NeurIPS | 2022 New Orleans. Nov 28-Dec 09, 2022
Thirty-sixth Conference on Neural Information Processing Systems

NL4Opt: Natural Language for Optimizing Modeling —— 3rd Place Solution

Jiayu Liu[1,2], Longhu Qin[1,2], Yuting Ning[1,2], Tong Xiao[1,2], Shangzi Xue[1,2], Zhenya Huang[1,2], Qi liu[1,2], Enhong Chen[1,2], Jinze Wu[2, 3]

[1]Anhui Province Key Lab. of Big Data Analysis and Application, School of Computer Science and Technology & School of Data Science, University of Science and Technology of China, [2]State Key Laboratory of Cognitive Intelligence, [3]iFLYTEK AI Research (Central China), iFLYTEK Co., Ltd.

Introduction

Background

- Many real-world decision-making problems can be formulated and solved as mathematical optimization problems.
- Modeling a problem into a proper formulation is a complex and time-consuming process.

Research Problem

- Can learning-based natural language interfaces be leveraged to convert linear programming (LP) word problems into a format that solvers can understand?
  - Challenge 1: detect problem entities from the problem description
  - Challenge 2: generate a precise meaning representation of the optimization formulation

Problem Definition

Subtask1

Given

- LP problem description:
  - $Q = \{w_0, w_1, ..., w_n\}$

Goal

- LP problem entities: $E = \{E_0, E_1, \dots, E_m\}$
  - $E_i = < I_i^{start}, I_i^{end}, c_i >$
  - $I_i^{start} \in [0, n]$, $I_i^{end} \in [0, n]$, $c_i \in C$

Subtask2

Given

- LP problem description: Q
- LP problem entities: E

Goal

- LP problem solution template
  $A = \{S_0^{const}, S_1^{const}, \dots, S_k^{const}, S^{obj}\}$
  - Declaration template : $S = \{s_0, s_1, \dots, s_n\}$

Subtask1 Summary

- **Pre-processing**
  - **Data Augmentation**
    - Swap variables, synonym replace for OBJ_NAME, randomly replace numbers
  - **Label Standardization**
    - Label inconsistency: The word times after numbers are sometimes regarded as a part of PARAM and LIMIT entity, but sometimes not.
    - Remove times after numbers in PARAM and LIMIT entity
- **Adversarial Training**
  - Use Projected Gradient Descent[1] to improve the robustness and generalization ability of the model
- **Ensemble Learning**
  - xlm-roberta-base
  - xlm-roberta-large
- **Post-processing**

Subtask2 Summary

- **Pre-processing**
  - Repeat CONST_DIR
- **Prompt Re-design**
  - Version1: Insert the prompt in the document
  - Version2: Entity-enhanced prompt
- **Encoder-decoder model**
  - **Model:** BART
  - **Adversarial Training**
    - A training method that introduces noise and improve the robustness and generalization ability of model
- **Intermediate representation**
  - **Ensemble**
    - **obj_declaration**: version2 prompt-guided input
    - **const_declarations**: version1 prompt-guided input + CONST_DIR repeat + adversarial training
  - **Post-processing**
    - **Rule1:** correct wrong operator direction
    - **Rule2:** correct wrong unseen numbers

Prompt Design

- **Pre-processing:**
  - Repeat CONST_DIR

Avian would like the commercials to be seen by **at least** 86 million young girls and 72 million middle-aged women.

- 2 LIMIT entities are related to 1 CONST_DIR.
- 2 declarations are generated by 1 CONST_DIR entity.

Preprocessing

Avian would like the commercials to be seen by **at least** 86 million young girls and **at least** 72 million middle-aged women.

- Repeat the CONST_DIR entity before the second LIMIT entity

- **Re-designed prompt**
  - Version1: Insert the prompt in the document
  - Version2: Entity-enhanced prompt

<CONST_DIR>at most</CONST_DIR><s>A bakery sells two types of cakes. They sell a chocolate cake and a vanilla cake. Let's say they make x1 chocolate cakes, at a profit of $10 each, and x2 vanilla cakes, at a profit of $11 each (x1 and x2 are unknowns both greater than or equal to 0). The daily demand for these cakes is at most 50 chocolate cakes and **at most** 30 vanilla cakes. The bakery is short staffed and can make a maximum of 50 cakes of either type per day. How much of each cake should the bakery make in order to maximize profit?</s>

Original prompt of baseline

Re-designed prompt

<s>A bakery sells two types of cakes. They sell a chocolate cake and a vanilla cake. Let's say they make x1 chocolate cakes, at a profit of $10 each, and x2 vanilla cakes, at a profit of $11 each (x1 and x2 are unknowns both greater than or equal to 0). The daily demand for these cakes is **<CONST_DIR> at most <CONST_DIR>** 50 chocolate cakes and **at most** 30 vanilla cakes. The bakery is short staffed and can make a maximum of 50 cakes of either type per day. How much of each cake should the bakery make in order to maximize profit?</s>

Version1: Insert the prompt in the document

<CONST_DIR> at most <CONST_DIR> <s>A bakery sells two types of cakes. They sell a <VAR> chocolate cake </VAR> and a <VAR> vanilla cake </VAR>. Let's say they make x1 <VAR> chocolate cakes </VAR>, at a <OBJ_NAME> profit </OBJ_NAME> of $<PARAM> 10 </PARAM> each, and x2 <VAR>vanilla cakes</VAR>, at a <OBJ_NAME> profit <OBJ_NAME> of $<PARAM> 11 </PARAM> each (x1 and x2 are unknowns both greater than or equal to 0). The daily demand for these cakes is <CONST_DIR> at most <CONST_DIR> <LIMIT> 50 </LIMIT>…

Version2: Entity-enhanced prompt

Framework

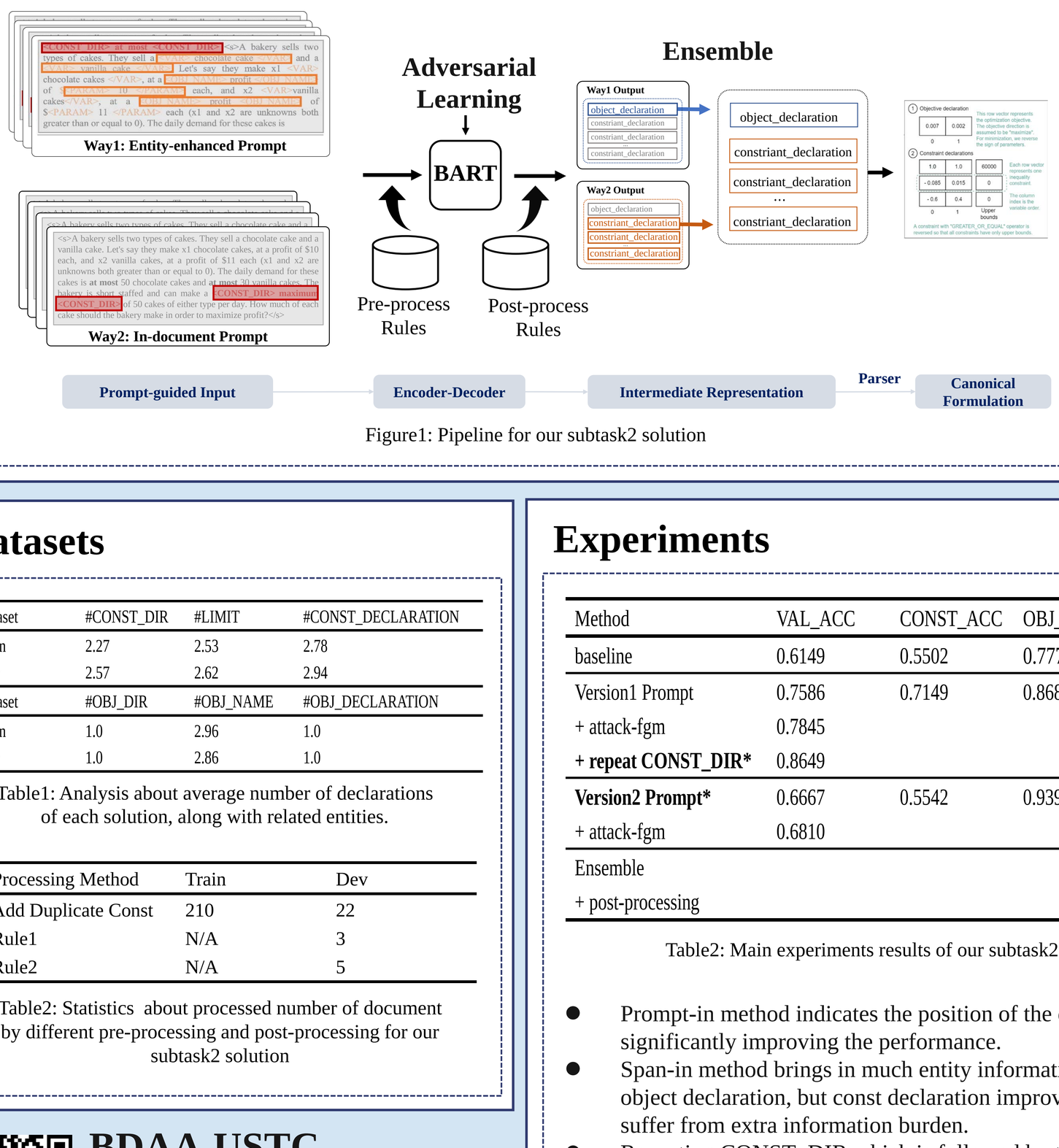

Figure1: Pipeline for our subtask2 solution

Datasets

Dataset	#CONST_DIR	#LIMIT	#CONST_DECLARATION
Train	2.27	2.53	2.78
Dev	2.57	2.62	2.94
Dataset	#OBJ_DIR	#OBJ_NAME	#OBJ_DECLARATION
Train	1.0	2.96	1.0
Dev	1.0	2.86	1.0

Table1: Analysis about average number of declarations of each solution, along with related entities.

Processing Method	Train	Dev
Add Duplicate Const	210	22
Rule1	N/A	3
Rule2	N/A	5

Table2: Statistics about processed number of document by different pre-processing and post-processing for our subtask2 solution

Experiments

Method	VAL_ACC	CONST_ACC	OBJ_ACC	TEST_ACC
baseline	0.6149	0.5502	0.7778	0.641
Version1 Prompt	0.7586	0.7149	0.8687	0.803
+ attack-fgm	0.7845			0.825
+ repeat CONST_DIR*	0.8649			**0.856**
Version2 Prompt*	0.6667	0.5542	0.9394	**0.708**
+ attack-fgm	0.6810			0.693
Ensemble				0.867
+ post-processing				**0.880**

Table2: Main experiments results of our subtask2 solution

- Prompt-in method indicates the position of the declaration, significantly improving the performance.
- Span-in method brings in much entity information, greatly improving object declaration, but const declaration improve little, which may suffer from extra information burden.
- Repeating CONST_DIR which is followed by two LIMIT alleviates the mismatch between the number of the ground truth and predicted declarations.

Conclusion

- We designed two powerful types of prompts. Experiments show that prompt design is important for text generation tasks, including what to prompt and where to prompt.
- We designed useful Post-processing and pre-processing methods based on mathematical knowledge, which directly correct some errors for model.
- We leveraged adversarial training to improve the robustness of the model, which helps for both Natural Language Understanding and Natural Language Generation.

Future work

- Data augmentation or more pre-processing rules.
- More attempts of prompt design.
- Constraint the prefix of intermediate representation.

BDAA USTC
Anhui Province Key Laboratory Of Big Data Analysis and Application