *formative assessments* and instructional scaffolding (Shepard 2005). In the case of evaluation-oriented assessments, or *summative assessments*, using unlabelled data is trickier, given the need to ensure that the questionnaire covers problems overall specified learning tasks intended to be assessed. High accuracy tentative schema/task labels, such as those from a classifier, would help ease such questionnaire building. In this manner, both retrieval and classification can be of value for e-learning scenarios.

[2] https://sites.google.com/view/nlp-word-problem-solving/

[3] https://fasttext.cc/

Problem Definition

The task that we address in this paper is that of unsupervised representation learning for datasets of arithmetic word problems. Specifically, given a dataset of word problems, $\mathcal{W} = \{W_1, W_2, \ldots, W_n\}$, we would like to develop a representation learning method that would convert this to a dataset in a vector space $\mathbb{R}^d$, where d is a user-specified hyper-parameter. Formally:

$$\{W_1, W_2, \ldots, W_n\} \xrightarrow[\text{Learning}]{\text{Representation}} \{U_1, U_2, \ldots, U_n\} \quad (1)$$

where $U_i \in \mathbb{R}^d$ is a vector-space representation of the word problem W_i. We would like the resultant dataset in $\mathbb{R}^d$, denoted as $\mathcal{U}$, to be aligned well with the schemas of word problems. Specifically, we would like that word problem pairs that belong to the same *schema* be placed, on an average, close to each other than pairs that belong to different schemas. We address the problem of schema-aligned representation learning within an unsupervised setting, so the method would be applicable for scenarios where schema labelled data is not available.

Schemas

The schema of word problems, which is indicative of the learning task they relate to, plays a central role in the nature of the solution process to be used to solve the word problem. Automated word problem solvers (e.g., (Fletcher 1985)) often rely on schema-specific processing pipelines since the solution approach is often widely different across schemas. There is no universal standard of schemas; practitioners categorize word problems according to what suits their target dataset.

We do not make any assumptions in our method about the particular schemas present within a dataset. Further, our task being unsupervised, the schema categorization of the dataset is used only during the evaluation of the results of our representation learner. However, since an understanding of schemas is critical to appreciate our approach and evaluation metrics, we describe some schemas in our dataset briefly herein.

- *Combine* - Word problems of this type often have some entities being grouped as in a subset-superset relationship or a union of entities.
- *Change* - These word-problems model a sequence of events.
- *Vary* - 'Vary' word problems also describe a fixed relationship, but they are of a multiplicative nature.
- *Compare* - Comparison word problems describe a fixed relationship between two objects.

Methodology

Our task, as outlined in the problem definition, is to enable an unsupervised transformation from the space of word problems into vectors in $\mathbb{R}^d$ such that the vectors are aligned with the schemas of the word problem. There has been previous work in supervised learning which uses a similar construction; a pioneering work (Chopra, Hadsell, and LeCun 2005) targets conversion of a dataset of images into a vector space such that the notion of similarity in the vector space is aligned well with a notion of semantic similarity in the original space. Our task, being the unsupervised version of this task, is understandably harder.

Transformation Operators: We distil our knowledge of the domain of word problems into what we call as *operators*; these operators take a word problem text as input and produce a transformed word problem text as output. Our operators belong to one of two categories; *schema preserving* and *schema agnostic* operators. We will describe the nature of these operators and how these operators can be used to learn word-problem representations.

Neural Architecture for Representation Learning: Word problems along with their operator-transformed versions are then used within a purpose-built neural architecture to learn embeddings. Pairs of word problems (original, transformed) are fed into our neural architecture. The neural architecture is designed using BiLSTMs and GCNs to leverage the text as well as grammatical structures respectively, in our learning task. Once the training is finished, the neural framework can be used to obtain word-problem representations which then form the output of our task. We describe the two parts in separate subsections herein.

Transformation Operators

Our key insight in developing transformation operators comes from the observation that some linguistic structures within a word problem can be altered without changing the learning task associated with it. Analogously, the transformation of some other linguistic structures does not necessarily preserve the schema. This leads us to two kinds of operators for transforming word problems; *schema preserving* and *schema agnostic*.

Schema Preserving Operators Word problems often deal with people, entities and places. A transformation that replaces people names with other names would evidently not cause any change of schema. This intuition leads us to the definition of three schema preserving operators. These involve pre-processing using NER or POS Tagger from Stanford CoreNLP (Manning et al. 2014).

- *Person Change:* This operator involves replacing a person name within a word problem with another person name.
- *Place Change:* This replaces a place name with another.
- *Entity Change:* Analogous to the above, this replaces an entity mention (e.g., mango, orange etc.) with another.

Schema Agnostic Operators Unlike the case of schema preserving operators, changing certain linguistic structures could potentially (but not necessarily) lead to transforming the word problem to a different schema. We call them schema agnostic operators. They have been implemented using POS Tagger and WordNet (Miller 1995).

- *Verb Change:* This operator involves replacing a verb with another random verb from the dataset.
- *Adjective Change:* This operator changes the adjectives in a particular word problem. Adjectives are instrumental in identifying types of entities or the type of operation.
- *Antonym Introduction:* This operator replaces a verb or an adjective in the word problem with its antonym as obtained from WordNet.

Illustration of Operators We now illustrate the transformations caused by the various operators. Consider the following example *combine* schema word problem.

Mary put 5 red apples in a basket. John placed 2 red apples. How many apples are in the basket altogether?

The transformed versions of this problem under the various operators are listed in Table 2.

Operator	Transformed Word Problem	Schema
Person Change	**Stephen** put 5 red apples in a basket. **Clarissa** placed 2 red apples. How many apples are in the basket altogether?	[combine]
Place Change	Mary put 5 red apples in a **bowl**. John placed 2 red apples. How many apples are in the **bowl** altogether?	[combine]
Entity Change	Mary put 5 red **cherries** in a basket. John placed 2 red **cherries**. How many **cherries** are in the basket altogether?	[combine]
Verb Change	Mary put 5 red apples in a basket. John **ate** 2 red apples. How many apples are in the basket altogether?	[change]
Adjective Change	Mary put 5 red apples in a basket. John placed 2 **more** apples. How many apples are in the basket altogether?	[compare]
Antonym Introduction	Mary put 5 red apples in a basket. John **removed** 2 red apples. How many apples are in the basket altogether?	[change]

Table 2: Examples of Operator Application

It is pertinent to note that only one operator is applied at a time. For example, a single application of a verb change (by using one of 123 verbs available in the dataset) transforms a problem X to Y, whereas an adjective change would be separately applied to transform X to another version, Z. We trained a language model over a subset of the data and analyzed the conformance of the remainder as well as its transformed versions, using perplexity. The perplexities were 305.71 and 314.16 respectively, with 314.88 for the schema agnostic operator transformed subset. While this is a superficial analysis, it indicates that the well-formedness is not disturbed much.

Operator-driven Representation Learning

We now describe our method for operator-driven representation learning. First, we outline a novel neural architecture for the task. Second, we will illustrate how the operators introduced earlier can be used within the neural architecture to accomplish representation learning.

Bi-LSTM+GCN Siamese Neural Network Architecture Our neural network architecture, as illustrated in Figure 1, is designed to process a pair of word problems in the input; let $[W_i, W_j]$ represent the input word problem pair. Each word problem in the pair is converted into two kinds of representations; the *embeddings* representation, and the *dependency* representation. These are fed into Bi-LSTM and GCN layers respectively. The representations from the Bi-LSTM and GCN layers are then fused into a dense layer which yields a vector representation. While this is done separately for each word problem in the pair through replicating the Bi-LSTM and GCN layers (thus yielding separate dense layer representations), the weights for the neural layers are tied in typical Siamese network fashion. These weights are fine-tuned through a learning process described later.

Embedding Representation + Bi-LSTM Layer: Consider each word problem as a sequence of L words; shorter problems padded appropriately. For each word in the L length sequence in the input, the embedding representation is formed by simply replacing the word by its vector embedding from a pre-trained word embedding dataset. We use a dataset of GloVe (Pennington, Socher, and Manning 2014) embeddings trained over a very general dataset as the source of the word embedding vectors. This lookup transforms the sequence of L words in the input to an L-length sequence of word-embedding vectors. This sequence of L vectors is then passed through a Bi-LSTM layer which produces a numeric vector as the output.

Dependency Representation + GCN Layer: Each word problem is passed through a dependency parser to generate a dependency parse tree; we use the SpaCy dependency parser[4] to derive the dependency parse tree representation. This dependency parse tree is then passed through a graph convolutional network (GCN) (Wu et al. 2019) that makes use of graph proximity (as opposed to 2D proximity in conventional convolutional neural networks) in order to derive a vectorial representation.

Fusion and Weight Tying: The separate outputs from the Bi-LSTM and GCN above are then simply fused by means of a dense layer to form a d length numeric vector. This pipeline thus yields two separate d length dense layer representations for each word problem in the input pair. The pair of d length representations are then used to form a loss that would then be propagated backwards in a learning process, which is described later. While the d length representations are learnt separately for the two input word problems, we impose the constraint that the weights for the two separate Bi-LSTM and GCN layers be the same. Such weight-tied networks, referred to as siamese networks, have gained popularity in various contexts, notably in question-answer processing (Koch, Zemel, and Salakhutdinov 2015). The weight-tying is indicated in Figure 1 using the middle box with W indicating a common set of weights.

[4]https://spacy.io/usage/linguistic-features#dependency-parse

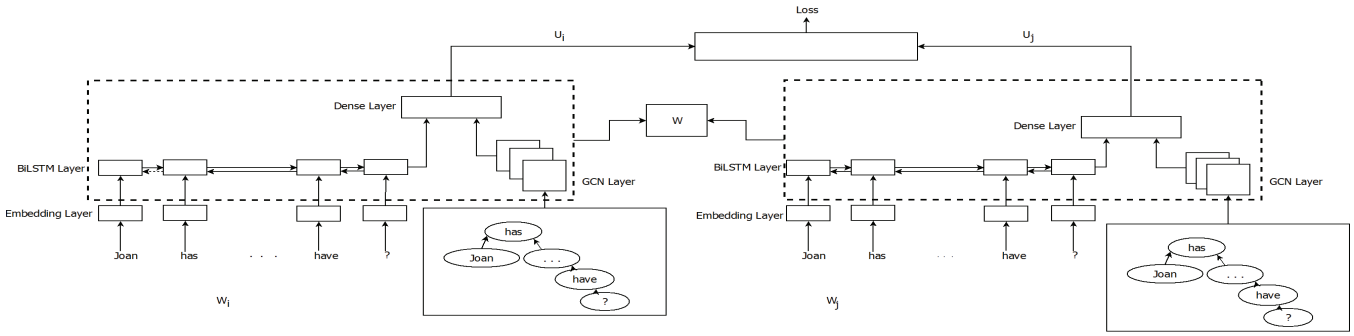

Figure 1: System Architecture

Training Process: The outputs from the two (Bi-LSTM/GCN)+dense layers are separate d length vectors, and these serve as the representations for the corresponding word problems. We refer to these vectors as U_i and U_j respectively (corresponding to the word problems W_i and W_j, the input pair). The training of the siamese network is facilitated by quantifying the distance between U_i and U_j. We use the standard formulation for contrastive loss (Hadsell, Chopra, and LeCun 2006) as follows:

$$\mathcal{L}(U_i, U_j) = \begin{cases} (ED(U_i, U_j))^2 & \text{if } [W_i, W_j] \in SP \\ (max\{0, 1.0 - ED(U_i, U_j)\})^2 & \text{if } [W_i, W_j] \in SA \end{cases} \quad (2)$$

where $ED(U_i, U_j)$ denotes the euclidean distance between U_i and U_j. SP is a set of word problem pairs where we expect the resultant U vectors to be as close as possible to each other; thus, the loss in this case is simply the square of the euclidean distance, that being the amount by which they deviate from the expectation. In the other set of word problem pairs, denoted by SA, we expect the resultant U vectors to be far away from each other, ideally at a distance of 1.0 or more. We will describe the construction of the sets SP and SA in the next section. The neural architecture is simply trained by passing word problem pairs, and propagating back the losses through the Bi-LSTM/GCN layers appropriately.

Training Dataset Construction We now outline the construction of the datasets SP and SA using the operators. A pair within SP is expected to comprise two word problems which belong to the same schema, whereas those in SA are not necessarily so. These pair datasets are created by using schema-preserving (SPO) and schema-agnostic (SAO) operators respectively:

$$SP = \{[W, W'] \,|W \in \mathcal{W} \wedge W' = SPO(W)\}$$
$$SA = \{[W, W'] \,|W \in \mathcal{W} \wedge W' = SAO(W)\}$$

where $SPO(W)$ and $SAO(W)$ indicate the transformation of W produced by a schema-preserving and schema-agnostic operator respectively. It may be noted that a particular word problem can be transformed by a chosen operator multiple times to yield multiple results given the possibility of choosing from among multiple replacement words.

Representation Learning: Having described our neural network architecture and the construction of its training dataset, we now describe the end-to-end method:

- **Training Dataset Creation** Convert the dataset $\mathcal{W}$ to two pair-datasets SP and SA using operators, which will be used in training.
- **Neural Network Training** Initialize the neural architecture in Figure 1 and train it using SP and SA as outlined.
- **Representations** Take one 'leg' of the network comprising Bi-LSTM & GCN layers (with weight-tying, both legs are identical), and pass each $W_i \in \mathcal{W}$ through them, reading off the d-length vector as U_i.

Experimental Study

Dataset

Though there are quite a few public datasets for math word problem solving (Koncel-Kedziorski et al. 2016), the only dataset which comprises word problems attached with unique schemas is the **SingleOp** (Roy and Roth 2016) dataset with 562 word problems. Thus, we choose this dataset for our empirical study. The dataset is manually annotated[5] by schema labels for each problem and these labels are used only during evaluation. The schema labellings were done by two graduate students, who labelled different subsets. On the overlapping subset, the agreement was 0.63 using the kappa statistic, which is 'substantial agreement' (Cohen 1960). The differences arose mainly due to confusion between 'change' and 'group' schemas. After unifying that dominant dissenting pair, we were able to get an agreement of 0.81 ('almost perfect agreement'). A third annotator sorted the errors to arrive at a final labelling.

Experimental Setup

Pre-trained GloVe embeddings of 100 dimensions were used in all the experiments as the source of the word vectors in the

[5] http://tiny.cc/word-probs-data

embedding layer. Wherever LSTMs were used, the state size was kept at 256. We undersampled the dataset for uniform representation across schemas. The GCN network had 4 convolutional layers with 200 hidden states each. The word problems in the dataset was subjected to 50 applications of each operator (as was mentioned earlier, using multiple replacement word candidates enables multiple applications of the same operator), resulting in an 'augmented' dataset of 27852 problems that will be used in training.

Retrieval-based Evaluation Framework

Our primary evaluation of the learnt representations is over a retrieval task. The proposed network is trained using word problems from $\mathcal{W}$ and their operator-transformed versions. After the training phase, we read off the d length vectors from one of the legs of the neural network; this forms a dataset $\mathcal{U}$ with each element of it having a one-to-one correspondence to $\mathcal{W}$. For each word problem $W_i \in \mathcal{W}$, we do the following to measure how well aligned the representations are, to the schemas of the word problems:

- Find the top-k most similar vectors to U_i from among the other vectors in $\mathcal{U}$, where similarity is assessed using the Euclidean distance.
- For each of the top-k retrieved results, *we check whether their corresponding word problem bears the same schema as* W_i; we mark each such result as correct, and others (i.e., those where the schema is different) as incorrect.
- We then use conventional Information Retrieval evaluation measures such as precision and Normalised Discounted Cumulative Gain (NDCG) (Wang et al. 2013) . While precision simply measures the fraction of correct/same-schema results in the top-k, NDCG takes into account the relative ordering within the top-k as well, in quantifying result set quality.

Baselines

We use two baseline approaches that perform unsupervised representation learning, in our evaluation. Each of these models is trained over the dataset augmented with operator-transformed versions of word problems, to ensure fairness in comparison. These are: (i) LSTM Autoencoder (lstm-ae) and (ii) LSTM Denoising Autoencoder (lstm-dae). Apart from this, we also compare against two variations of our architecture - using the Bi-LSTM layer or the GCN layer alone in the Siamese architecture; this helps illustrate the utility of using both text and grammatical information within the learning process. We use this baseline to perform an ablation study on the usefulness of the proposed components of our architecture. We describe the baselines below.

- **LSTM Autoencoder (lstm-ae):** Autoencoders are popular methods of deriving an object representation in an unsupervised manner. We leverage LSTM units in deriving a document embedding for word problems from a corpus of documents.
- **De-noising LSTM Autoencoder (lstm-dae):** This baseline is implemented by training LSTM autoencoders with the original input as well as noisy input corrupted by adding a random noise from $[0, 0.5]$ to each input element.
- **Bi-LSTM only (bilstm-sm):** This is our architecture which excludes the GCN layer and contains just the Bi-LSTM.
- **GCN only (gcn-sm):** Analogously, this architecture contains only the GCN layer and excludes the Bi-LSTM.

Comparative Evaluation over kNN Retrieval

We have evaluated the quality of embeddings for the original SingleOp dataset of 562 word problems. The precision and NDCG at various values of k are listed in Table 3, with the best value achieved for each k highlighted in boldface. We have performed statistical significance analysis based on the independent t-test (Student 1908). If our method beats the nearest competitor among the non-Siamese baselines (i.e., lstm-ae and lstm-dae) by a statistical significance of $p < 0.05$, we denote it with a $^\circ$ in the table. Similarly, if it has a statistical significance of $p < 0.01$, we have denoted it with a $^\bullet$. In a similar fashion, we include statistical significance results against the nearest Siamese baselines (i.e., bilstm-sm and gcn-sm), with * and * denoting statistical significance with $p < 0.05$ and $p < 0.01$ respectively.

As may be observed therein, our method consistently beats the baselines across the measures across a range of values of k. It is notable that the improvements recorded by our method decline very gradually with increasing k; this indicates that our embeddings are able to ensure a wider schema-homogeneous neighbourhood.

	Precision			NDCG		
k →	3	5	10	3	5	10
lstm-ae	0.7432	0.7021	0.6413	0.8653	0.8636	0.8494
lstm-dae	0.7419	0.7149	0.6489	0.8653	0.8602	0.8505
bilstm-sm	0.7272	0.6847	0.6276	0.8597	0.8607	0.8618
gcn-sm	0.7503	0.6943	0.6290	0.8729	0.8667	0.8522
Ours	**0.8114**$^{\bullet\star}$	**0.7790**$^{\bullet\star}$	**0.7129**$^{\bullet\star}$	**0.8973**$^{\bullet\star}$	**0.8956**$^{\bullet\star}$	**0.8876**$^{\bullet\star}$
Impr.	8.14%	8.96%	9.86%	2.79%	3.33%	2.99%

Table 3: Precision and NDCG of Schema-Based Retrieval

Classification

We now demonstrate how our embeddings can help classify word problems into the labelled schemas in Table 4. We use the SingleOp dataset across three different classifiers - (i) random forest classifier, (ii), AdaBoost and (iii) a multi layer perceptron with two hidden layers. We consistently provide better classification and the percentage improvement over the best is indicated.

Generalizability Study

Equation Homogeneity in Retrieval: The actual equations used in solving a word problem provide another indicator, though less reliable, of the learning task involved. Equations are less reliable indicators of the learning task than schemas since the same equation could lead to different learning tasks based on the choice of the unknown variable. We now evaluate the utility of our schema-aligned representations in

	Random Forest Classifier	Multilayer Perceptron	AdaBoost
lstm-ae	0.8407	0.8053	0.5310
lstm-dae	0.8318	0.8141	0.6106
bilstm-sm	0.8053	0.7434	0.5929
gcn-sm	0.8142	0.8584	0.5664
Ours	**0.8673**	**0.8761**	**0.6726**
Impr.	3.16%	2.06%	10.14%

Table 4: Classification Accuracy

ensuring homogeneity in the space of equations associated with the word problems. We converted the equations of the SingleOp dataset to templates by replacing the numerals with tokens such as $n1$, $n2$, etc. We replace the schema matching operation in the retrieval evaluation framework with exact match over equation templates. The results, tabulated in Table 5, illustrate that our method is able to outperform the baselines convincingly in this task as well.

	Precision			**NDCG**		
k →	3	5	10	3	5	10
lstm-ae	0.5546	0.4978	0.3989	0.6167	0.6409	0.7343
lstm-dae	0.5670	0.5103	0.4053	0.6131	0.6332	0.7284
Ours	**0.6797•**	**0.6203•**	**0.5162•**	**0.7203•**	**0.7249•**	**0.8032•**
Impr.	19.87%	21.55%	27.36%	16.79%	13.11%	9.38%

Table 5: Precision and NDCG of Equation-Based Retrieval

Transfer Learning: We now consider the generalizability of our embedding method, across different datasets. The MAWPS dataset (Koncel-Kedziorski et al. 2016) is a large dataset of word problems with no schema information, but with equation information across its 2373 word problems that are associated with single equations. Since it is an amalgam of many existing datasets, it has good variability. Having trained our model over the SingleOp dataset, we pass each of the MAWPS word problems in one forward pass to get an embedding for each of them. We then evaluate the equation-homogeneity in retrieval, much like in the previous experiment. The results, outlined in Table 6, indicate that the learnings over SingleOp transfer well to the MAWPS dataset, with our method consistently performing better than the baselines. We observed eq/schema correlations, though far from a one-to-one correspondence, given that equations and schemas are different concepts. For example, the template 'n1-n2' is split across 'change' and 'compare' schemas. The Cramér's V correlation (Cramir 1946) was found to 0.59 indicating a moderate correlation towards the higher side of the [0,1] range. There is a significant drop in both precision and NDCG from Table 5, which is expected given that the evaluation was performed over a different dataset; however, our method remains consistently better than the baselines.

Conclusions and Future Work

Developing embedding methods for text has been a subject of high interest within the NLP community in recent years. The nuances of arithmetic word problems require them to be treated differently from text documents, making embedding methods for general text documents inadequate over them. In this paper, we addressed this task and developed a framework for representation learning for arithmetic word problems that correlates well with the learning task involved. We proposed to represent the domain knowledge of the realm of word problems into a set of operators that can transform word problems in a schema-preserving or schema-agnostic manner; these, we illustrated, can be used to create an augmented training set for representation learning. We then devised a Siamese-inspired neural network that uses LSTMs and GCNs to use word problems and their operator-transformed versions in order to learn embeddings for word problems. Through an evaluation over a public dataset, we illustrated the consistent improvements that are achieved by our method over contemporary generic representation learners. To our best knowledge, our work presents the first approach for specialized representation learning for arithmetic word problems.

	Precision			**NDCG**		
k →	3	5	10	3	5	10
lstm-ae	0.4432	0.3766	0.2500	0.5478	0.5634	0.6063
lstm-dae	0.4438	0.3764	0.2498	0.5437	0.5653	0.6121
Ours	**0.4862•**	**0.4211•**	**0.3015•**	**0.5712•**	**0.5854•**	**0.6335•**
Impr.	9.55%	11.82%	20.60%	4.27%	3.55%	3.49%

Table 6: Precision and NDCG Results for Equation-based Retrieval on MAWPS dataset

Future Work

Having established an operator-driven framework for representation learning, the natural next step would be to devise sophisticated operators that can help improve on the retrieval accuracy further. On an application front, we are quantifying the gains obtained by using these embeddings within the retrieval component of an instructional scaffolding framework for formative learning. We expect our method to generalize, with appropriate simple modifications to operators, for tasks such as kinematics word problem solving and associated scenarios within other topics in the vast e-learning space. Generalizability to argumentation and entailment would be an interesting direction to explore, though we suspect those may call for non-trivial adaptations.

Acknowledgements

We would like to thank Tata Consultancy Services for supporting this work through a grant awarded to the first author. The second author was partially supported by SPARC (#P620). The authors would also like to thank Krishna M. Sivalingam (IIT Madras) for his constant support and encouragement.

References

Agrawal, A.; An, A.; and Papagelis, M. 2018. Learning emotion-enriched word representations. In *ICCL*.

Bakman, Y. 2007. Robust understanding of word problems with extraneous information. *arXiv - math/0701393*.

Baldi, P. 2012. Autoencoders, unsupervised learning, and deep architectures. In *ICML workshop on unsupervised and transfer learning*.

Bobrow, D. G. 1964. A question-answering system for high school algebra word problems. In *the fall joint computer conference*. ACM.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR 2005*, volume 1. IEEE.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological measurement* 20(1):37–46.

Cramir, H. 1946. Mathematical methods of statistics. *Princeton U. Press, Princeton* 500.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *JASIST* 41(6).

Fletcher, C. R. 1985. Understanding and solving arithmetic word problems: A computer simulation. *BRM* 17(5).

F.R.S., K. P. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The Philosophical Magazine*.

Greer, B. 1997. Modelling reality in mathematics classrooms: The case of word problems. *LAI* 7(4).

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*. IEEE.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8).

Karpicke, J. D. 2012. Retrieval-based learning: Active retrieval promotes meaningful learning. *CDPS* 21(3).

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Koncel-Kedziorski, R.; Hajishirzi, H.; Sabharwal, A.; Etzioni, O.; and Ang, S. D. 2015. Parsing algebraic word problems into equations. *TACL* 3.

Koncel-Kedziorski, R.; Roy, S.; Amini, A.; Kushman, N.; and Hajishirzi, H. 2016. Mawps: A math word problem repository. In *NAACL*.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. *ACL (1)*.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*.

Liu, P.; Chang, S.; Huang, X.; Tang, J.; and Cheung, J. C. K. 2019. Contextualized non-local neural networks for sequence learning. In *AAAI*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Marshall, S. P. 1996. *Schemas in Problem Solving*. Cambridge University Press.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*.

Miller, G. A. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11).

Mukherjee, A., and Garain, U. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review* 29(2).

Nivre, J.; Hall, J.; Kübler, S.; McDonald, R.; Nilsson, J.; Riedel, S.; and Yuret, D. 2007. The conll 2007 shared task on dependency parsing. In *(EMNLP-CoNLL)*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv:1802.05365*.

Robaidek, B.; Koncel-Kedziorski, R.; and Hajishirzi, H. 2018. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*.

Roy, S., and Roth, D. 2016. Solving general arithmetic word problems. *arXiv:1608.01413*.

Saxena, M. 2010. Reconceptualising teachers' directive and supportive scaffolding in bilingual classrooms within the neo-vygotskyan approach. *JALPP* 7(2).

Shepard, L. A. 2005. Linking formative assessment to scaffolding. *Educational leadership* 63(3).

Student. 1908. The probable error of a mean. *Biometrika*.

Sundaram, S. S., and Khemani, D. 2015. Natural language processing for solving simple word problems. In *12th International Conference on Natural Language Processing*.

Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; and Qin, B. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, volume 1.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*. ACM.

Wang, Y.; Wang, L.; Li, Y.; He, D.; Chen, W.; and Liu, T.-Y. 2013. A theoretical analysis of ndcg ranking measures. In *COLT*, volume 8.

Wang, Y.; Che, W.; Guo, J.; and Liu, T. 2018. A neural transition-based approach for semantic dependency graph parsing. In *AAAI*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *arXiv:1901.00596*.

Yang, B., and Mitchell, T. 2017. A joint sequential and relational model for frame-semantic parsing. In *EMNLP*.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *AAAI*.