A Neural-Symbolic Approach to Natural Language Understanding

Zhixuan Liu*
Shanghai Jiaotong University
lzx993124494@sjtu.edu.cn

Zihao Wang*
CSE, HKUST
zwanggc@cse.ust.hk

Yuan Lin
ByteDance AI Lab
linyuan.0@bytedance.com

Hang Li
ByteDance AI Lab
lihang.lh@bytedance.com

Abstract

Deep neural networks, empowered by pre-trained language models, have achieved remarkable results in natural language understanding (NLU) tasks. However, their performances can drastically deteriorate when logical reasoning is needed. This is because NLU in principle depends on not only analogical reasoning, which deep neural networks are good at, but also logical reasoning. According to the dual-process theory, analogical reasoning and logical reasoning are respectively carried out by *System 1* and *System 2* in the human brain. Inspired by the theory, we present a novel framework for NLU called Neural-Symbolic Processor (NSP), which performs analogical reasoning based on neural processing and logical reasoning based on both neural and symbolic processing. As a case study, we conduct experiments on two NLU tasks, question answering (QA) and natural language inference (NLI), when numerical reasoning (a type of logical reasoning) is necessary. The experimental results show that our method significantly outperforms state-of-the-art methods in both tasks.[1]

1 Introduction

Natural language understanding (NLU) has made remarkable progress recently, when pre-trained language models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and BART (Lewis et al., 2020) are exploited. The deep neural networks based on pre-trained language models even exhibit performances superior to humans in the tasks of question answering (QA) (Rajpurkar et al., 2016) and natural language inference (NLI) (Ghaeini et al., 2018). However, language understanding requires not only analogical reasoning, which deep neural networks are good at (Bengio et al., 2021), but also logical reasoning, including numerical reasoning. For example, to answer the questions in Table 1 or to infer the entailment relations in Table 2, we need to first 'interpret' the meanings of the input texts and then perform numerical reasoning, in general, logical reasoning, to obtain the final results. Exploiting deep neural networks alone would not easily accomplish the goal. Recently, there has been research to address the problems in QA and NLI. For example, a dataset called Discrete Reasoning Over Paragraphs (DROP) has been created for QA, and several methods have been proposed (Dua et al., 2019; Ran et al., 2019; Chen et al., 2020). Among them, Numerically-Aware QANet (NAQANet) (Dua et al., 2019) utilizes deep neural networks to individually solve the sub-problems of span extraction, counting, and numerical addition/subtraction. A dataset called AWPNLI (Ravichander et al., 2019) has also been created for NLI in which numerical reasoning is needed.

According to the dual-process theory developed by the psychologist Kahneman and others (Kahneman, 2003), human thinking is carried out by two different systems in the brain. *System 1* is a fast, unconscious, and effortless mode of thinking, often associated with analogical reasoning. In contrast, *System 2* is a slow, conscious, and effortful mode of thinking, also evoking logical reasoning. The theory should also hold for human language understanding, a specific case of thinking. Inspired by the theory, we propose a new framework for language understanding, which performs both analogical reasoning and logical reasoning, corresponding to *System 1* and *System 2* respectively. In fact, designing AI systems containing *System 1* and *System 2* is a popular research topic recently (e.g., (Bengio et al., 2021)). Our key idea is to employ a neural network to conduct analogical reasoning on the text input as usual and in the meantime, to employ

*The work was done when the first and second authors were interns at ByteDance AI Lab.

[1]The code and data are available at https://github.com/chadlzx/NSP_QA; https://github.com/zihao-wang/Number-NLI.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2159–2172
December 7-11, 2022

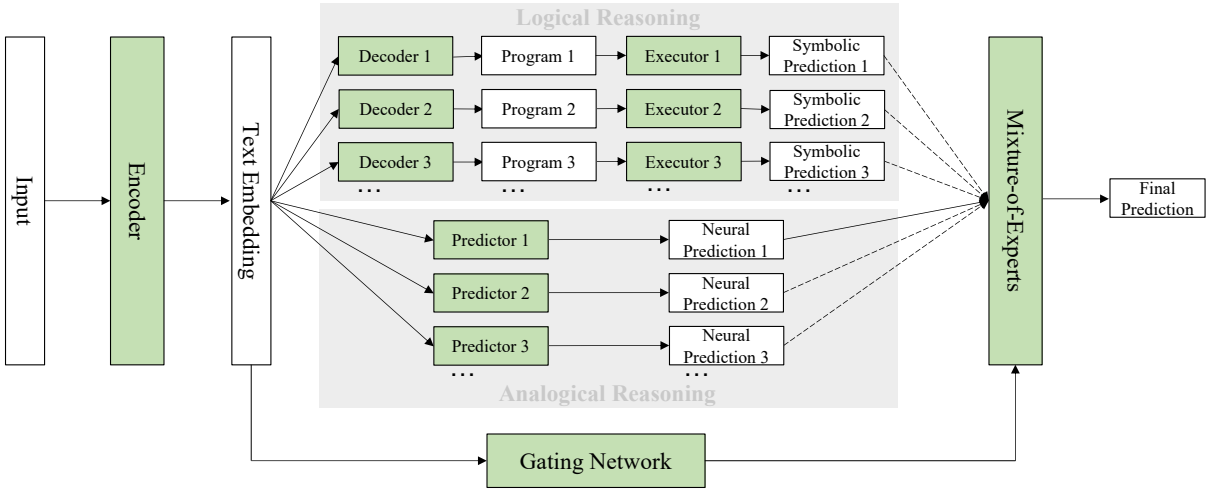

Figure 1: An overview of the Neural-Symbolic Processor framework. Analogical reasoning is performed by the predictors (neural processing). Logical reasoning is performed by the decoders and executors (neural and symbolic processing). A mixture-of-experts is used to make the final prediction.

another neural network to translate the text input into a program and a symbolic system to execute the program to perform logical reasoning.

The new framework for natural language understanding, called Neural-Symbolic Processor (NSP), is shown in Figure 1. First, the encoder transforms the input texts (the question and text in QA, the two texts in NLI) into a text embedding (an intermediate representation). Then, the predictors take the text embedding as input and generate neural predictions. In parallel, the decoders transform the text embedding into programs. Note that the encoder and the decoders form sequence-to-sequence models. Moreover, the executors take the programs as input and generate symbolic predictions. Finally, the mixture-of-experts (with a gating network) takes all the neural predictions and symbolic predictions as input and selects one of the predictions to make the final prediction.

We evaluate the NSP framework in QA and NLI. For the QA task, we sample a subset of DROP (about 32K instances), referred to as DROP-subset, and annotate a program for each question-answer pair requiring numerical reasoning. The experiments on the DROP-susbset show that our approach outperforms the baselines by 2.40% and 2.51% in terms of F1 score and exact match. In particular, for question-answer pairs requiring multiplication, division, and averaging, our approach improves F1 by 56.42%. For question-answer pairs requiring addition and subtraction, our approach improves F1 by 3.91%. For the NLI task, we use the AWPNLI dataset (Ravichander et al., 2019) and also annotate programs for each text pair. The experiments on AWPNLI show that our approach exceeds the baseline with a large margin of 20.7% in terms of F1 score.

The contribution of our work is as follows:

- We propose a new framework for NLU, Neural-Symbolic Processor, to conduct both analogical reasoning and logical reasoning.
- We perform experiments on QA and NLI to verify the effectiveness of our approach and show that our approach can achieve remarkable improvements in the tasks when they need numerical reasoning.
- We add programs into the two datasets of QA and NLI, DROP-subset and AWPNLI, and will release the annotated data.

2 Related Work

Question Answering with Logical Reasoning Given a short text and a question, the QA task is to predict the answer obtained from the short text. QA has made considerable progress in recent years. Models such as BiDAF (Seo et al., 2016), R-NET (Wang et al., 2017), and QANet (Yu et al., 2018) have been proposed, which have shown excellent performances on the benchmark dataset of SQuAD (Rajpurkar et al., 2016). Recently, the QA models empowered by the pre-trained language models of BERT (Devlin et al., 2018), XL-Net (Yang et al., 2019), and RoBERTa (Liu et al., 2019) have significantly advanced the performance of the task. Nonetheless, they still cannot effectively handle cases that need complex reasoning.

Passage	Question	Program & Prediction
... kicker Kris Brown getting a 53 - yard@N9 and a 24 - yard@N10 field goal. ...	How many more yards was Kris Browns's first@Q1 field goal over his second@Q2?	**Program**: diff(N9,N10) **Symbolic Prediction**: 29 **Ground-Truth**: 29
... The first@N1 issue in 1942@N2 consisted of denominations of 1@N3, 5@N4, 10@N5 and 50@N6 centavos and 1@N7, 5@N8, and 10@N9 Pesos. The next year@N10 brought "replacement notes" of the 1@N11, 5@N12 and 10@N13 Pesos ...	In which year@Q1 were there replacement notes of the 1@Q2 , 5@Q3 , and 10@Q4 pesos ?	**Program**: add(N2,N10) **Symbolic Prediction**: 1943 **Ground-Truth**: 1943

Table 1: Two examples of question answering requiring numerical reasoning. In our method, the numbers in the input are attached with special tokens (in orange). The programs are generated for logical reasoning, in addition to analogical reasoning. The symbolic predictions are obtained by executions of the programs.

To tackle the problem, several models have been developed. NAQANet (Dua et al., 2019) adapts the output layer of QANet to numerical reasoning by predicting answers from arithmetic computation over the numbers in a text. NumNet (Ran et al., 2019) and QDGAT (Chen et al., 2020) further utilize a numerically-aware graph neural network to encode numbers. The approaches still rely on neural networks, which are good at analogical reasoning but not logical reasoning, and thus cannot fundamentally resolve the problem.

There has also been existing work trying to perform symbolic processing for QA. BERT-Calculator (Andor et al., 2019) and NeRd (Chen et al., 2019b) generate executable programs to produce the final answers. (Gupta et al., 2020) employ a parser to generate a program comprised of neural modules from the question and a program executor based on neural module networks to find the answer. Unfortunately, the methods are not suitable for all QA problems. There are cases in which deep neural networks based on pre-trained language models can easily make accurate predictions. Employing the symbolic approach alone would not effectively solve the problem.

Natural Language Inference with Logical Reasoning NLI is a task of predicting the entailment relation between two texts, i.e., premise and hypothesis. The benchmark datasets for NLI, such as SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) are widely used. Pre-trained language models achieve state-of-the-art performances (Liu et al., 2019).

However, sometimes logical reasoning is also crucial for NLI (MacCartney and Manning, 2007). It has been shown that using pre-trained language models in NLI is sub-optimal when logical reasoning is needed, when it involves conjunction (Saha et al., 2020) and quantitative reasoning (Ravichander et al., 2019). Increasing the size of training data for fine-tuning cannot effectively address the issue. There are also rule-based methods to handle quantitative reasoning (Roy et al., 2015). However, the results are usually not satisfactory.

Applications of dual-process theory Several research groups have been working on the application of *System 1* and *System 2* into machine learning. For example, (Mittal et al., 2017) take the vector space model and reasoning in knowledge graphs as fast thinking and slow thinking, respectively, and propose a hybrid query processing engine for search. (Anthony et al., 2017) use a tree search as an analog of *System 2* to strengthen the planning in sequential decision-making. (Bengio, 2017) proposes a consciousness prior theory for learning high-level concepts and points out that the *System 2* abilities are closely related to consciousness. In (Chen et al., 2019a), the authors propose an end-to-end framework including a generative decoder (fast thinking) and a reasoning module (slow thinking) to solve complex tasks.

3 Neural-Symbolic Processor

We describe the proposed framework Neural-Symbolic Processor (NSP) in this section.

3.1 Overview

Figure 1 shows the architecture of the NSP framework. The framework contains an encoder, several predictors, several decoders and executors, and a mixture-of-experts. There are two types of reasoning: analogical reasoning and logical reasoning.

Premise	Hypothesis	E-Program	C-Program	Symbolic Prediction
Sam had 98.0@M1 pennies in his bank and he spent 93.0@M2 of his pennies.	He has 5.0@N1 pennies now.	diff(M1, M2)=N1	diff(M1, M2)!=N1	Entailment
In a school, there are 542@M1 girls and 387@M2 boys.	928@N1 pupils are there in that school.	add(M1,M2)=N1	add(M1,M2)!=N1	Contradiction

Table 2: Two examples of natural language inference requiring numerical reasoning. In our method, the numbers in the input are attached with special tokens (in orange). The E-programs and C-programs are generated for logical reasoning, in addition to analogical reasoning. The symbolic predictions are obtained by executions of the programs.

The predictors are for analogical reasoning, and the executors and decoders are for logical reasoning. The encoder and the mixture-of-experts are shared. The framework can be utilized in an NLU task such as QA and NLI.

The encoder takes a pair of texts as input and transforms it into a text embedding (intermediate representation). A predictor takes the embedding as input and generates a neural prediction. The prediction can be classification, span extraction, or sequence tagging. In parallel, a decoder takes the embedding as input and generates a program. The executor executes the program and generates a symbolic prediction. The program can represent a logical reasoning for the task. Finally, the mixture-of-experts takes the neural and symbolic predictions and makes a final prediction. Note that the encoder and each of the decoders form a sequence-to-sequence model. The encoder, predictors, and decoders are all assumed to be based on a pre-trained language model such as BART, RoBERTa, and BERT.

For example, in the first example of QA in Table 1, to give the correct answer, one needs to calculate 53-24 to obtain the result of 29. Employing neural reasoning alone would not easily accomplish the task. Using NSP, we attach 53 and 24 with the special tokens N9 and N10 representing the numbers in the pre-processing. We also generate the program diff(N9, N10) describing the calculation and obtain the correct result by executing the program. This is in parallel with analogical reasoning.

In the first example of NLI in Table 2, to give the correct answer, one needs to calculate 98-93 to obtain the result of 5. In our method, we attach the numbers 98 and 93 with the special tokens M1 and M2. We also generate the two programs and obtain the correct result by executing the programs.

3.2 Encoder

The encoder is a Transformer encoder. The output of the encoder is used for both analogical and logical reasoning. (By default there is only one encoder. There can be also two encoders, one for analogical reasoning and the other for logical reasoning.)

There is a pre-processing before the encoder, in which the numbers in the input are attached with special tokens, as shown in Table 1 and Table 2.

3.3 Analogical Reasoning

The system for analogical reasoning predicts the answers from the input, using the predictors. Each of the predictors is a task-specific layer from the encoder. We next give the details in QA and NLI.

QA Task For the QA task, the encoder takes the passage-question pair as input and outputs `[CLS]` and token representations. In our experiments, the encoder is built on RoBERTa. The predictors take the representations as input and output neural predictions. Each predictor deals with one type of answer: span extraction, sequence labeling, or classification. It is a standard model for QA.

- Span extraction: The predictor predicts the answer as a contiguous span in the passage or in the question. It calculates each token's beginning/end probability and extracts the span with the largest probability. The probability of an answer is defined as the product of the probabilities of the beginning and end tokens.
- Sequence labeling: The predictor predicts the answer as non-contiguous spans in the passage using the token representations. It decides for each token whether or not it belongs to the answer.
- Classification: The predictor views QA as a classification problem. It predicts the answer

E-Program	C-Program	NLI prediction
True	True	Invalid
True	False	Entailment
False	True	Contradiction
False	False	Neutral

Table 3: The relation between the outputs of E-Program and C-Program and the NLI predictions.

as a class label using the `[CLS]` representation. For example, the classes can be ten digits 0-9.

NLI Task For the NLI task, the encoder takes a pair of texts as input (premise and hypothesis). The predictor makes a three-class classification using the `[CLS]` representation, deciding the relation between the text pair: entailment, neutral and contradiction. It is a standard model for NLI.

3.4 Logical Reasoning

The system for logical reasoning predicts the answers from the input, using the decoders and executors. A decoder transforms the input into a program based on the output of the encoder. The corresponding executor then executes the program. If the generated program is not valid, the executor will return NULL. In our experiments, the sequence-to-sequence models are based on BART (Lewis et al., 2020). Note that the numbers in the input are attached with special tokens for generating the programs, and the programs also utilize the special tokens. The definitions of functions in the programs are given in Appendix A. We next give details in QA and NLI.

QA Task There is only one decoder. The decoder generates the programs for QA, like those in Table 1. For example, the first program diff(N9, N10) means subtracting N9 from N10 in the input. The executor takes the program, makes substitutions N9=53 and N10=24, and then obtains the result of 29.

NLI Task For the NLI task, one program is insufficient to make a prediction. Therefore, we use two decoders to generate two programs: an entail program (E-Program) and a contradiction program (C-Program). E-Program becomes true if the premise entails the hypothesis. C-Program becomes true if the premise contradicts the hypothesis. Table 3 shows how to make the NLI prediction using the results of E-Program and C-Program.

For example, the E-Program is diff(M1,M2)=N1 and the C-Program is diff(M1,M2)!=N1 in the first example in Table 2. The first program predicts true if the equation of diff(M1-M2) equals N1 holds, and the second program predicts true if the equation of diff(M1-M2) equals N1 does not hold. The executors take the programs, make substitutions M1=98.0, M2=93.0, and N1=5.0, obtain the results of true and false, and make the final prediction as entailment, based on the decision rules in Table 3.

3.5 Mixture-of-Experts

The mixture-of-experts (MoE) (e.g., (Shazeer et al., 2017)) takes the neural predictions and symbolic predictions as input and selects the valid prediction with the highest probability (most likely to be correct) given by the gating network. The MoE will not select the logical reasoning result if it is NULL. The gating network is simply a prediction layer on the top of the encoder.

3.6 Training

Training has two phases: learning of the neural networks and learning of the MoE.

In the first phase, the encoder, predictors, and decoders are trained using the text inputs, programs, and ground-truth outputs. It is a multi-task learning with three objectives. The first objective is to make accurate predictions from the predictors, the second is to generate correct programs from the decoders, and the third is to make accurate predictions of answer types. The answer types indicate which types the correct answers belong to (classification, span, program execution, etc.).

In the second phase, the gating network of MoE is trained using the text inputs and the probabilities of predictions by answer types.

In the QA task, there are five types of answers (1) passage span extraction, (2) question span extraction, (3) sequence labeling, (4) number (0-9) classification, and (5) program execution, respectively corresponding to five analogical or logical reasoning modules. For each sample i in the training set, let $q_i = (q_{i1}, \cdots, q_{in})$ be the probabilities of predictions by answer types, where n denotes the number of answer types ($n = 5$ here). Let $p_i = (p_{i1}, \cdots, p_{in})$ be the outputs of the gating network, also by answer types. The parameters of the gating network are learned by using the text inputs and minimizing the KL-divergence in pre-

diction of answer types

$$\mathcal{L} = -\sum_{i}\frac{1}{n}\sum_{k=1}^{n}[q_{ik}\log(p_{ik}) + (1-q_{ik})\log(1-p_{ik})].$$

In the NLI task, there are only two types of answers, classification and program execution, respectively. Therefore, we adopt a simple strategy of selecting the type of predictions with the highest accuracy in the development dataset.

4 Experiment

4.1 QA Task

4.1.1 Dataset

DROP (Dua et al., 2019) is a dataset suitable for question answering requiring complex reasoning, such as multi-span extraction, arithmetic computation, counting, and multi-step reasoning. DROP is constructed from Wikipedia by crowd-sourcing, which contains 77,409 / 9,536 / 9,622 instances in the training / development / testing split. The training dataset does not have programs that we need in NSP. We sample a subset of the training dataset of DROP, named DROP-subset, containing 32,011 instances, and let human annotators annotate programs for the instances. (It is too costly to annotate all DROP data). In addition, we annotate programs for all data in the development dataset. The detailed annotation process is described in Appendix B. We use the development and test datasets of DROP as offline test and online test sets, respectively. Note that online test results are obtained from the official website, and thus there is no breakdown of the results.

Following the previous work (Dua et al., 2019), we adopt two evaluation metrics, Exact Match (EM) and F1 score, to conduct evaluations.

4.1.2 Baselines

We compare our method with several baselines. The first baseline is NA-RoBERTa (Numerically-Aware RoBERTa), which has a similar architecture to NAQANet (Dua et al., 2019), but uses RoBERTa as the encoder. NA-RoBERTa can be regarded as an approximation of using analogical reasoning in our method.

The other baselines are models previously applied to DROP: (1) QANet (Yu et al., 2018), a traditional reading comprehension model combining convolution and self-attention models. (2) NAQANet (Dua et al., 2019), a model which improves the output of QANet by adding a module to predict answers through arithmetic computation. (3) NumNet (Ran et al., 2019), a model which uses GNN to enhance the embedding of numerical features. (4) QDGAT (Chen et al., 2020), a model which improves NumNet by performing GNN on a heterogeneous graph. (5) NeRd (Chen et al., 2019b), a symbolic reasoning model which uses BERT as encoder and LSTM as a decoder to generate a program and then executes the program to produce the answer.

Method	Offline Test		Online Test	
	EM	F1	EM	F1
QDGAT	78.25	81.41	78.72	82.11
NA-RoBERTa	78.30	81.61	78.76	82.25
NSP (our method)	**80.81**	**84.01**	**79.74**	**83.31**

Table 4: The results of our method and baselines trained on DROP-subset in offline test and online test.

4.1.3 Experimental setting

For analogical reasoning, the encoder of our method is based on RoBERTa-large. All predictors (span extraction, sequence labeling, number classification) share the same encoder. We perform an end-to-end multi-task training for 20 epochs using the AdamW optimizer (Loshchilov and Hutter, 2018) with a batch size of 16. For the encoder, the learning rate is 1.5e-5, and the L2 weight decay is 0.01. For the predictor (prediction layer), the learning rate is 1e-4, and the L2 weight decay is 5e-5.

For logical reasoning, the sequence-to-sequence model of our method is based on BART-large. We train the model for 100 epochs using AdamW, with a batch size of 16. The learning rate is 1e-5.

4.1.4 Experimental Results

Table 4 shows the experimental results. Our method of NSP outperforms all baselines, achieving 84.01 in F1 score and 80.81 in EM in the offline test. NA-RoBERTa can be regarded as a model only performing neural reasoning. Our method outperforms NA-RoBERTa by 2.40 in F1 score and 2.51 in EM. Our method outperforms QDGAT by 2.60 in terms of F1 score and by 2.56 in terms of EM. Our method also performs better than all baselines in the online test. The experimental results demonstrate the effectiveness of our method.

Method	Offline Test		Online Test	
	EM	F1	EM	F1
QANet	27.50	30.44	25.50	28.36
NAQANet	46.20	49.24	44.07	47.01
NumNet	64.92	68.31	64.56	67.97
NeRd	78.55	81.85	78.33	81.71
QDGAT	82.74	85.85	83.23	86.38

Table 5: The results of methods trained on the full DROP training dataset in offline test and online test. The results are taken from the original papers.

Table 5 reports the results of existing methods trained on the *full* DROP training dataset and evaluated in terms of EM and F1 scores in the same offline test and online test. Except for QDGAT, the results are lower than NSP trained on DROP-subset. Note that when trained on DROP-subset, NSP outperforms QDGAT, as shown in Table 4.

We investigate the performances of our method of NSP and the baselines on different answer types in the offline test, as shown in Table 6. Our method of NSP performs the best on the answers relating to numbers and dates, demonstrating that NSP is strong in numerical reasoning compared with the other methods.

We further investigate the performances of our method of NSP and the baselines on different program types, add/diff, max/min, count, and mul/div/avg, as shown in Table 7. We can observe that NSP significantly outperforms in terms of F1 (+56.42) in mul/div/avg. It appears that NSP can generate programs for multiplication, division, and averaging, as shown in Appendix C, while NA-RoBERTa and QDGAT do not have the ability. The F1 of NSP also improves 3.91 and 2.6, respectively, in add/diff and max/min. The improvements are also significant. NSP can generate programs to perform multi-step addition, subtraction, and max/min operations, as shown in Appendix C. In contrast, it is hard for NA-RoBERTa and QDGAT to do so. The percentage of generated programs with invalid execution results (NULL) on the DROP-dev dataset is 0.1%.

4.2 Ablation Study

We conduct an ablation study. Table 8 presents the results. We compare NSP, and its components of logical reasoning only and analogical reasoning only in offline test with answer-types of number, span(s), and date. The results indicate that NSP performs much better than the two components, indicating that both components are necessary for NSP. Analogical reasoning is suitable for the span(s)-type. On the other hand, logical reasoning is suitable for the number and date types.

Method	Number	Span(s)	Date	Total
QDGAT	80.41	83.90	65.38	81.41
NA-RoBERTa	80.54	84.14	67.36	81.61
NSP (our method)	**84.24**	**84.36**	**68.49**	**84.01**

Table 6: The results of our method and baselines trained on DROP-subset and evaluated in terms of F1 on offline test. Number, span(s) and date are three answer types.

Method	add/ diff	max/ min	count	mul/ div/avg
Number of cases	4317	282	913	52
QDGAT	81.99	92.15	79.85	4.48
NA-RoBERTa	81.49	92.68	82.15	8.33
NSP (our method)	**85.90**	**95.28**	**82.91**	**64.75**

Table 7: The results of our method and baselines in terms of F1 on cases corresponding to different program types, add/diff, max/min, count, and mul/div/avg.

Table 9 reports the F1 scores of predictors and decoder of NSP for different answer types. We observe that the F1 score of NSP, after combining all components, achieves the highest performances in the number, span(s), and date types. It indicates that the components are complementary and that NSP can successfully ensemble them.

NSP utilizes additional program annotations compared with the baselines above. For further comparison, we consider two other alternatives based on NA-RoBERTa that also use the program annotations. The first method adds a program generation sub-task during the training, and only uses the original neural predictions in the inference. The second method utilizes programs to supervise its arithmetic computation, which predicts each number's coefficients (+1, -1, 0). In

Method	Number	Span(s)	Date	Total
Logical reasoning only	83.23	5.62	52.08	54.58
Analogical reasoning only	27.49	**84.39**	63.87	48.74
NSP (our method)	**84.24**	84.36	**68.49**	**84.01**

Table 8: Ablation study results on offline test of DROP. Compare only ensemble logical reasoning module and only ensemble analogical reasoning module with NSP.

Method	Number	Span(s)	Date	Total
passage span extraction	15.52	78.32	63.87	39.11
question span extraction	0.24	38.18	26.61	14.44
sequence labeling	5.47	58.91	41.07	25.45
classification (0-9)	20.19	0.86	0.30	12.85
program	83.23	5.62	52.08	54.58
NSP (our method)	**84.24**	**84.36**	**68.49**	**84.01**

Table 9: The results of NSP and its components in terms of F1 score.

Method	Number	Span(s)	Date	Total
NA-RoBERTa	80.54	84.14	67.36	81.61
NA-RoBERTa w/ program generation	79.43	82.90	66.37	80.46
NA-RoBERTa w/ program supervision	81.70	84.18	67.55	82.35
NSP (our method)	**84.24**	**84.36**	**68.49**	**84.01**

Table 10: Results of different baseline methods utilizing program annotation.

Table 10, we report the F1 scores of the additional methods. We observe that NSP still works the best. NA-RoBERTa with program supervision outperforms NA-RoBERTa. However, NA-RoBERTa with program generation performs even worse than NA-RoBERTa.

For the MoE in NSP, we train a gating network using text as input. We compare the MoE method with two alternatives, which respectively take the probabilities of answer predictions and the probabilities of answer-type predictions as input of the gating network. Table 11 shows the results. We observe that using text as input performs the best.

4.3 NLI Task

4.3.1 Dataset

AWPNLI (Ravichander et al., 2019) is a dataset suitable for NLI requiring reasoning. AWPNLI is created from arithmetic math problems in which symbolic processing is needed. AWPNLI is a small dataset with only 722 instances. We have human annotators annotate the E-Program and C-Program for each instance. The annotation process is described in Appendix B.

4.3.2 Baselines

We compare our method with neural and rule-based baselines. For the neural baselines, we use RoBERTa and BART as models for classification. The rule-based method Q-REAS proposed in (Ravichander et al., 2019) is also chosen. The performances of Q-REAS and two neural models are taken from the original paper (Ravichander et al., 2019).

Input of GN	Number	Span(s)	Date	Total
answer-prob	82.31	82.13	68.39	82.00
answer-type-prob	82.96	83.62	**69.69**	82.97
text	**84.24**	**84.36**	68.49	**84.01**

Table 11: Results of different input of gating networks (GN) in NSP.

RoBERTa and BART can be viewed as NSP with only analogical reasoning. We also consider NSP with only logical reasoning (also based on BART). The baselines can serve for ablation study.

4.3.3 Experimental Setting

We conduct a ten-fold cross-validation on the AWPNLI dataset and report the average and standard deviation of the results. The sequence-to-sequence model of our method is fine-tuned by using the input text and the annotated program based on BART. The predictor model is fine-tuned by using the input text and the ground truth based on RoBERTa. The optimizer is Adam, and the learning rate is 1e-5. The learning converges in 25k steps.

4.3.4 Results and Discussion

Table 12 shows the results of methods on the AWPNLI dataset. We find that our method of NSP is slightly better than its variant of logical reasoning only. The accuracies of the two methods are significantly better than those of the other methods. The results indicate that our method of NSP is effective when numerical reasoning is needed in NLI.

Furthermore, we can see that the methods of RoBERTa and BART are much worse than NSP and logical reasoning only. The results reported in previous work under the zero-shot setting are also significantly lower. We conclude that utilization of local reasoning capabilities such that of NSP is necessary for the task.

5 Conclusion

This paper proposes a novel framework for natural language understanding (NLU), referred to as Neural-Symbolic Processor (NSP). NSP employs two types of reasoning, analogical reasoning and logical reasoning. To 'understand' language, ana-

Model	Accuracy
Zero Shot Evaluation	
BART-large	42.2*
GPT	50.0*
Rule-based Q-REAS	71.5*
10-fold Cross Validation	
RoBERTa	49.85±0.35
BART	49.85±6.28
Logical Reasoning Only	**88.05±4.71**
NSP (our method)	**92.24±4.68**

Table 12: The results of NSP and baselines on AWPNLI. The results with * are taken from the original papers.

logical reasoning is performed by using neural networks as usual. In the meantime, logical reasoning is performed by using neural networks to generate programs and then using symbolic systems to execute the programs. Such an architecture is similar to that of humans, and the two types of reasoning correspond to *System 1* and *System 2* respectively in the human brain. Our approach thus is powerful in dealing with the challenging problems which conventional neural-network-only approaches suffer from. We evaluate our approach in two NLU tasks, QA and NLI. The experiments show that our method surpasses previous state-of-the-art methods with remarkable improvements when logical reasoning is needed.

Limitations

Although NSP outperforms the baselines in both QA and NLI, there are still complicated cases that the current NSP cannot effectively deal with.

For the QA task, We conduct an error analysis of NSP on the DROP dataset. Table 13 provides three typical categories of hard cases for NSP. The first type (first example) is related to multi-step reasoning. Such cases need a deep reasoning path or many arguments in functions. Another type (second example) is about the counting of long strings. Defining a standard for program annotation is hard because different human annotators may select strings with different lengths. The last type (third example) is related to complex conditions. The programs of NSP currently use a simple grammar language, which still cannot represent complicated conditions in reasoning.

We do not introduce a complicated programming language, because there is not enough training data to learn a model for generating complex programs. We leave this to future work.

We also conduct an error analysis of NSP for the NLI task. The primary type of errors (hard cases) is related to redundant numbers in the input texts, as shown in Table 14. One possible solution would be to increase the size of training data.

Acknowledgements

The authors thank Yuchen Zhang at ByteDance for his insightful comments in technical discussion. The authors thank anonymous reviewers for their helpful suggestions.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. 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)*, pages 5947–5952.

Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. *Advances in Neural Information Processing Systems*, 30.

Yoshua Bengio. 2017. The consciousness prior. *arXiv preprint arXiv:1709.08568*.

Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. 2021. Deep learning for AI. *Communications of the ACM*, 64(7):58–65.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Di Chen, Yiwei Bai, Wenting Zhao, Sebastian Ament, John M Gregoire, and Carla P Gomes. 2019a. Deep reasoning networks: Thinking fast and slow. *arXiv preprint arXiv:1906.00855*.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020. Question directed graph attention network for numerical reasoning over text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6759–6768.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V Le. 2019b. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *International Conference on Learning Representations (ICLR)*.

Passage & Question	Prediction
Complicated multi-step computation	
Passage: ... In the second quarter@N4, New Orleans regained the lead as QB Drew Brees (a former Charger) completed a 12 - yard@N5 TD pass to WR Devery Henderson (with a failed PAT) and RB Deuce McAllister getting a 1 - yard@N6 TD run. San Diego answered as QB Philip Rivers completed a 12 - yard@N7 TD pass to RB LaDainian Tomlinson, but the Saints replied with Brees completing a 30 - yard@N8 TD pass to WR Lance Moore. The Chargers closed out the half with Rivers completing a 12 - yard@N9 TD pass to TE Antonio Gates. In the third quarter@N10, New Orleans increased its lead Brees completing a 1 - yard@N11 TD pass to TE Mark Campbell ... San Diego tried to rally as Kaeding nailed a 31 - yard@N16 field goal, Rivers completed a 14 - yard@N17 TD pass to WR Vincent Jackson **Question:** How many more yards of touchdown passes did Drew Brees make than Philip Rivers?	**Prediction**: diff(add(N5,N8,N11,N17),add(N7,N9)) **Result:** 33 **Manual:** diff(add(N5,N11,N8),add(N7,N9,N17)) **Ground truth:** 5
Counting long strings	
Passage: ... They explain that the gap may persist due to the crack epidemic, the degradation of African - American family structure, the rise of fraud in the educational system (especially with respect to No Child Left Behind), the decrease in unskilled real wages and employment among African - Americans due to globalization and minimum wage increases, differences in parental practices (such as breastfeeding or reading to children), and " environmental conditions shaped by [African - Americans] themselves. ... **Question:** How many reasons are cited as causing the persistent gap between white and black IQs?	**Prediction**: count("crack epidemic","the degradation of African - American family structure","the rise of fraud in the educational system")" **Result:** 3 **Manual:** count("crack epidemic","the degradation of African - American family structure","the rise of fraud in the educational system (especially with respect to No Child Left Behind)","the decrease in unskilled real wages and employment among African - Americans due to globalization and minimum wage increases","differences in parental practices (such as breastfeeding or reading to children)","environmental conditions shaped by [African - Americans] themselves") **Ground truth:** 6
Questions with complex conditions	
Passage: ... yet the Raiders would answer with kicker Sebastian Janikowski getting a 33 - yard@N5 and a 30 - yard@N6 field goal. Houston would tie the game in the second quarter@N7 with kicker Kris Brown getting a 53 - yard@N8 and a 24 - yard@N9 field goal. ... **Question:** How many field goals did both teams kick in the first@Q0 half ?	**Prediction**: count(N5,N6,N8,N9) **Result:** 4 **Manual:** count(N5,N6) **Ground truth:** 2

Table 13: Examples of incorrect predictions by NSP in offline test of DROP.

Premise & Hypothesis	Program	Result	Ground Truth
Disturbance by redundancy numbers			
Premise: Melanie picked 7.0@M1 plums and 4.0@M2 oranges from the orchard and Sam gave her 3.0@M3 plums. **Hypothesis:** She has 10.0@N1 plums now.	**Prediction:** E: add(diff(M1,M2),M3)=N1 C: add(M1,M3)!=N1 **Manual:** E: add(M1,M3)=N1 C: add(M1,M3)!=N1	neutral	entailment
Premise: Sally had 39.0@M1 baseball cards , and 9.0@M2 were torn and Sara bought 24.0@M3 of Sally 's baseball cards. **Hypothesis:** Sally has 15.0@N1 baseball cards now.	**Prediction:** E: diff(add(M1,M2),M3)=N1 C: diff(add(M1,M2),M3)!=N1 **Manual:** E: diff(M1,M3)=N1 C: diff(M1,M3)!=M1	contradiction	entailment

Table 14: Examples of incorrect predictions by NSP on AWPNLI.

Function	Arguments	Description
add	a set of special tokens/programs	compute the summation of the terms
diff	a pair of special tokens/programs	compute the difference of the two terms
max min	a set of special tokens/programs	select the maximum/minimum among the terms
mul div	a pair of special tokens/programs	compute the multiplication/division of the two terms
avg	a set of special tokens/programs	compute the average of the terms
count	a set of special tokens/programs	count the number of the terms
year month day	a span in input text	convert the span to the corresponding year/month/day in numerical expression
hour minute second	a span in input text	convert the span to the corresponding hour/minute/second in numerical expression
= !=	a pair of special tokens/programs	return two terms are equal/not equal

Table 15: Description of the functions in the programs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. 2018. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. In *Proceedings of NAACL-HLT*, pages 1460–1469.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text.

Daniel Kahneman. 2003. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Sudip Mittal, Anupam Joshi, and Tim Finin. 2017. Thinking, fast and slow: Combining vector spaces and knowledge graphs. *arXiv preprint arXiv:1708.03310*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of EMNLP*.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Eduard Hovy. 2019. Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 349–361.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Swarnadeep Saha, Yixin Nie, and Mohit Bansal. 2020. Conjnli: Natural language inference over conjunctive sentences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8240–8252.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations (ICLR)*.

W Wang et al. 2017. R-net: machine reading comprehension with self-matching networks. natural language computer group, microsoft reserach. asia, beijing. Technical report, China, Technical Report 5.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

A Definition of functions in programs

The functions in programs are provided in Table 15.

B Annotation Process

The annotation team has ten annotators from a commercial company in data annotation. We sign a

contract with the company and pay the company for the annotation work at a market price in China. They are all college graduates with high capabilities in English and math. The required skill is the capability of understanding the texts in English and formulating the answering processes into programs. Both the QA and NLI datasets have the same annotation process. The annotation task is that given input texts with special tokens and the corresponding labels, the annotator: a) decides whether it needs logical reasoning to get the answers; b) if yes, labels the corresponding programs; if not, labels a [NULL]. We first conduct training for the annotators about the annotation rules. Then we separate the dataset into ten batches. Each annotator labels one batch of the dataset. We check 15% of samples in each batch. The whole batch will be relabeled if annotation accuracy is below 90%. Each data instance is labeled by only one annotator.

C Examples only NSP can predict correct results

Table 16 gives examples that only NSP can give correct results.

passage	question	prediction
mul & div & avg operators		
... The Dolphins ended the period with kicker Jay Feely getting a 53 - yard@N6 field goal. In the second quarter@N7, Miami drew closer as Feely kicked a 44 - yard@N8 field goal, yet New York replied with kicker Mike Nugent getting a 29 - yard@N9 field goal. ...	How many yards long does the average field goal measure when only the first@Q0 three@Q1 are taken into account ?	**Prediction**: div(add(N6,N8,N9),Q1) **Result:** 42 **Ground truth:** 42
... Dave Rayner nailed a 23-yard@N5 field goal ... Green Bay managed to get two@N8 more field goals , as Rayner got a 54-yarder@N9 and a 46-yarder@N10 to end the half ... David Akers got a 40-yard@N13 field goal ...	How many yards long was the average length across all field goals scored ?	**Prediction**: avg(N5,N9,N10,N13) **Result:** 40.75 **Ground truth:** 40.75
add & diff operators		
... Jermichael Finley caught a 20-yard@N6 touchdown pass from Aaron Rodgers ... The Lions responded with a Calvin Johnson 25-yard@N10 touchdown pass from Matthew Stafford ... The Packers then scored a touchdown when Randall Cobb caught a 22-yard@N13 pass from Aaron Rodgers ...	How many more yards of touchdown passes did Aaron Rodgers throw than Matthew Stafford ?	**Prediction**: diff(add(N6,N13),N10) **Result:** 17 **Ground truth:** 17
... Of 162@N1 cities worldwide, MasterCard ranked Bangkok as the top destination city by international visitor arrivals in its Global Destination Cities Index 2018@N2 ...	How many cities were n't ranked the top destination city by international visitor arrivals in the Global Destination Cities Index 2018@Q0	**Prediction**: diff(N1, count("Bangkok")) **Result:** 161 **Ground truth:** 161

Table 16: Examples of questions related to numerical reasoning on DROP, where only NSP can give correct program and final results.