'John ate 5 apples' != 'John ate some apples': Self-Supervised Paraphrase Quality Detection for Algebraic Word Problems

Rishabh Gupta*, Venktesh V*, Mukesh Mohania, and Vikram Goyal

Indraprastha Institute of Information Technology, Delhi
{rishabh19089,venkteshv,mukesh,vikram}@iiitd.ac.in

Abstract. This paper introduces the novel task of scoring paraphrases for Algebraic Word Problems (AWP) and presents a self-supervised method for doing so. In the current online pedagogical setting, paraphrasing these problems is helpful for academicians to generate multiple syntactically diverse questions for assessments. It also helps induce variation to ensure that the student has understood the problem instead of just memorizing it or using unfair means to solve it. The current state-of-the-art paraphrase generation models often cannot effectively paraphrase word problems, losing a critical piece of information (such as numbers or units) which renders the question unsolvable. There is a need for paraphrase scoring methods in the context of AWP to enable the training of good paraphrasers. Thus, we propose ParaQD, a self-supervised paraphrase quality detection method using novel data augmentations that can learn latent representations to separate a high-quality paraphrase of an algebraic question from a poor one by a wide margin. Through extensive experimentation, we demonstrate that our method outperforms existing state-of-the-art self-supervised methods by up to 32% while also demonstrating impressive zero-shot performance.

1 Introduction

Algebraic Word Problems (AWPs) describe real-world tasks requiring learners to solve them using mathematical calculations. However, providing the same problem multiple times may result in the learner memorizing the mathematical formulation for the corresponding questions or exchanging the solution approach during exams without understanding the problem. Hence, paraphrasing would help prepare diverse questions and help to evaluate whether the student can arrive at the correct mathematical formulation and solution[1].

The paraphrasing task can be tackled using supervised approaches like in [4] or self-supervised approaches like in [10]. As shown in Figure 1, we observed that the generated paraphrases are of low quality as critical information is lost and the solution is not preserved. Some common issues that arose for the paraphrasing models were replacement or removal of numerical terms, important entities, replacement of units with irrelevant ones and other forms of information loss. These issues result in the generated

* Equal Contribution
[1] https://cutt.ly/MWqHsN8

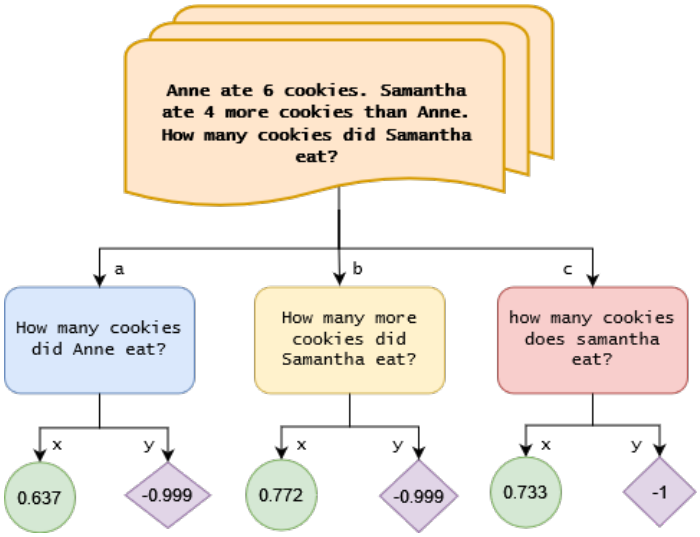


Fig. 1: Paraphrases by SOTA generation models. *a* is output from PEGASUS fine-tuned on PAWS, *b* is from T5 fine-tuned on Quora Question Pairs dataset and *c* is from PARROT paraphraser built on T5. *x* represents the cosine similarity scores assigned by the pretrained encoder MiniLM, while *y* represents the scores with our proposed approach, ParaQD.

question having a different solution or being rendered impossible to solve. Thus, there exists a need to automatically evaluate if a paraphrase preserves the semantics and solution of the original question. This is a *more challenging problem* than detecting similarity for general sentences. The existing state-of-the-art semantic similarity models give a relatively high score even to very low-quality paraphrases of algebraic questions (where some critical information has been lost), as seen in Figure 1 and Table 1. In Figure 1, our approach ParaQD assigns the cosine similarity as -0.999, thereby preventing the low-quality paraphrases from getting chosen. There is a need for solutions like ParaQD because poor paraphrases of algebraic questions cannot be given to the students as they are either unsolvable (as observed in the figure) or do not preserve the original solution.

To tackle the issues mentioned above, we need a labelled dataset for training a proper scoring model. However, there does not exist a dataset for AWP with labelled paraphrases. Therefore, we propose multiple unsupervised data augmentations to generate positive and negative paraphrases for an input question. To model our negative augmentations, we identify crucial information in AWPs like numbers, units and key entities and design operators to perturb them. Similarly, for the positive augmentations, we design operators that promote diversity and retain the crucial information, thereby yielding a semantically equivalent AWP. On the other hand, existing augmentation methods like SSMBA [11] and UDA [22] do not capture the crucial information in AWPs. Using the positive and negative paraphrases, we train a paraphrase scoring model using triplet loss. It explicitly

allows for the separation of positives and negatives to learn representations that can effectively score paraphrases. In summary, our core contributions are :

- We formulate a novel task of *detecting paraphrase quality for AWPs*, which presents a different challenge than detecting paraphrases for general sentences.
- We propose a new unsupervised data augmentation method that drives our paraphrase scoring model, *ParaQD*.
- We demonstrate that our method leads to a scoring model that surpasses the existing state-of-the-art text augmentation methods like SSMBA and UDA.
- We evaluate ParaQD using test sets prepared using operators disjoint from train augmentation operators and observe that ParaQD demonstrates good performance. We also demonstrate the zero-shot performance of ParaQD on new AWP datasets.

Code and Data are available at: `https://github.com/ADS-AI/ParaQD`

2 Related Work

This section briefly discusses prior work in text data augmentation methods. One of the notable initial works in data augmentation for text [26] replaced words and phrases with synonyms to obtain more samples for text classification. In the work [23], the authors propose noising methods for augmentations where words are replaced with alternate words based on unigram distribution, but it introduces a noising parameter. A much easier text augmentation method, EDA, was proposed in the work [21]. The authors propose several operators such as random word deletion and synonym replacement to generate new sentences. The above works are based on heuristics and depend on a hyperparameter for high-quality augmentations.

More recently, self-supervised text augmentation methods have provided a superior performance on multiple tasks. In UDA [22], the authors propose two text augmentation operators, namely backtranslation and TF-IDF based word replacement, where words with low TF-IDF scores are replaced. In SSMBA [11], the authors propose a manifold-based data augmentation method where the input sentences are projected out of the manifold by corrupting them with token masking, followed by a reconstruction function to project them back to the manifold. Another self-supervised augmentation method named InvDA (Inverse Data Augmentation) was proposed in Rotom [10] which was similar to SSMBA in that it tried to reconstruct the original sentence from the corrupted version. Several rule-based text augmentation methods have also been proposed, like [7] which uses Natural Language Inference (NLI) for augmentation, and [1] leverages linguistic knowledge for the question-answering task.

3 Methodology

In this section, we describe the proposed method for paraphrase quality detection for algebraic word problems. The section is divided into two components: Data Augmentation and Paraphrase Quality Detection.

3.1 Data Augmentation

For data augmentation, we define 10 distinct operators to generate the training set. Out of the 10, 4 are positive (i.e. information preserving) transformations, and 6 are negative (information perturbing) transformations. Our negative operators are carefully chosen after observing the common mistakes made by various paraphrasing models to *explicitly teach* the quality detection model to assign a low score for incorrect paraphrases.

Let $Q = \{Q_1, Q_2, Q_3, \dots Q_n\}$ denote the set of questions. Each question Qi can be tokenized into sentences $Q_{i1}, Q_{i2} \dots Q_{ip}$ where p denotes the number of sentences in question Q_i. Let an augmentation be denoted by a function f, such that $f_i(Q_j)$ represents the output of the ith augmentation on the jth question.

The function $\lambda : Q \times Q \mapsto \{0, 1\}$ represents a labelling function which returns 1 if the input (Q_i, Q'_i) is a valid paraphrase, and 0 if not. Based on the design of our augmentations (explained in the next section), we work under the following assumption for the function f:

$$\lambda(Q_a, f_i(Q_a)) = \begin{cases} 1, & 1 \leq i \leq 4 \\ 0, & 5 \leq i \leq 10 \end{cases}$$

For the purposes of explanation, we will use a running example with question $\mathbf{Q_0} =$ *Alex travelled 100 km from New York at a constant speed of 20 kmph. How many hours did it take him in total?*

3.2 Positive Augmentations

f_1: Backtranslation Backtranslation is the procedure of translating an example Q_i from language A to language B, and then translating it back to language A, yielding a paraphrase Q'_i. In our case, given an English question Q_i comprised of precisely p sentences $Q_{i1} \dots Q_{ip}$, we translate each sentence Q_{ij} to German Q^*_{ij}, and then translate Q^*_{ij} back to English yielding Q'_{ij} $\forall j \in \{1, 2, \dots p\}$. Further details are provided in Appendix A.

$$f_1(Q_i) = concat(Q'_{i1}, Q'_{i2} \dots Q'_{ip})$$

$\mathbf{f_1(Q_0)}$: *Alex was driving 100 km from New York at a constant speed of 20 km / h. How many hours did it take in total?*

f_2: Same Sentence Inspired by SimCSE [5], we explicitly provide the same sentence as a positive augmentation as the standard dropout masks in the encoder act as a form of augmentation.
$\mathbf{f_2(Q_0)}$: *Alex travelled 100 km from New York at a constant speed of 20 kmph. How many hours did it take him in total?*

f_3: Num2Words Let α be a function that converts any number to its word form. Given a question Q_i, we extract all the numbers $N_i = \{n_{i1}, n_{i2} \dots n_{ik}\}$ from Q_i. For each number $n_{ij} \in N_i$, we generate its word representation $\alpha(n_{ij})$, and replace n_{ij} by $\alpha(n_{ij})$ in Q_i to get $f_3(Q_i)$. This is done because paraphrasing models can replace numbers with their word form, and thus to ensure the scoring model does not consider it as a negative, we explicitly steer it to consider it a positive.
$\mathbf{f_3(Q_0)}$: *Alex travelled one hundred km from New York at a constant speed of twenty kmph. How many hours did it take him in total?*

f_4: UnitExpansion Let υ be a function that converts the abbreviation of a unit into its full form. We detect all the abbreviated units $U_i = \{u_{i1}, u_{i2} \dots u_{ik}\}$ from Q_i (using a predefined vocabulary of units and regular expressions). For each unit $u_{ij} \in U_i$, we generate its expansion $\upsilon(u_{ij})$, and replace u_{ij} by $\upsilon(u_{ij})$ in Q_i. This transformation helps the model to learn the units and their expansions, and consider them as the same when scoring a paraphrase.
$\mathbf{f_4(Q_0)}$: *Alex travelled 100 kilometre from New York at a constant speed of 20 kilometre per hour. How many hours did it take him in total?*

3.3 Negative Augmentations

f_5: Most Important Phrase Deletion The removal of unimportant words like stopwords (the, of, and) from an algebraic question will not perturb the solution or render it impossible to solve.

Thus, to generate hard negatives, we chose the most critical phrase, p_{imp} in any question, deleting which would generate Q_i' such that $\lambda(Q_i, Q_i') = 0$. Let $\Psi : Q \mapsto P$ denote a function which returns the set of k most critical phrases $(p_1, p_2, \dots, p_k)$ in the input Q_i.

$$p_{imp} = \underset{p}{\operatorname{argmin}}(cossim(Q_i, Q_i \backslash p)) \qquad \forall p \in \Psi(Q_i)$$
$$f_5(Q_i) = Q_i \backslash p_{imp}$$

where $cossim$ denotes cosine similarity and $Q_i \backslash p$ denotes the deletion of p from Q_i. Further details are present in Appendix A.
$\mathbf{f_5(Q_0)}$: *Alex travelled 100 km from New York at a constant speed of 20 kmph. How did it take him in total?*

f_6: Last Sentence Deletion When using existing paraphrasing models such as Pegasus, the last few words or even the complete last sentence of the input question got deleted in the generated paraphrase in some cases. Thus, to account for this behaviour, we use this transformation as a negative. More formally, let the input Q_i be tokenized into p sentences $Q_{i1}, Q_{i2} \dots Q_{ip}$ and the sentence Q_{i1} be tokenized into k tokens $Q_{i11}, Q_{i12} \dots Q_{i1k}$. Then,

$$f_6(Q_i) = \begin{cases} concat(Q_{i11}, Q_{i12} \dots Q_{i1(k-3)}) & p = 1 \\ concat(Q_{i1}, Q_{i2} \dots Q_{i(p-1)}) & p > 1 \end{cases}$$

$\mathbf{f_6(Q_0)}$: *Alex travelled 100 km from New York at a constant speed of 20 kmph.*

f_7: Named Entity Replacement Since named entities are an important part of questions, we either replace them with a random one of the same category (from a precompiled list) or with the empty string (deletion). Let $\epsilon : Q \mapsto E$ denote a function which returns a set of all named entities present in the input Q_i, such that $(e_1, e_2, \ldots, e_k) = \epsilon(Q_i)$. We randomly sample w elements $E_i = (e_a, e_b \ldots e_w)$ from $(e_1, e_2, \ldots, e_k)$ and replace/delete the entities. We set $w = rand(1, min(3, k))$ where $rand(a, b)$ represents the random selection of a number from a to b (inclusive). This restricts w from being more than 3, thus increasing the difficulty of the generated negative.
$\mathbf{f_7(Q_0)}$: *Sarah travelled 100 km from at a constant speed of 20 kmph. How many hours did it take him in total?*

f_8: Numerical Entity Deletion Since numbers are critical to algebraic questions, their removal perturbs the solution and helps generate hard negatives. Let $\nu : Q \mapsto N$ represent a function which returns a set of all numbers present in the input Q_i, such that $(n_1, n_2, \ldots, n_k) = \nu(Q_i)$. We randomly sample a subset of numbers N_i from $(n_1, n_2, \ldots, n_k)$, and sample a string s from S = *("some", "a few", "many", "a lot of", "")*. For each number $n_j \in N_i$, we replace it by s in Q_i. We set $|max(N_i)| = 2$. Similar to f_7, this makes it more challenging for the scoring model as we don't necessarily delete all the numbers, thereby generating harder negatives. This allows the model to learn that even the loss of one number renders the resultant output as an invalid paraphrase, thus getting assigned a low score.
$\mathbf{f_8(Q_0)}$: *Alex travelled some km from New York at a constant speed of some kmph. How many hours did it take him in total?*

f_9: Pegasus Pegasus [25] is a transformer-based language model, fine-tuned on PAWS [27] for our purpose. Pegasus consistently gave poor results for paraphrasing algebraic questions, as shown in Figure 1. This provided the impetus for using it to generate hard negatives.
$\mathbf{f_9(Q_0)}$: = *The journey from New York to New Jersey took Alex 100 km at a constant speed.*

f_{10}: UnitReplacement Paraphrasing models sometimes have a tendency to replace units with similar ones (such as *feet* to *inches*). Since this would change the solution to an algebraic question, we defined this transformation to replace a unit with a different one from the same category. We identified 5 categories, C = *[Currency, Length, Time, Weight, Speed]* to which most units appearing in algebraic problems belong. Our transformation was defined such that a unit u_a belonging to a particular category C_i is replaced with a unit u_b, such that $u_b \in C_i$ and $u_a \neq u_b$. For instance, *hours* could get converted to *minutes* or *days*, *grams* could get converted to *kilograms*.

Let C be the set of identified unit categories and $\Upsilon : U \mapsto U$ be a function that takes as input unit $u_a \in C_i$ and returns a different unit $u_b \in C_i$, where $C_i \in C$. Given the input

Q_i containing units $U_i = (u_a, u_b \dots u_n)$, we sample a set of units $U_{is} = \{u_x, \dots u_z\}$ and replace them with $\{\Upsilon(u_i)\ \forall u_i \in U_{is}\}$ to generate $f_{10}(Q_i)$.
$\mathbf{f_{10}(Q_0)}$: *Alex travelled 100 m from New York at a constant speed of 20 kmph. How many hours did it take him in total?*

In the next section, we will detail our approach to training a model to detect the quality of paraphrases and how it can be used to score paraphrases.

3.4 Paraphrase Quality Detection

For detecting the quality of the paraphrases, we use MiniLM [20] as our base encoder (specifically, the version with 12 layers which maps the input sentences into 384-dimensional vectors)[2]. We utilize the implementation from SentenceTransformers [14], where the encoder was trained for semantic similarity tasks using over a billion training pairs and achieved high performance with a fast encoding speed[3].

We train the model using triplet loss. For each question Q_i, let the positive transformation Q_i^+ be denoted by $pos(Q_i)$ and the negative transformation Q_i^- by $neg(Q_i)$ where $pos \in (f_1, \dots f_4)$ and $neg \in (f_5, f_6 \dots f_{10})$. Let the vector representation of any question Q_i when passed through the encoder be denoted as $ENC(Q)$. Then the loss is defined as

$$Loss(Q, Q^+, Q^-) = \sum_i max(0, \alpha - dist(Q_i, Q_i^-) + dist(Q_i, Q_i^+))$$

where α is the margin parameter, $dist(Q_i, Q_i^l) = 1 - cossim(ENC(Q_i), ENC(Q_i^l))$ and $l \in \{+, -\}$. The loss ensures that the model yields vector representations such that the distance between Q_i and Q_i^+ is smaller than the distance between Q_i and Q_i^-.

At inference time, to obtain the paraphrase score of Q_i and Q_i', we use cosine similarity. Let $score : Q \times Q \mapsto [-1, 1]$ denote the scoring function, then for a pair of questions (Q_i, Q_i'):

$$\rho_i, \zeta_i = ENC(Q_i), ENC(Q_i')$$
$$score(Q_i, Q_i') = cossim(\rho_i, \zeta_i) = \frac{\rho_i \cdot \zeta_i}{|\rho_i| \cdot |\zeta_i|}$$

4 Experiments

All the experiments were performed using a Tesla T4 and P100. All models, including the baselines, were trained for 9 epochs with a learning rate of 2e-5 using AdamW as the optimizer with seed 3407. We used a linear scheduler, with 10% of the total steps as warm-up having a weight decay of 0.01.

[2] https://bit.ly/3F2c9vH
[3] https://sbert.net/docs/pretrained_models.html

4.1 Datasets

The datasets used in the experiments are:

AquaRAT [8] (Apache, V2.0) is an algebraic dataset consisting of 30,000 (post-filtering) problems in the training set, 254 problems for validation and 220 problems for testing. After applying the test set operators to yield paraphrases, we get 440 samples for testing with manual labels.

EM_Math is a dataset consisting of mathematics questions for students from grades 6-10 from our partner company ExtraMarks. There are 10,000 questions in the training set and 300 in the test set. After applying the test operators, we get 600 paraphrase pairs.

SAWP (Simple Arithmetic Word Problems) is a dataset that we collected (from the internet) consisting of 200 algebraic problems. We evaluate the proposed methods in a zero-shot setting on this dataset by using the model trained on the AquaRAT dataset. After applying the test set operators, we get 400 paraphrase pairs.

PAWP (Paraphrased Algebraic Word Problems) is a dataset of 400 algebraic word problems collected by us. We requested two academicians from the partnering company (paid fair wages by the company) to manually write paraphrases (both valid and invalid) rather than using our test set operators. We use this dataset for zero-shot evaluation to demonstrate the performance of our model on human-crafted paraphrases.
Our data can also be used as a **seed set** for the task of paraphrase generation for algebraic questions.

4.2 Test Set Generation

For generating the synthetic test set (for AquaRAT, EM_Math and SAWP), we define a different set of operators to generate positive and negative paraphrases to test the ability of our method to generalize to a different data distribution. For any question Q_i in the test set, we generate two paraphrases and manually annotate the question-paraphrase pairs with the help of two annotators. The annotators were instructed to mark valid paraphrases as 1 and the rest as 0. We observed Cohen's Kappa values of **0.79**, **0.84** and **0.70** on AquaRAT, EM_Math and SAWP, respectively, indicating a substantial level of agreement between the annotators.

Operator Details We defined two positive (f_a, f_b) and three negative (f_c, f_d, f_e) test operators. For each question, we randomly chose one operator from each category for generating paraphrases. These functions are:

f_a**: Active-Passive**: We noticed that most algebraic questions are written in the active voice. We used a transformer model for converting them to passive voice[4], followed by a grammar correction model[5]on top of this to ensure grammatical correctness.

f_b**: Corrupted Sentence Reconstruction**: We corrupt an input question by shuffling, deleting and replacing tokens, similar to ROTOM [10] but with additional leniency (Appendix A). We then train a sequence transformation model (t5-base) to reconstruct the original question from the corrupted one, which yields a paraphrase.

[4] https://bit.ly/3FbPIEu
[5] https://bit.ly/3HGOMcQ

f_c**: TF-IDF Replacement**: Instead of the usual replacement of words with low TF-IDF score [22], we replace the words with high TF-IDF scores with random words in the vocabulary. This helps us generate negative paraphrases as it removes the meaningful words in the original question rendering it unsolvable.

f_d**: Random Deletion**: Random deletion is the process of randomly removing some tokens in the input example [21] to generate a paraphrase.

f_e**: T5**: We used T5 [13] fine-tuned on Quora Question Pairs to generate negatives as it was consistently resulting in paraphrases with missing information (Figure 1).

4.3 Baselines

We compare against two SOTA data augmentation methods, UDA and SSMBA. For all the baselines, we use the same encoder (MiniLM) as for our method to maintain consistency across the experiments and enable a fair comparison.

UDA: UDA uses backtranslation and TF-IDF replacement (replacing words having a low score) to generate augmentations for any given input.

SSMBA: SSMBA is a data augmentation technique that uses corruption and reconstruction functions to generate the augmented output. The corruption is performed by masking some tokens in the input and using an encoder (such as BERT [3]) to fill the masked token.

Since the baselines are intended to generate positive paraphrases, we consider other questions in the dataset (in-batch) as negatives to train using the triplet loss. Alongside the direct implementation of UDA and SSMBA, we also compare pseudo-labelled versions of these baselines. The version of baselines without pseudo-labelling is used in all the experiments unless stated with suffix *(with pl)*. The details of pseudo labelling are provided in Appendix B.

4.4 Metrics

Our main goal is to ensure the separation of valid and invalid paraphrases by a wide margin. This allows for extrapolation to unseen and unlabelled data (the distribution of scores for positive and negative paraphrases is unknown, thus threshold can be set to the standard 0.5 or a nearby value due to wider margins). It allows for the score to be used as a selection metric using maximization strategies like Simulated Annealing [9] or as reward using Reinforcement Learning [16,24] to steer generation. To this end, along with Precision, Recall, and F1 (both macro and weighted), we compute the separation between the mean positive and mean negative scores. More formally, let the score of all (Q_i, Q_i^+) pairs be denoted by $score(Q, Q^+)$ and the score of all (Q_i, Q_i^-) pairs be denoted by $score(Q, Q^-)$ where $\lambda(Q_i, Q_i^+) = 1$ and $\lambda(Q_i, Q_i^-) = 0$. Then,

$$\mu^s\ (separation) = \mu^+ - \mu^-$$
$$\mu^l = E[score(Q, Q^l)]\ \forall\ l \in \{+, -\}$$

4.5 Test Set Details

The number of positive and negative pairs are (139, 301) in AquaRAT, (223, 377) in EM, (130, 270) in SAWP and (199, 201) in PAWP. The details of the success of test set

operators are shown in the form of confusion matrices in Figure 6 (supplementary). The average precision, recall and accuracy of the operators across the datasets are 0.4, 0.59 and 0.56. The low precision is due to the inability of positive operators to generate valid paraphrases consistently, as the task of effectively paraphrasing algebraic questions is challenging. This further demonstrates the usefulness of a method like ParaQD that can be effectively used to distinguish the paraphrases as an objective to guide paraphrasing models (4.4).

Table 1: Precision, Recall, F1 and Separation across all methods and datasets.

Dataset	**Method**	**Macro**			**Weighted**			μ^+	μ^-	μ^s
		P	R	F1	P	R	F1			
AquaRAT	Pretrained	0.658	0.502	0.569	0.784	0.318	0.453	0.977	0.897	0.080
	UDA	0.661	0.512	0.577	0.786	0.332	0.467	0.995	0.966	0.029
	UDA (w pl)	0.659	0.507	0.573	0.785	0.325	0.460	0.996	0.973	0.023
	SSMBA	0.645	0.554	0.596	0.757	0.395	0.520	0.965	0.829	0.137
	SSMBA (w pl)	0.663	0.522	0.584	0.787	0.345	0.480	0.997	0.928	0.069
	ParaQD (ours)	0.678	0.695	**0.687**	0.762	0.625	**0.687**	0.770	-0.010	**0.780**
EM_Math	Pretrained	0.694	0.534	0.604	0.773	0.415	0.540	0.955	0.796	0.158
	UDA	0.648	0.523	0.579	0.716	0.403	0.516	0.991	0.912	0.079
	UDA (w pl)	0.683	0.587	0.631	0.751	0.485	0.589	0.963	0.751	0.213
	SSMBA	0.615	0.564	0.588	0.669	0.470	0.552	0.871	0.729	0.142
	SSMBA (w pl)	0.655	0.586	0.619	0.716	0.492	0.583	0.937	0.629	0.308
	ParaQD (ours)	0.665	0.665	**0.665**	0.708	0.622	**0.662**	0.667	0.012	**0.655**
SAWP	Pretrained	0.162	0.500	0.245	0.106	0.325	0.159	0.964	0.896	0.068
	UDA	0.557	0.514	0.535	0.636	0.358	0.458	0.958	0.912	0.046
	UDA (w pl)	0.667	0.519	0.583	0.783	0.350	0.484	0.990	0.929	0.061
	SSMBA	0.662	0.594	0.626	0.763	0.460	0.574	0.929	0.758	0.172
	SSMBA (w pl)	0.649	0.537	0.588	0.757	0.378	0.504	0.978	0.864	0.115
	ParaQD (ours)	0.636	0.645	**0.640**	0.709	0.582	**0.640**	0.656	0.068	**0.589**
PAWP	Pretrained	0.749	0.502	0.602	0.751	0.500	0.600	0.948	0.905	0.042
	UDA	0.558	0.507	0.532	0.559	0.505	0.530	0.960	0.948	0.012
	UDA (w pl)	0.668	0.510	0.578	0.669	0.507	0.577	0.988	0.961	0.026
	SSMBA	0.536	0.512	0.524	0.536	0.510	0.523	0.874	0.853	0.021
	SSMBA (w pl)	0.551	0.510	0.530	0.552	0.507	0.529	0.939	0.913	0.026
	ParaQD (ours)	0.703	0.669	**0.685**	0.703	0.668	**0.685**	0.749	0.076	**0.673**

5 Results and Analysis

The performance comparison and results of all methods are shown in Table 1. Across all datasets, for the measures macro-F1, weighted-F1 and separation, ParaQD outperforms all the baselines by a significant margin. For instance, the margin of separation in

Table 2: Summarizing the top-2 positive (Op+) and negative (Op-) operators across datasets.

Dataset	Op+		Op-	
	1	**2**	**1**	**2**
AquaRAT	f_3	f_1	f_9	f_5
EM_Math	f_4	f_1	f_9	f_8
SAWP	f_2	f_1	f_9	f_6
PAWP	f_1	f_2	f_{10}	f_9

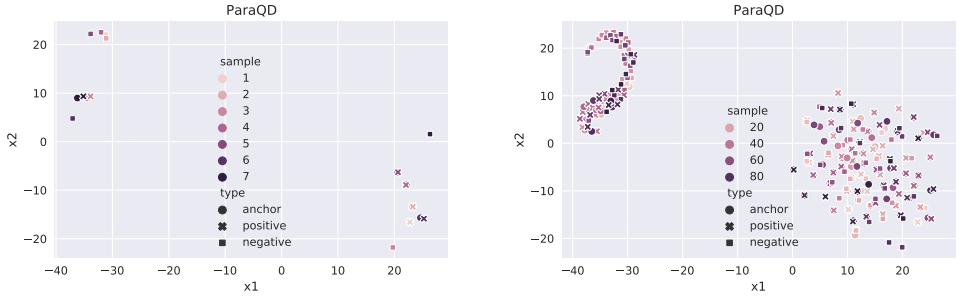
ParaQD is 5.69 times the best baseline SSMBA. To calculate the precision, recall and F1 measures, we threshold the obtained scores at the standard $\tau = 0.5$. Since this is a self-supervised method, there are no human-annotated labels available for the training and validation set. This means that the distribution of scores is unknown, and thus, the threshold can not be tuned on the validation set.

5.1 Performance

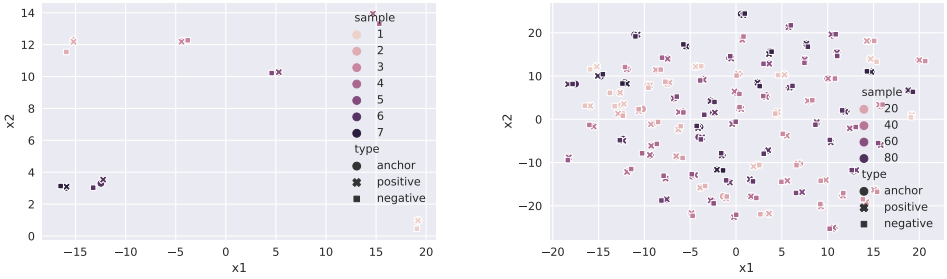
Our primary metric is separation (for reasons detailed in 4.4). Weighted F1 is more representative of the actual performance than macro F1 due to imbalanced data (4.5), and the results are discussed further.

AquaRAT and EM_Math : ParaQD outperforms the best-performing baseline by 32.1% weighted F1 on AquaRAT and 12.4% weighted F1 on EM_Math. The separation achieved by ParaQD on AquaRAT is 0.78 while the best performing baseline achieves 0.137, and on EM_Math, our method achieves a separation of 0.655 while the best performing baseline achieves a separation of 0.308.
SAWP: Evaluating zero-shot performance on SAWP, ParaQD outperforms the best performing baseline by 11.5% weighted F1 and achieves a separation of 0.589 as compared to the 0.172 achieved by the best baseline. This demonstrates the ability of our method to perform well even on zero-shot settings, as the distribution of this dataset is not identical to the ones that the model was trained on.
PAWP: Our method beats the best performing baseline by 14% weighted F1 on the manually created dataset PAWP, which also consists of a zero-shot setting. It demonstrates an impressive separation of 0.673, while the best performing baseline only has a separation of 0.042. This is practically applicable as it highlights that our method can also be used to evaluate paraphrases that have been manually curated by academicians (especially on online learning platforms) instead of only on automatically generated paraphrases.

To analyze and gain a deeper insight into these results, we plotted the confusion matrices (Figure 7), and observed that ParaQD is able to consistently recognize invalid paraphrases to a greater extent than the baselines as it learns to *estimate the true distribution of negative samples* more effectively through our novel data augmentations.



(a) ParaQD



(b) Pretrained

Fig. 2: Embedding plots on AquaRAT. Figure 5 in supplementary covers remaining plots.

5.2 Embedding Plots

To qualitatively evaluate ParaQD, we use t-SNE to project the embeddings into a two-dimensional space (Appendix C) as seen in Figure 2. We observe that the separation between anchors and negatives of triplets is minimal for the baselines, while ParaQD is able to separate them more effectively. Perhaps a more interesting insight from Figure 2a is that our method is able to cluster negatives together, which is not explicitly optimized by triplet loss as it does not account for inter-sample interaction. We note that our negative operators (with the possible exception of f_7 and f_{10}) are designed to generate unsolvable problems serving as good negatives for training the scoring model (ParaQD).

5.3 Operator Ablations

To measure the impact of all operators, we trained the model after removing each operator one by one. The summary of the results is in Table 2 (complete in Table 4 (supplementary)). We note that f_1 (defined in Section 3.2) seems to be the most consistently important operator amongst the positives, while f_9 (defined in Section 3.3) is the most consistently important operator amongst the negatives. One possible reason for the success of f1 could be that it is the only positive operator that actually changes the words and sentence structure, which is replicated by our test operators and by the human-generated paraphrases.

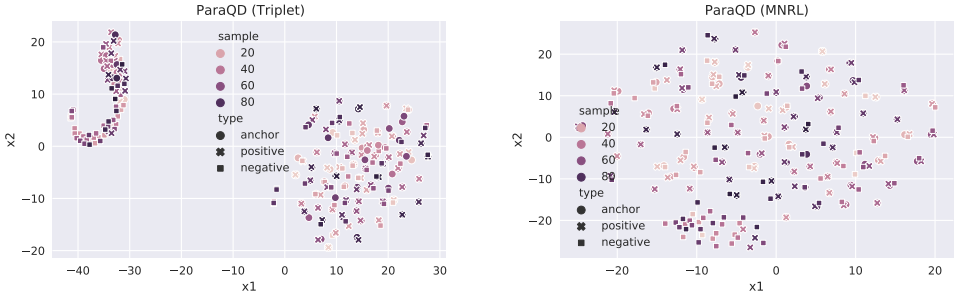
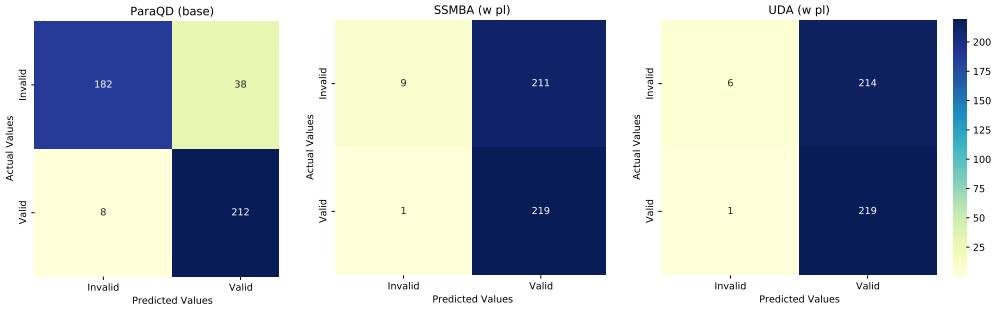


Fig. 3: Embedding plots for different loss functions on AquaRAT



(a) ParaQD (b) SSMBA with PL (c) UDA w pl

(d) Pretrained (e) SSMBA (f) UDA

Fig. 4: Confusion matrices for all methods on AquaRAT. Others can be found in supplementary (Figures 9, 10)

Also, for the synthetically generated test sets (for AquaRAT, EM_Math and SAWP), since f_9 is a transformer model, it might generate paraphrases with a closer distribution (especially to f_e), but it also performs well on the human crafted paraphrases on PAWP. f_4 performs really well on EM_Math as the dataset involves more mathematical symbols,

and thus the distribution of the data is such that technical operators (like f_4 and f_8) would have a more profound impact on the dataset.

The results also show that operator importance depends on the data, as certain data distributions might possess patterns that are more suitable to a certain set of operators. We also note that all operators are critical as removing any operator reduces performance for multiple datasets, thus demonstrating the usefulness of the combination of augmentations as a general framework.

Table 3: Analysis of model scores for different examples

Original	Paraphrase	Label	ParaQD
A bag of cat food weighs 7 pounds and 4 ounces. How much does the bag weigh in ounces?	A bag of cat food weighs 7 pounds and ounces. How much does the bag in ounces?	0	-0.922
A cart of 20 apples is distributed among 10 students. How much apple does each student get?	20 hats in a cart are equally distributed among 10 students. How much apple does each student get?	0	-0.999
A cart of 20 apples is distributed among 10 students. How much apple does each student get?	20 hats in a cart are equally distributed among 10 students. How many hats does each student get?	1	0.999
John walked 200 kilometres. How long did he walk in terms of metres?	john walked 200 centimetres. How long did he walk in terms of metres?	0	-0.999
John walked 200 kilometres. How long did he walk in terms of metres?	john walked 200 km. How long did he walk in terms of metres?	1	0.999

5.4 Effects of Loss Functions, Encoder and Seed

We analyzed the impact of the loss function by performing an ablation with Multiple Negative Ranking Loss (MNRL) (Appendix D) when training ParaQD. Since MNRL considers inter-sample separation, rather than explicitly distancing the generated hard negative, it is not able to provide a high margin of separation between the positives and negatives (μ^s = 0.416) as high as the triplet loss (μ^s = 0.78) but does result in a minor increase in the F1 scores. This can be observed in Figure 3 and Table 5 (supplementary). We also analyzed the effects of the encoder and seed across methods on AquaRAT (Table 6, 8; detailed analysis in Appendix F) to demonstrate the robustness of our approach. We observe that we outperform the baselines on all the metrics for three encoders we experimented with, namely MiniLM (12 layers), MiniLM (6 layers) and MPNet for different seeds.

5.5 Error Analysis and Limitations

Does the model check for the preservation of numerical quantities?: From example 1 in Table 3, we observe that the number **4** is missing in the paraphrase rendering the

problem unsolvable. Our model outputs a negative score, indicating it is a wrong paraphrase. This general phenomenon is observed in our reported results.
Does the model check for entity consistency?: We also observe that our model checks for entity consistency. For instance, in example 2, we observe that the paraphraser replaces *apples* with *hats* in the first sentence of the question. However, it fails to replace it in the second part of the question retaining the term *apple* which leads to a low score from ParaQD due to inconsistency. We observe from example 3 that when entity replacement is consistent throughout the question (*apple* replaced by *hats*, the model outputs a high score indicating it is a valid paraphrase.
Does the model detect changes in units?: Changing the units in algebraic word problems sometimes may render the question unsolvable or change the existing solution requiring manual intervention. For instance, from example 4 in Table 3, we observe that the unit *kilometres* is changed to *centimetres* in the paraphrase, which would change the equation to solve the question and by consequence the existing solution. Since we prefer solution preserving transformation of the question, ParaQD assigns a low score to this paraphrase. However, when *kilometres* is contracted to *km* in example 5, we observe that our model correctly outputs a high score.
Does the model make errors under certain scenarios?: We also analyzed the errors made by the model. We noted that samples that have valid changes in numbers are not always scored properly by the model. Thus, a limitation of this approach is that it is not robust to changes in numbers that preserve the solution. For instance, if we change the numbers 6 and 4 to 2 and 8 in Figure 1, the underlying equation and answer would still be preserved. But ParaQD may not output a high score for the same. We must note, however, that generating these types of paraphrases is something that is beyond the ability of general paraphrasing models. As a potential solution (in the future), we propose that numerical changes can be handled through feedback from an automatic word problem solver.

6 Conclusion

In this paper, we formulated the novel task of scoring paraphrases for algebraic questions and proposed a self-supervised method to accomplish this. We demonstrated that the model learns valuable representations that separate positive and negative paraphrases better than existing text augmentation methods and provided a detailed analysis of various components. In the future, we plan to use the scoring model as an objective to steer language models for paraphrasing algebraic word problems and also investigate the usage of representations learned by our method for the novel task of solvable problem detection.

7 Acknowledgements

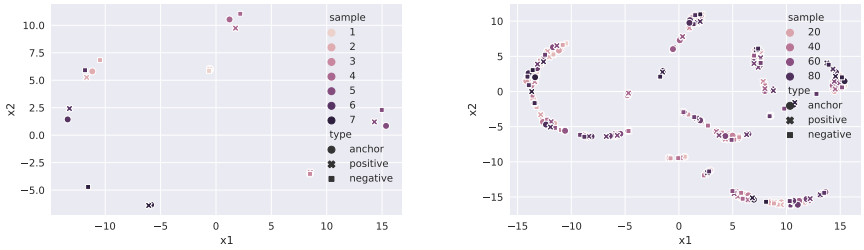
We would sincerely like to thank Extramarks Education India Pvt. Ltd., SERB, FICCI (PM fellowship) and TiH Anubhuti (IIITD) for supporting this work.

References

1. Asai, A., Hajishirzi, H.: Logic-guided data augmentation and regularization for consistent question answering (2020)
2. Bougouin, A., Boudin, F., Daille, B.: TopicRank: Graph-based topic ranking for keyphrase extraction. In: Proceedings of the Sixth International Joint Conference on Natural Language Processing. pp. 543–551. Asian Federation of Natural Language Processing, Nagoya, Japan (Oct 2013), https://aclanthology.org/I13-1062
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018)
4. Egonmwan, E., Chali, Y.: Transformer and seq2seq model for paraphrase generation. In: Proceedings of the 3rd Workshop on Neural Generation and Translation. pp. 249–255. Association for Computational Linguistics, Hong Kong (Nov 2019). https://doi.org/10.18653/v1/D19-5627, https://aclanthology.org/D19-5627
5. Gao, T., Yao, X., Chen, D.: Simcse: Simple contrastive learning of sentence embeddings (2021)
6. Henderson, M., Al-Rfou, R., Strope, B., hsuan Sung, Y., Lukacs, L., Guo, R., Kumar, S., Miklos, B., Kurzweil, R.: Efficient natural language response suggestion for smart reply (2017)
7. Kang, D., Khot, T., Sabharwal, A., Hovy, E.: Adventure: Adversarial training for textual entailment with knowledge-guided examples (2018)
8. Ling, W., Yogatama, D., Dyer, C., Blunsom, P.: Program induction by rationale generation : Learning to solve and explain algebraic word problems (2017)
9. Liu, X., Mou, L., Meng, F., Zhou, H., Zhou, J., Song, S.: Unsupervised paraphrasing by simulated annealing. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 302–312. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.28, https://aclanthology.org/2020.acl-main.28
10. Miao, Z., Li, Y., Wang, X.: Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond, p. 1303–1316. Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3448016.3457258
11. Ng, N., Cho, K., Ghassemi, M.: Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness (2020)
12. Ng, N., Yee, K., Baevski, A., Ott, M., Auli, M., Edunov, S.: Facebook fair's wmt19 news translation task submission (2019)
13. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer (2020)
14. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China (Nov 2019)
15. Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y.: Mpnet: Masked and permuted pre-training for language understanding (2020)
16. Stiennon, N., Ouyang, L., Wu, J., Ziegler, D.M., Lowe, R., Voss, C., Radford, A., Amodei, D., Christiano, P.: Learning to summarize from human feedback (2020)
17. Thakur, N., Reimers, N., Daxenberger, J., Gurevych, I.: Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks (2021)

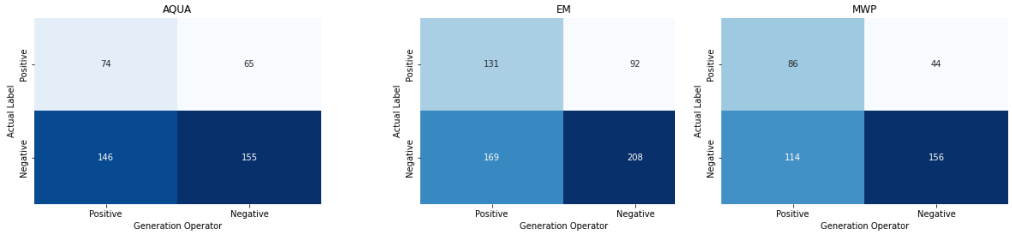
18. Vijayakumar, A.K., Cogswell, M., Selvaraju, R.R., Sun, Q., Lee, S., Crandall, D., Batra, D.: Diverse beam search: Decoding diverse solutions from neural sequence models (2018)
19. Wang, K., Reimers, N., Gurevych, I.: Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning (2021)
20. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers (2020)
21. Wei, J., Zou, K.: Eda: Easy data augmentation techniques for boosting performance on text classification tasks (2019)
22. Xie, Q., Dai, Z., Hovy, E., Luong, M.T., Le, Q.V.: Unsupervised data augmentation for consistency training (2020)
23. Xie, Z., Wang, S.I., Li, J., Lévy, D., Nie, A., Jurafsky, D., Ng, A.Y.: Data noising as smoothing in neural network language models (2017)
24. Yasui, G., Tsuruoka, Y., Nagata, M.: Using semantic similarity as reward for reinforcement learning in sentence generation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop. pp. 400–406. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-2056, https://aclanthology.org/P19-2056
25. Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: Pegasus: Pre-training with extracted gap-sentences for abstractive summarization (2019)
26. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification (2016)
27. Zhang, Y., Baldridge, J., He, L.: Paws: Paraphrase adversaries from word scrambling (2019)

(a) UDA



(b) SSMBA

Fig. 5: Further Embedding plots on AquaRAT



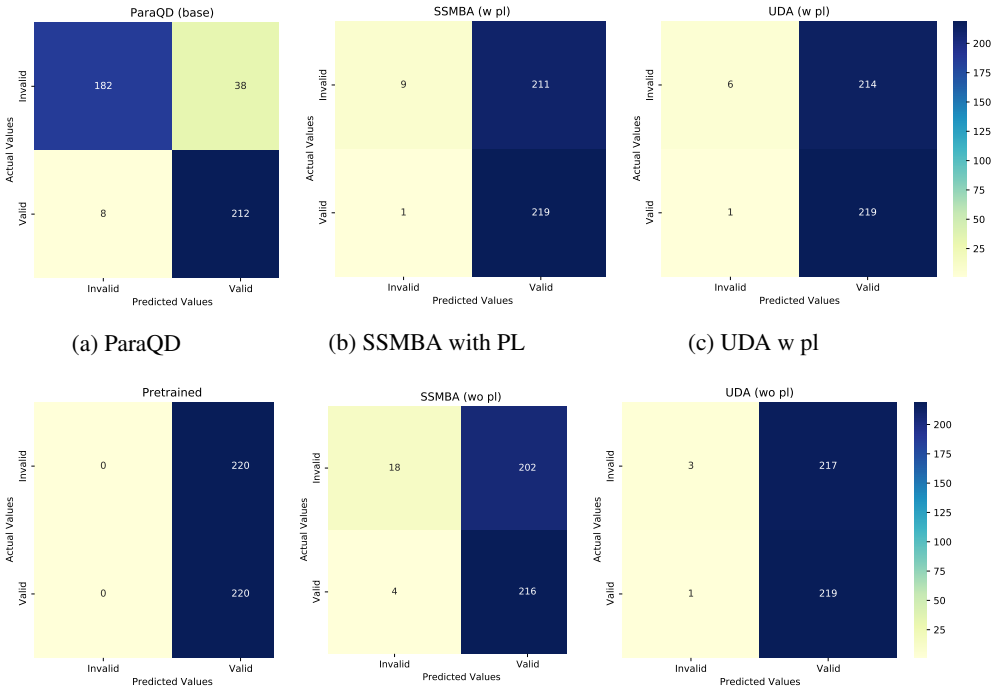
(a) AquaRAT (b) EM (c) SAWP

Fig. 6: Statistics for Test Operators for AquaRAT, EM and SAWP.

A Augmentation: Further Details

A.1 f_1 : Backtranslation

We used WMT'19 FSMT [12] *en-de* and *de-en* translation models, with language A being $English$ and B being $German$. We used diverse beam search [18] for decoding



(a) ParaQD (b) SSMBA with PL (c) UDA w pl

(d) Pretrained (e) SSMBA (f) UDA

Fig. 7: Confusion matrices for all methods on AquaRAT

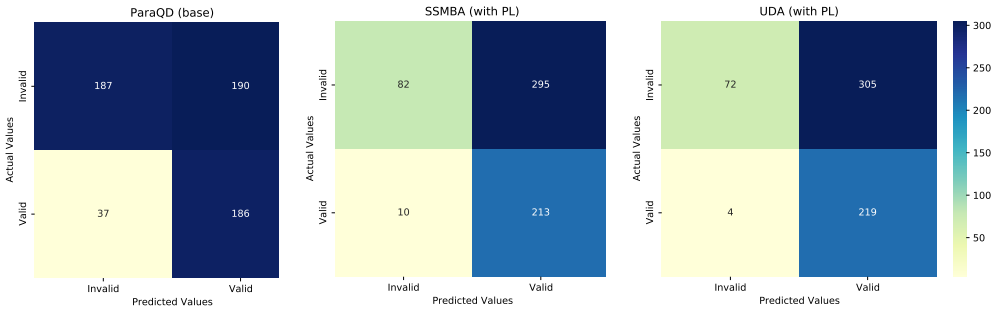
in the reverse translation step to introduce diversity and chose the candidate paraphrase with the maximum Levenshtein distance. This ensures that the model learns to give a high score for diverse paraphrases that retain critical information.

A.2 f_5: Most Important Phrase Deletion

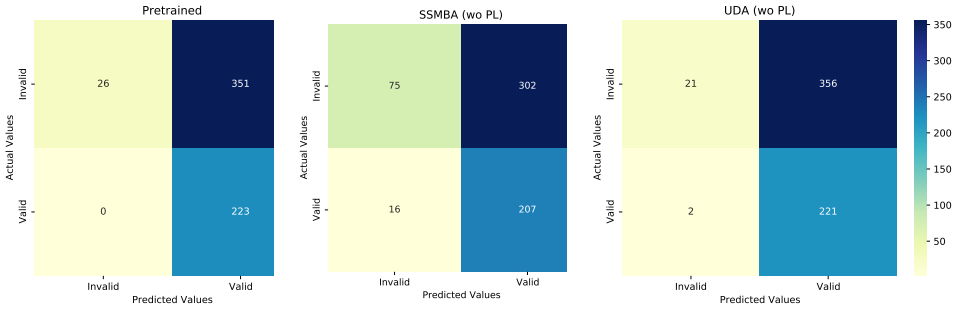
To model Ψ, we use TopicRank [2]. The methodology to select the most critical phrase is inspired by TSDAE [19].

A.3 f_b: Corrupted Sentence Reconstruction

When corrupting the input sentence, we preserve the numbers, units and the last three tokens. This is done because if we corrupt the numbers or the units, the model cannot accurately reconstruct them and will replace them with random numbers and units. We preserve the last three tokens because corrupting them might lead the model to change the question as the last three tokens in a word problem are generally indicative of the question.



(a) ParaQD (b) SSMBA with pl (c) UDA with pl



(d) Pretrained (e) SSMBA (f) UDA

Fig. 8: Confusion matrices for all methods on EM_Math.

B Baselines: Pseudo Labelling

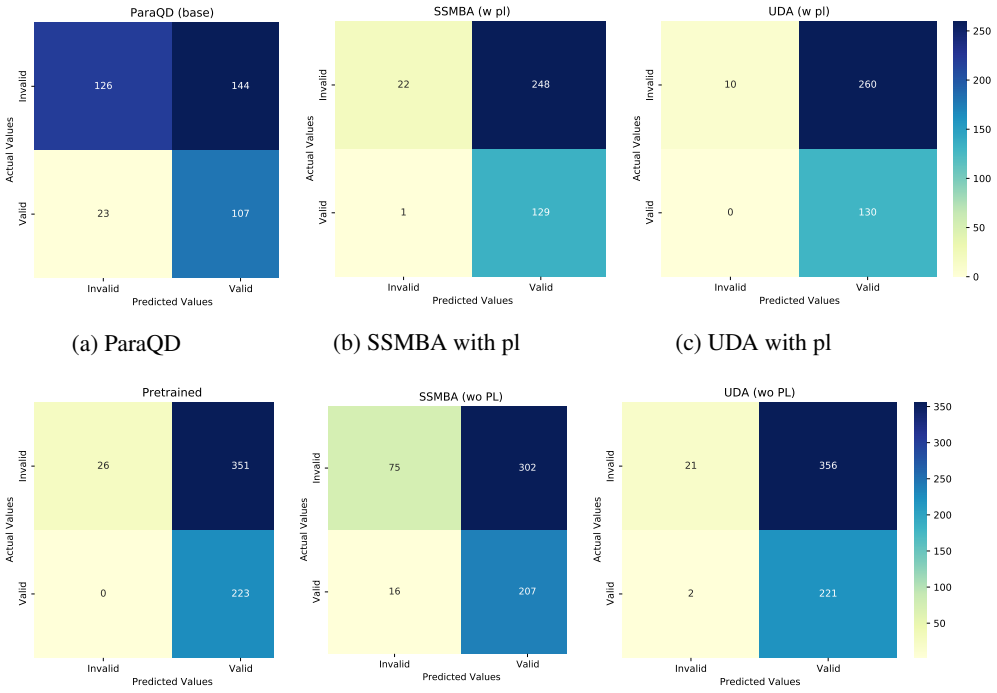
We use the same pretrained encoder (MiniLM) to first pseudo-label the samples (without being trained) and then train it using the pseudo-labelled samples. More formally, given an input Q_i and a paraphrase Q'_i, we use the encoder to determine whether Q'_i is a positive or negative paraphrase of Q_i as follows:

$$\rho_i, \zeta_i = ENC(Q_i), ENC(Q'_i)$$

$$\lambda(Q_i, Q'_i) = \begin{cases} 1 & if\ cossim(\rho_i, \zeta_i) > \iota \\ 0 & if\ cossim(\rho_i, \zeta_i) \leq \iota \end{cases}$$

where ι is the threshold for the cosine similarity, which we set to 0.8.

We observe that on AquaRAT, the performance decreases for both UDA and SSMBA due to pseudo labelling, while it increases on EM_Math dataset. This can be due to the much higher percentage of pseudo-labelled negative samples for EM_Math as shown in Table 9, thus providing more information about detecting invalid paraphrases as seen in Figure 8.



(a) ParaQD (b) SSMBA with pl (c) UDA with pl

(d) Pretrained (e) SSMBA (f) UDA

Fig. 9: Confusion matrices for all methods on SAWP.

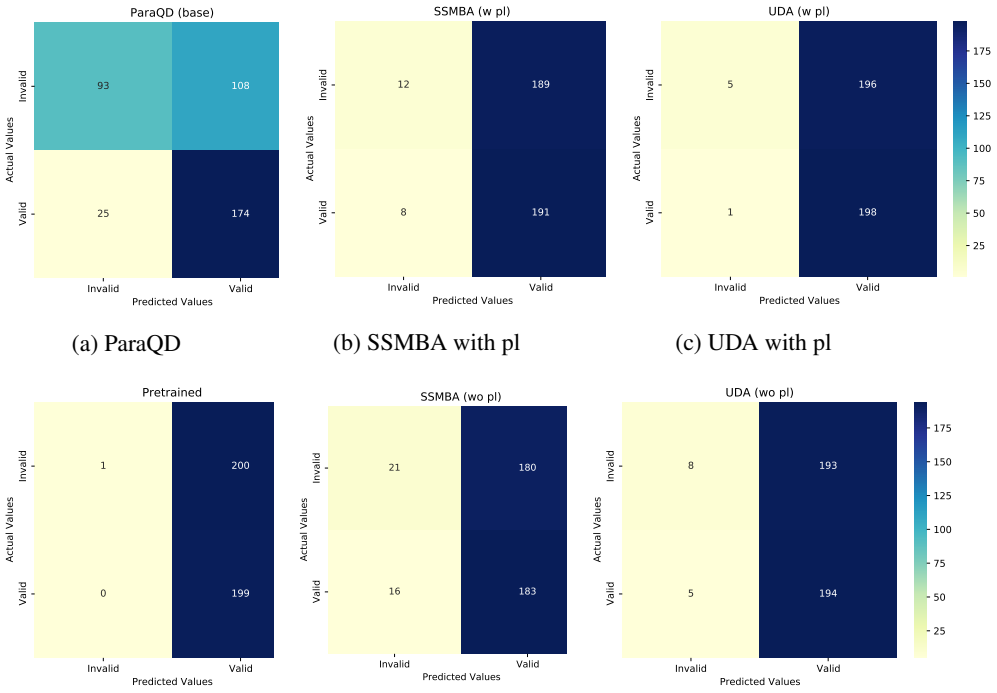
C Embedding Plots

We plotted the embeddings across triplets in the test set to observe the separation margin. The colour represents a triplet, while the symbol represents which component of the triplet it is (anchor, positive, negative). The left embedding plot is for 7 randomly chosen triplets in the data (to closely visualize the distances), while the one on the right is for all triplets (93).

D Loss Functions

Multiple Negatives Ranking Loss [6]is a loss function, which, for anchor a_i in the triplet (a_i, p_i, n_i) considers p_i as a positive sample and all p_j and n_k in the batch (such that $j! = i$) as negatives. It works by maximizing the log-likelihood of the softmax scores. The equation is similar to the one in [6].

Note 1. `https://www.sbert.net/docs/package_reference/losses.html#multiplenegativesrankingloss`



(a) ParaQD (b) SSMBA with pl (c) UDA with pl

(d) Pretrained (e) SSMBA (f) UDA

Fig. 10: Confusion matrices for all methods on PAWP

E Performance on test set with training operators

We also evaluated our method by generating the test set of AquaRAT using training operators. The results are present in Table 7. ParaQD outperforms the baselines by a significant margin across all metrics. However, this test set is not representative of real data distribution as it is suited for our method. Thus, we report the results only for the sake of completion and they should not be taken as representative.

F Encoder Ablations and Seed Optimization

The results of varying the encoder are shown in Table 8. We vary the encoder and experiment with MiniLM (12 layers), MiniLM (6 layers), and MPNet [15]. We choose the encoders considering different metrics such as average performance on semantic search and encoding speed ($\#\ of sentences/sec$)[7]. For instance MPNet can encode about *2500* sentences/sec whereas MiniLM (12 layers) (all-minilm-L12-v1) can encode about 7500 sentences/sec and MiniLM (6 layers) (all-minilm-L6-v2) can encode about

[7] The metrics are obtained from `https://www.sbert.net/docs/pretrained_models.html`

14200 sentences/sec. Also the average performance on semantic search benchmarks is of order $all - minilm - L12 - v1 > all - minilm - L6 - v2 >$ MPNet. In our setting, from table 8 we can observe that MiniLM (12 layers) surpasses the other encoders as measured by the separation metric (μ^s). However, when we observe Macro and weighted F1 scores, MPNet surpasses the other two encoders. Since we are more concerned about how positive and negative paraphrases are separated in the vector space, we choose MiniLM (12 layers) (all-minilm-L12-v1) for all our main experiments, as shown in Table 1. We can also observe that the proposed method (ParaQD) outperforms all other baselines. This demonstrates the robustness of the proposed augmentation method and shows the performance gain when compared to other baselines is invariant to changes in encoders. We also vary the seed values to check for the robustness of our method. We compare the random seed value (3407) with the seed optimization method proposed in the Augmented SBERT paper [17]. For seed optimization, we search for the best seed in the range [0-4] as recommended in the original work by training 20% of the data and comparing the results on the validation set. We then select the best performing seed and train using that particular seed. The results of the experiments are shown in Table 6. We observe that the best seed obtained through seed optimization and the seed value of 3407 nearly yield similar performance. We also observe that the proposed method outperforms the baselines demonstrating the robustness of the proposed method to seed randomization.

Table 4: An ablative analysis of each operator across all datasets. Here, each operator f_i represents the results when we train after removing that operator. The numbers in bold represent the lowest scores for positive operators ($f_1, \ldots f_4$) and negative operators ($f_5, \ldots f_{10}$) each, thereby demonstrating the impact of that operator.

Dataset	**Op**	**Macro**			**Weighted**			μ^+	μ^-	μ^s
		P	R	F1	P	R	F1			
AquaRAT	f_1	0.667	0.681	0.674	0.749	0.611	0.673	0.742	0.024	0.718
	f_2	0.682	0.694	0.688	0.769	0.616	0.684	0.813	0.047	0.766
	f_3	0.663	0.669	**0.666**	0.75	0.586	**0.658**	0.783	0.115	**0.668**
	f_4	0.679	0.686	0.682	0.768	0.602	0.675	0.827	0.084	0.744
	f_5	0.664	0.67	0.667	0.751	0.589	0.66	0.791	0.119	0.672
	f_6	0.667	0.669	0.668	0.756	0.582	0.658	0.813	0.138	0.675
	f_7	0.667	0.678	0.673	0.753	0.602	0.669	0.771	0.069	0.702
	f_8	0.671	0.679	0.675	0.758	0.598	0.668	0.796	0.09	0.706
	f_9	0.653	0.657	**0.655**	0.74	0.573	**0.646**	0.77	0.145	**0.625**
	f_{10}	0.678	0.687	0.683	0.766	0.607	0.677	0.813	0.078	0.735
EM_Math	f_1	0.635	0.644	0.64	0.67	0.627	0.648	0.425	-0.144	0.569
	f_2	0.648	0.651	0.65	0.689	0.613	0.649	0.598	-0.005	0.603
	f_3	0.666	0.669	0.667	0.709	0.628	0.666	0.661	-0.017	0.678
	f_4	0.638	0.635	**0.636**	0.681	0.588	**0.631**	0.633	0.096	**0.538**
	f_5	0.653	0.653	0.653	0.697	0.608	0.65	0.651	0.042	0.609
	f_6	0.657	0.652	0.655	0.703	0.602	0.648	0.696	0.088	0.608
	f_7	0.66	0.658	0.659	0.705	0.612	0.655	0.681	0.048	0.633
	f_8	0.646	0.648	**0.647**	0.688	0.608	0.646	0.606	0.018	**0.588**
	f_9	0.656	0.649	0.652	0.702	0.597	**0.645**	0.704	0.108	0.596
	f_{10}	0.672	0.674	0.673	0.716	0.632	0.671	0.677	-0.012	0.689
SAWP	f_1	0.618	0.627	0.623	0.688	0.572	0.625	0.572	0.06	0.512
	f_2	0.617	0.621	**0.619**	0.69	0.552	**0.614**	0.631	0.152	**0.479**
	f_3	0.624	0.63	0.627	0.697	0.565	0.624	0.632	0.114	0.518
	f_4	0.646	0.648	0.647	0.725	0.57	0.638	0.739	0.16	0.579
	f_5	0.648	0.644	0.646	0.729	0.56	0.633	0.777	0.205	0.572
	f_6	0.629	0.632	0.631	0.705	0.56	0.624	0.679	0.148	0.53
	f_7	0.63	0.637	0.634	0.704	0.572	0.632	0.649	0.096	0.553
	f_8	0.643	0.64	0.642	0.723	0.558	0.63	0.754	0.195	0.559
	f_9	0.626	0.622	**0.624**	0.705	0.538	**0.61**	0.723	0.245	**0.478**
	f_{10}	0.645	0.642	0.643	0.725	0.56	0.632	0.763	0.2	0.562
PAWP	f_1	0.644	0.623	**0.634**	0.645	0.622	**0.633**	0.64	0.148	**0.492**
	f_2	0.68	0.636	0.658	0.681	0.635	0.657	0.769	0.226	0.543
	f_3	0.712	0.659	0.684	0.712	0.658	0.684	0.819	0.192	0.627
	f_4	0.714	0.654	0.683	0.714	0.652	0.682	0.847	0.224	0.623
	f_5	0.694	0.644	0.668	0.695	0.642	0.668	0.8	0.229	0.57
	f_6	0.705	0.646	0.674	0.706	0.645	0.674	0.829	0.244	0.585
	f_7	0.698	0.651	0.674	0.698	0.65	0.673	0.795	0.185	0.61
	f_8	0.702	0.629	0.663	0.702	0.628	0.663	0.859	0.343	0.516
	f_9	0.663	0.619	0.64	0.663	0.618	0.64	0.759	0.284	0.475
	f_{10}	0.655	0.587	**0.619**	0.656	0.585	**0.618**	0.845	0.493	**0.352**

Table 5: Effect of the Loss Function for ParaQD (on AquaRAT)

Loss	Macro			Weighted			μ^+	μ^-	μ^s
	P	**R**	**F1**	**P**	**R**	**F1**			
Triplet	0.678	0.695	0.687	0.762	0.625	0.687	0.77	-0.01	**0.78**
MultipleNegativeRankingLoss	0.708	0.716	**0.712**	0.801	0.627	**0.704**	0.89	0.474	0.416

Table 6: An ablative analysis of all methods for different seeds on AquaRAT.

Seed	Method	Macro			Weighted			μ^+	μ^-	μ^s
		P	**R**	**F1**	**P**	**R**	**F1**			
3407	ParaQD	0.678	0.695	**0.687**	0.762	0.625	**0.687**	0.77	-0.01	**0.78**
	UDA	0.661	0.512	0.577	0.786	0.332	0.467	0.995	0.966	0.029
	SSMBA	0.645	0.554	0.596	0.757	0.395	0.52	0.965	0.829	0.137
Seed Search	ParaQD	0.684	0.694	**0.689**	0.772	0.614	**0.684**	0.828	0.055	**0.772**
	UDA	0.659	0.503	0.571	0.784	0.32	0.455	0.998	0.985	0.013
	SSMBA	0.634	0.552	0.59	0.742	0.395	0.516	0.957	0.833	0.124

Table 7: Performance of all methods on the test set of AquaRAT created using train operators.

Method	Macro			Weighted			μ^+	μ^-	μ^s
	P	**R**	**F1**	**P**	**R**	**F1**			
Pretrained	0.25	0.5	0.333	0.25	0.5	0.333	0.966	0.92	0.046
UDA	0.626	0.505	0.559	0.626	0.505	0.559	0.995	0.973	0.022
UDA (w pl)	0.681	0.511	0.584	0.681	0.511	0.584	0.99	0.965	0.025
SSMBA	0.667	0.532	0.592	0.667	0.532	0.592	0.965	0.871	0.094
SSMBA (w pl)	0.705	0.518	0.597	0.705	0.518	0.597	0.987	0.927	0.06
ParaQD (ours)	0.903	0.895	**0.899**	0.903	0.895	**0.899**	0.927	-0.656	**1.583**

Table 8: An ablative analysis of all methods for 3 different encoders on AquaRAT. We observe that regardless of the encoder used, we outperform the baselines on all metrics.

Encoder	Method	Macro			Weighted			μ^+	μ^-	μ^s
		P	**R**	**F1**	**P**	**R**	**F1**			
all-minilm-L12-v1 (base)	ParaQD	0.678	0.695	**0.687**	0.762	0.625	**0.687**	0.77	-0.01	**0.78**
	UDA	0.661	0.512	0.577	0.786	0.332	0.467	0.995	0.966	0.029
	SSMBA	0.645	0.554	0.596	0.757	0.395	0.52	0.965	0.829	0.137
MPNet	ParaQD	0.703	0.726	**0.714**	0.785	0.659	**0.717**	0.858	0.201	**0.656**
	UDA	0.659	0.503	0.571	0.784	0.32	0.455	0.99	0.953	0.037
	SSMBA	0.66	0.508	0.574	0.785	0.327	0.462	0.985	0.94	0.045
all-minilm-L6-v2	ParaQD	0.671	0.679	**0.675**	0.758	0.598	**0.668**	0.799	0.083	**0.716**
	UDA	0.659	0.503	0.571	0.784	0.32	0.455	0.994	0.979	0.015
	SSMBA	0.661	0.513	0.578	0.786	0.334	0.469	0.992	0.941	0.051

Table 9: Pseudo Labelling Statistics. Positive% represents the percentage of total samples pseudo-labelled as positive, while Negative% represents the percentage of total samples pseudo-labelled as negatives.

Dataset	Method	Positive %	Negative %
AquaRAT	UDA (w pl)	87.29	12.71
	SSMBA (w pl)	75.69	24.31
EM_Math	UDA (w pl)	72.16	27.84
	SSMBA (w pl)	55.09	44.91