# SYMGraph: A Neural Symbolic Approach with Numerically-Aware Graph for Discrete Reasoning

**Xiangru Tang**[1,2,3], **Xinhui Tu**[1,2]*
[1]School of Computer, Central China Normal University, China
[2]Hubei Provincial key Laboratory of Artificial Intelligence and Smart Learning, China
[3]Department of EECS, UC Berkeley, USA
xrtang@berkeley.edu ,tuxinhui@mail.ccnu.edu

## Abstract

Numerical reasoning is essential for reading comprehension, which requires symbolic numerical understanding and complex reasoning. Most existing approaches rely on specialized neural modules, which are hard to adapt to multiple domains or multi-step reasoning. In this work, we aim at developing a hybrid neural-symbolic approach for performing numerical reasoning (specifically on the DROP dataset). We present SYMGraph, which includes a BERT encoder, a graph neural network(GNN) and a symbolic function module. Our main point is to handle the last restriction of NumNet, having output layers restricted to question type. We address this by outputting a program which has operators defined both for answering numerical questions as well as in-text span based questions, which involve simple arithmetic as well as extraction. Moreover, our model generates a program and executes it directly, instead of using expression templates. DROP is a machine reading comprehension dataset that tests the comprehensive understanding of paragraphs and includes arithmetic, counting, and sorting, as well as a strong contextual understanding of the text, and we show a 34.99%/35.95% absolute improvement of EM/F1 by SYMGraph over NABERT.

## Introduction

Machine reading comprehension (MRC) is rapidly becoming a proxy for gauging a model's natural language understanding capability. One of the most commonly used reading comprehension datasets in recent years is SQuAD (Rajpurkar et al. 2016). In this dataset, the answers to the questions are spans of text contained within the passages. Numerous attention-based models have been developed that have achieved high performance nearing or exceeding human performance, such as QANet (Yu et al. 2018b) and BiDAF (Seo et al. 2016), as well as many models taking advantage of BERT (Devlin et al. 2018) word embeddings.

These attention-based models face huge challenges when questions become more complex since it requires multiple steps of reasoning against paragraphs, especially when they involve discrete and symbolic operations. More specifically, we are focusing on a large-scale MRC data set named DROP(Dua et al. 2019), where an agent needs to understand the compositional reasoning structure of the questions, perform accurate information extraction from the passage (eg. extract lengths, kickers, and more. for the field goals and touchdowns), and perform symbolic reasoning (eg. counting, sorting, and more). DROP presents both opportunities and challenges to researchers and requires a comprehensive understanding of both the question semantics and the text.

The existing methods that try to answer these compositional questions can be sorted into two broad categories: semantic parsing and pre-trained language model. Semantic parsing, such as SRL, maps natural language question to a expression and then yield the answers; however, it is limited to answering questions against structured knowledge and has a strong restriction of question type. The pre-trained language model, such as BERT for solving the DROP task, must follow some specialized output modules corresponding to each type of question. These specialized output modules aim at achieving different numerical reasoning, especially four types: (1) span; (2) arithmetic expression; (3) count number; (4) negation on numbers in DROP dataset. Evidently, the specialized output modules rely on the handcrafted decoder, which needs human to predefine the set of answer types. However, it is difficult to scale this model to multi-step complex reasoning. Our main point is to handle the restriction of these handcrafted decoder, having output layers restricted to question type.

In this work, we propose the SYMGraph for discrete reasoning over paragraphs, which consists of three parts: (a) an encoder(BERT) that encodes text into a vector; (b) a graph neural network to explicitly represent the numerical relationships; and (c) a symbolic module that includes a function generator and function calculator. It can scale better than other approaches to multi-step complex reasoning because it does not differ the question type and feed the whole question-answer pairs into the same following modules. Compared with the newest model NumNet(Ran et al. 2019), our model outputs a program after the graph neural network, rather than having several different output layers dependent on the question type. In other words, the main point of this work is to handle that last restriction of NumNet(Ran et al. 2019), having output layers restricted to ques-

*Corresponding author

tion type. Our work paper addresses this by outputting a program which has operators defined both for answering numerical questions as well as in-text span based questions.

The main process in SYMGraph is that, firstly, it feeds the question and passage vector represented by BERT into a numerically-aware graph neural network. Then, a symbolic module produces compositional functions with predefined symbolic operators, which are executed similar to semantic parsing, to determine a final answer. This innovative paper combines the graph neural network and a symbolic module that generates functions, which are executed to produce answers. The main challenge when training SYMGraph is the lack of annotations, which was resolved with hard EM with thresholding. To tackle it, our training algorithm works by heuristically searching for programs that answer the question, and maximizing the likelihood of these programs with hard-EM-style parameter updates.

## Related Work

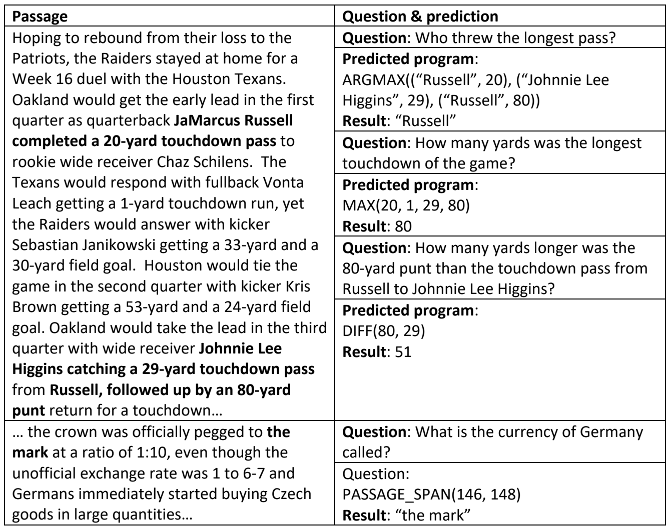
### Machine Reading Comprehension

Machine reading comprehension (MRC) tasks that provide full supervision for answer spans (Rajpurkar et al. 2016) have seen significant progress. (Seo et al. 2016; Xiong, Zhong, and Socher 2017; Yu et al. 2018a; Devlin et al. 2018). Recently, the community has moved towards more challenging tasks such as distantly supervised MRC (Joshi et al. 2017), reading comprehension with free-form human generated answers (Kočiskỳ et al. 2018) and reading comprehension requiring discrete or multi-hop reasoning (Dua et al. 2019; Yang et al. 2018). These tasks introduce new learning challenges since the optimal solution that is required to answer the question (e.g. a span or an equation) is not given. Nevertheless, not much work has been done for this particular learning challenge. Most work on MRC focuses on the model architecture and simply chooses the first span or a random span from the document (Joshi et al. 2017; Tay et al. 2018; Talmor and Berant 2019), rather than modeling this uncertainty as a latent choice. Others maximize the sum of the likelihood of multiple spans (Kadlec et al. 2016; Swayamdipta, Parikh, and Kwiatkowski 2017; Clarke et al. 2010; Lee, Chang, and Toutanova 2019), but it is unclear if it gives a meaningful improvement.

### Discrete Reasoning Over Paragraphs

In order to improve arithmetic performance, we turn to diction and syntax in a particular word problem. Essentially, we are attempting to perform numerical reasoning alongside reading comprehension, as we usually need to evaluate a mathematical expression described in a passage to arrive at the correct answer. In some work (Kushman et al. 2014; Wang, Liu, and Shi 2017), the general approach is to generate equation templates based on the word problem, and map numbers into the templates. Kushman et al. use the training set to obtain a set of possible templates, which are selected and filled at inference time, while Wang, Liu, and Shi use a RNN to encode the word problem to a context vector and another RNN to decode the context vector to an equation template.

In recent years, a new dataset, DROP, introduced questions that were expected to present a harder challenge for reading comprehension models (Dua et al. 2019). At the beginning Dua et al. used QANET (Yu et al. 2018b) as the encoder and added four different output layers onto it, including Passage Span, Question Span, Count and Arithmetic. Then, Min et al. extended a BERT-based extractive reading comprehension model to replace QANET with a lightweight extraction and composition layer. MTMSN (Hu et al. 2019) is the first to address the multi-span questions of DROP, which consists of dedicated categorical variable and the non-maximum suppression (NMS) algorithm to find the most probable set of non-overlapping spans. Following this, Efrat, Segal, and Shoham proposed a new approach for addressing multi-span questions based on sequence tagging. From a different perspective, Min et al. considered to derive some different mathematical equations by focusing the numbers in the reference text to calculate the answer. They introduced a discrete latent variable learning algorithm, which is a procedure of predicting the most likely solution in the precomputed set and further increases the likelihood of that solution. In the latest work, Ran et al. proposed a model named NumNet, especially for numerical reasoning, which utilizes a numerically-aware graph neural network to achieve numerical reasoning. They firstly extracted the number and then constructed the graph. However, it still divided the answer into four categories, which is not scalable to a multi-domain question-answering dataset because it relies on predefined answer decoders.



| **Passage** | **Question & prediction** |
|---|---|
| Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week 16 duel with the Houston Texans. Oakland would get the early lead in the first quarter as quarterback **JaMarcus Russell completed a 20-yard touchdown pass** to rookie wide receiver Chaz Schilens. The Texans would respond with fullback Vonta Leach getting a 1-yard touchdown run, yet the Raiders would answer with kicker Sebastian Janikowski getting a 33-yard and a 30-yard field goal. Houston would tie the game in the second quarter with kicker Kris Brown getting a 53-yard and a 24-yard field goal. Oakland would take the lead in the third quarter with wide receiver **Johnnie Lee Higgins catching a 29-yard touchdown pass** from **Russell, followed up by an 80-yard punt** return for a touchdown… | **Question**: Who threw the longest pass? |
| | **Predicted program**: ARGMAX((“Russell”, 20), (“Johnnie Lee Higgins”, 29), (“Russell”, 80)) **Result**: “Russell” |
| | **Question**: How many yards was the longest touchdown of the game? |
| | **Predicted program**: MAX(20, 1, 29, 80) **Result**: 80 |
| | **Question**: How many yards longer was the 80-yard punt than the touchdown pass from Russell to Johnnie Lee Higgins? |
| | **Predicted program**: DIFF(80, 29) **Result**: 51 |
| … the crown was officially pegged to **the mark** at a ratio of 1:10, even though the unofficial exchange rate was 1 to 6-7 and Germans immediately started buying Czech goods in large quantities… | **Question**: What is the currency of Germany called? |
| | Question: PASSAGE_SPAN(146, 148) **Result**: “the mark” |

Figure 1: Some examples of operators, passages, question-functions and results.

## SYMGraph

In this section, we will introduce the architecture of our model SYMGraph and describe the details of it for discrete reasoning over paragraphs. This is further illustrated in Figure 1.

| Operator | Arguments | Outputs | Description |
|---|---|---|---|
| `SPAN_p` `SPAN_q` | a0: the start index. a1: the end index. | a span. | Select a span from the passage or question. |
| `VALUE` | a0: index. | a number. | Select a number. |
| `KV` | a0: a span. a1: a number. | a key-value pair. | Select a key-value. |
| `DIFF` `SUM` | a0: index. a1: index. | a number. | Compute the difference sum of two numbers. |
| `COUNT` | a: spans. | a number. | Count the number of given spans. |
| `MAX` `MIN` | a: numbers. | a number. | Select the maximum. or minimum. |
| `ARG` | a: key-value | a span. | Select the argmax or argmin one |

Table 1: The illustration of different operators.

## Components

- Numerical Encoder: encodes question and passage into a vector, which represents a semantic meaning.
- Graph Neural Network: deciphers the relationship between the numbers in a paragraph.
- Symbolic Module: generate and execute the symbolic functions with pre-defined operators.

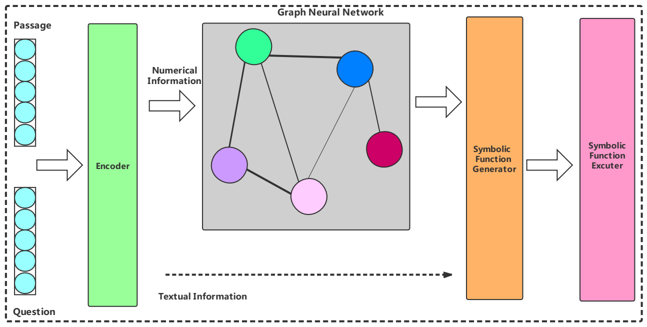


Figure 2: The structure of SYMGRAPH. It can be divided into three parts which are encoder, graph neural network and symbolic module.

## Numerical Encoder

In this part, we will thoroughly discuss the comparison to related graph neural network, in particular, the differences between NumNet(Ran et al. 2019) and our work, as they combine a Bert-style encoder with a numerically aware graph neural network. Our model outputs a program after the graph neural network, rather than having several different output layers dependent on the question type.

We consider the numbers from the question and passage as nodes in the graph for numerical reasoning. The encoding module includes BERT part and GNN part. For NumNet, there major contribution is the unique reasoning module, thus their work can be considered as a combination of NAQANet and GNN. And our graph encoding part is the same as theirs.

Specifically, for the embedding representation of encoder, we feed the question and passage jointly into BERT. The question Q and passage P are first encoded as below, followed by the passage-aware question representation and the question-aware passage representation:

$$P = \mathbf{BERT} - \mathbf{Encoder}(P) \tag{1}$$

$$Q = \mathbf{BERT} - \mathbf{Encoder}(Q) \tag{2}$$

$$\hat{P} = \mathbf{Attention}(P, Q); \hat{Q} = \mathbf{Attention}(Q, P) \tag{3}$$

## Graph Construction

Compared with NumNet, we don't utilize the graph to achieve supplementary reasoning, our aim is to extract numbers and enhance its representation. The intention is based on the fact that BERT cannot distinguish the meanings of different numbers clearly. We treat the numbers from the passage as nodes in the graph, which can be denoted as $V_Q$ and $V_P$, respectively arise in question and passage. We indicate the sets of nodes as $V = V_Q \cup V_P$ , and the number of node $v \in V$ as $n(v)$. To construct the graph, firstly, we establish a heterogeneous directed graph $G = (V; E)$, in which the nodes $V$ are the numbers in the question and passage, and edges $E$ are used to represent numerical relationships of these numbers.

We have two nodes $v_i, v_j \in V$, for each node $v_i^P \in V_P$, it is initialized as the corresponding column vector of $M_P$. The initial vector is $v_i^P = M^P[I^P(v_i^P)]$, where $I^P(v_i^P)$ indicates the word index of $v_i^P$ . In an identical way, the initial vector $v_j^Q$ for a node $v_j^Q \in V_Q$ is fixed as the corresponding column vector of $M_Q$. We represent all the initial node vector as $v_0 = v_i^P \cup v_j^Q$ and use a GNN to express graph $G$ and the node representations $v$. Then, we calculate the weight of each node for further computation.

$$\alpha_i = sigmoid(W_v v[i] + b_i) \tag{4}$$

In equation, $W_v$ is the weight of a node $v_i$, and $b_v$ is a bias. We employ relation-specific transform matrices in the process of message propagation and thus in the following we define the propagation function, which aims at computing the forward-pass.

$$\hat{v_i} = \frac{1}{|N_i|}(\sum \alpha_j W^{r_{ji}} v[j]) \tag{5}$$

where $\hat{v}_i$ is the message representation of node $v_i$, $r^i_j$ is the relation assigned to edge $e^i_j$, $W^{r_{ji}}$ are relation-specific transform matrices, and $N_i = \{j|(v_j, v_i) \in E\}$ is the neighbors of node $v_i$. Additionally, each node of the graph should also embody its own information as well, not only the neighbors' information.

$$v = \boldsymbol{ReLU}(W_f v_i + \hat{v}_i + b_f) \tag{6}$$

Finally, we indicate the above process as a single function to obtain a numerical relationship. And in this way, the output embedding of graph encoder will then concatenate to pre-trained model's embedding.

$$v = \mathrm{GCN}(G, v) \tag{7}$$

## Symbolic Module

Although NumNet(Ran et al. 2019) also combines a BERT-style encoder with a numerically-aware graph neural network, there are huge differences in our model. In particular, the differences between NumNet and this work are more thoroughly discussed in this part, which can be summarized that our model generates a program and executes it directly, instead of using expression templates

And most importantly, we design ten operators for this task of discrete reasoning over paragraphs: DIFF, SUM, COUNT, ARG, MAX, MIN, PASSAGE-SPAN, QUESTION-SPAN, VALUE, KEY-VALUE. The arguments and description of these operators are shown in Table 1. Moreover, these operators can be combined to obtain the target function.

Since it may be hard to understand what some of the operators are by its name, we give some examples (Figure 2) and each operator has one or two arguments. PASSAGE-SPAN and QUESTION-SPAN select a span from the text, whose arguments are the start index and the end index. For COUNT (count the number of given spans),MAX and MIN, the arguments are some spans or numbers. The arguments of KEY-VALUE are a span and a number, which select a key (span) value (number) pair from the passage. The argument of VALUE is a index, which select a number from the text. For ARGMAX and ARGMIN, the arguments are some key-value pairs, and which select the key (span) with the highest / lowest value.

There are some following problems, we do not have correct function and this incomplete data with a lack of the target program cannot be directly used in training. What is more, there must be only one function corresponding to the question, but actually we may produce a wrong function which leads a correct answer as well. Firstly, we try to find programs for questions answerable by span selection or arithmetic operations via an exhaustive search. But actually for counting or sorting, the space becomes too large for an exhaustive search and usually includes more than one span as an argument. Thus, we augment the span selection questions by replacing the interrogatives For counting problem and extract the key-value pairs with entity recognition bash for sorting problem.

## Hard EM

Another problem encountered in training the program was when a wrong program may predict a right answer. Hard EM uses the current model to select the program with the highest model probability among the ones that return the correct answer, and then maximizes the likelihood of the selected program.

However, there exists the third question that the annotated answer itself may is wrong. So when directly applying the hard EM algorithm, even if the model probabilities for all the programs are very small, it will still select a program for training. In the future, we can adopt RL-based approaches such as MAPO solve it.

**Hard EM Algorithm** The input is question-answer pairs $(x_i, y_i)$, we apply Data Augmentation for each $(x_i, y_i)$ to produce a program set. Then repeat: (a) get a new threshold*=decay and $D$ is a empty set; (b) for each $(x_i, y_i)$: we can pick a $program = argmax_k p_\theta(program|x_i)$ form the program set and if $p_\theta(program|x_i) > threshold$ and program set=1 then $D = D + (X_i, program)$; (c) Update $\theta$ by maximizing $\sum_D logp_\theta(program|x)$ until converge or early stop.

# Experiment

## Dataset and Evaluation Metrics

Our proposed model is evaluated on the DROP dataset (Dua et al. 2019), which is a public numerical MRC dataset. DROP dataset is constructed by crowd-sourcing, and the authors ask the annotators to generate question-answer pairs according to some given Wikipedia passages. Unlike the other datasets based on Wikipedia, DROP emphasize the numerical reasoning, such as addition, counting, and sorting over numbers, called as discrete reasoning. In details, DROP is divided into training set, which contains 77,409 samples, development set, which contains 9,536 samples, and 9,622 testing samples. These specifications can be found in Table 4. For this study and experiment, we adopt two metrics: (1) Exact Match (EM) and (2) numerically-focused F1 scores to evaluate our model following (Dua et al. 2019). If the predicted answer is mismatched for those questions with the numeric correct answer, F1 is set to 0, otherwise it will be 1.

## Baselines

In order to comparison, we select several public models as baselines, which are described as below. It's worth noting that we only compared with the single model approaches, and we don't consider some enhanced version, such as using a pretrained RoBERTa model.

- Syn Dep (Dua et al. 2019), the neural semantic parsing model (KDG) (Krishnamurthy, Dasigi, and Gardner 2017) with Stanford dependencies based sentence representations;
- OpenIE (Dua et al. 2019), KDG with open information extraction based sentence repre- sentations;
- SRL (Dua et al. 2019), KDG with semantic role labeling based sentence representations;

| Model | EM (Dev) | F1 (Dev) | EM (Test) | F1 (Test) |
|---|---|---|---|---|
| Heuristic Baseline (Dua et al. 2019) | 4.28 | 8.07 | 4.18 | 8.59 |
| Syn Dep (Dua et al. 2019) | 9.38 | 11.64 | 8.51 | 10.84 |
| OpenIE (Dua et al. 2019) | 8.80 | 11.31 | 8.53 | 10.77 |
| Semantic Role Labeling (Carreras and Màrquez 2004) | 9.28 | 11.72 | 8.98 | 11.45 |
| BiDAF (Seo et al. 2016) | 26.06 | 28.85 | 24.75 | 27.49 |
| QANet+ELMo (Yu et al. 2018b) | 27.71 | 30.33 | 27.08 | 29.67 |
| BERT BASE (Devlin et al. 2018) | 30.10 | 33.36 | 29.45 | 32.70 |
| NAQANet (Dua et al. 2019) | 46.20 | 49.24 | 44.07 | 47.01 |
| NABERT+ | 62.59 | 66.46 | 61.60 | 65.12 |
| MTMSN LARGE (Efrat, Segal, and Shoham 2019) | 76.54 | 80.76 | 75.88 | 79.99 |
| NumNet (Ran et al. 2019) | 64.92 | 68.31 | 64.56 | 67.97 |
| SYMGraph (Ours) | **81.24** | **83.09** | **79.06** | **82.96** |

Table 2: Comparing test and development set results of models from the official DROP leaderboard.

| **Model** | **Multi-Span (4.8%)** | | **Single-Span (31.7%)** | | **Number (61.9%)** | | **Date (1.6%)** | |
|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| NAQANet | 0.00 | 27.33 | 58.24 | 64.82 | 44.97 | 45.09 | 32.03 | 39.61 |
| NABERT | 6.53 | 27.48 | 65.56 | 70.62 | 54.27 | 54.29 | 37.58 | 46.38 |
| MTMSN | 25.17 | 62.86 | 77.58 | 82.88 | 80.93 | 81.15 | 55.71 | 69.07 |
| SYMGraph (test set) | 52.18 | 78.32 | 75.98 | 80.68 | 83.25 | 84.87 | 59.12 | 68.18 |

Table 3: Performance of different models on DROP's test/development set in terms of Exact Match (EM) and F1.

| **Statistic** | **Train** | **Dev** |
|---|---|---|
| Num passages | 5565 | 582 |
| Num questions | 77409 | 9536 |
| Avg questions / passage | 13.91 | 16.38 |
| Avg answer / question | 1.00 | 3.35 |
| Avg passage len (tokens) | 278.39 | 256.10 |
| Avg question len (tokens) | 12.89 | 13.39 |
| Passage vocab size (tokens) | 20986 | 11102 |
| Question vocab size (tokens) | 14051 | 5794 |

Table 4: DROP dataset specs.

and traditional MRC models:

- BiDAF (Seo et al. 2016), a MRC model, which utilizes a bi-directional attention flow network to encode the question and passage;
- QANet (Yu et al. 2018b), which utilizes convolutions and self-attentions as the building blocks of encoders to represent the question and passage; • BERT (Devlin et al., 2019), a pre-trained bidirectional Transformer based language model which achieves state-of-the-art performance on lots of public MRC datasets recently; and numerical MRC models:
- BERT (Devlin et al. 2018), a pre-trained bidirectional Transformer-based language model which achieves state-of-the-art performance on lots of public MRC datasets recently;

and numerical MRC models:

- NAQANet (Dua et al. 2019), a numerical version of QANet model.
- NAQANet+, an enhanced version of NAQANet implemented by ourselves, which further considers real numbers (e.g. "2.5"), richer arithmetic expression, data augmentation, and more. The enhancements are also used in our NumNet model and the details are given in the Appendix.
- NumNet (Ran et al. 2019), considers numerical comparing information among numbers when answering numerical questions. We don't compare with some enhanced versions for the fairness, because our intention doesn't include the tricks of MRC task.

## Result

Two metrics, namely Exact Match (EM) and F1 score, are utilized to evaluate models. We use the official script to compute these scores. Since the test set is hidden, we only submit the best single model to obtain test results. Table 2 shows the performance of our model and other competitive approaches on the development and test sets. SYMGraph outperforms all existing approaches by a large margin, and creates new state-of-the-art results by achieving an EM score of 79.06 and a F1 score of 82.96 on the test set. Since our best model utilizes BERT LARGE as encoder, we therefore compare our model with MTMSN LARGE and NABERT LARGE baseline. As we can see (Table 3), our model obtains 3.18/2.97 absolute gain of EM/F1 over MTMSN LARGE and 34.99/35.95 absolute gain of EM/F1 over NABERT LARGE, demonstrating the effectiveness of our approach. However, as the human achieves 95.98 F1 on the test set, our results suggest that there is still room for improvement.

## Analysis

Our experiment utilized 21,800 questions from the recently proposed DROP dataset (Dua et al. 2019) that are heuristically chosen based on their first n-gram such that they are covered by our designed modules. This is a significantly-sized subset that poses a wide variety of reasoning challenges and allows for controlled development and testing of models. We show that our model, which has interpretable intermediate outputs by design, significantly outperforms state-of-the-art black box models on this dataset.

## Conclusion and Future Work

In this paper, we explore the problem of discrete reasoning over paragraphs, including arithmetic, counting, and sorting, where complex questions could be correctly answered. To tackle this, we propose SYMGraph, a semantic parser that operates over passages and coupled with a numerically-aware graph. Our model handle the last restriction of existing graph model, such as NumNet, having output layers restricted to the question type. The key insight is to let the semantic parser point to locations in the text that can be used in further symbolic functions in a Hard-EM training. Then, we describe the SYMGraph and show that it can effectively answer questions better than baseline models on DROP datasets. Specifically, SYMGraph combines symbolic methods and neural networks, such as an advanced language model (BERT (Devlin et al. 2018)) and represents learning (graph neural network), enabling them to achieve numerical reasoning efficiently. Our work better defines the issues with the neural symbolic approach and sets the stage for future work on this problem. We plan to generalize this model to more complex and compositional machine reading comprehension data-sets with better strategies of the neural symbolic approach.

## Acknowledgments

We thank Liang Pang for useful discussions and Haoyuan Wang for assistance with data analysis. This work is substantially supported by by the National Natural Sci-ence Foundation of China under the grant number 61572223 and the University-Industry Collaborative Education Program between the Ministry of Education of China and Google Information Technology (China) Co., Ltd. (PJ190496).

## References

Carreras, X., and Màrquez, L. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, 89–97.

Clarke, J.; Goldwasser, D.; Chang, M.-W.; and Roth, D. 2010. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, 18–27. Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dua, D.; Wang, Y.; Dasigi, P.; Stanovsky, G.; Singh, S.; and Gardner, M. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Efrat, A.; Segal, E.; and Shoham, M. 2019. Tag-based multi-span extraction in reading comprehension. *arXiv preprint arXiv:1909.13375*.

Hu, M.; Peng, Y.; Huang, Z.; and Li, D. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. *arXiv preprint arXiv:1908.05514*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Kadlec, R.; Schmid, M.; Bajgar, O.; and Kleindienst, J. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Kočiskỳ, T.; Schwarz, J.; Blunsom, P.; Dyer, C.; Hermann, K. M.; Melis, G.; and Grefenstette, E. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics* 6:317–328.

Krishnamurthy, J.; Dasigi, P.; and Gardner, M. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1516–1526.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 271–281.

Lee, K.; Chang, M.-W.; and Toutanova, K. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Min, S.; Chen, D.; Hajishirzi, H.; and Zettlemoyer, L. 2019. A discrete hard em approach for weakly supervised question answering. *arXiv preprint arXiv:1909.04849*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ran, Q.; Lin, Y.; Li, P.; Zhou, J.; and Liu, Z. 2019. Numnet: Machine reading comprehension with numerical reasoning. *arXiv preprint arXiv:1910.06701*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Swayamdipta, S.; Parikh, A. P.; and Kwiatkowski, T. 2017. Multi-mention learning for reading comprehension with neural cascades. *arXiv preprint arXiv:1711.00894*.

Talmor, A., and Berant, J. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. *arXiv preprint arXiv:1905.13453*.

Tay, Y.; Luu, A. T.; Hui, S. C.; and Su, J. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems*, 4906–4917.

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854.

Xiong, C.; Zhong, V.; and Socher, R. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yu, A. W.; Dohan, D.; Le, Q.; Luong, T.; Zhao, R.; and Chen, K. 2018a. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018b. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.