Named Entity-Based Enrichment with BART

Natural Language for Optimization Modeling (NL4Opt) Subtask 2

Neeraj Gangwar, Nickvash Kani
Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
306 N Wright St, Urbana, IL, USA
Email: gangwar2@illinois.edu, kani@illinois.edu

NEURAL INFORMATION PROCESSING SYSTEMS

Abstract

- In this poster, we present our approach for generating the precise meaning representation of linear programming problems.
- Our approach leverages the labeled semantic entities that are part of the input and uses BART to generate the meaning representations.
- We also discuss our model's sensitivity to optimization and compare the difference in using the base and the large versions of BART.

Introduction

- This task aims to generate the meaning representations of linear programming word problems. The meaning representations can be converted to a format that is understood by linear programming solvers.
- The input to the model consists of a word problem, labeled semantic entities, and order mapping of variable mentions.

INPUT

PROBLEM:
A berry picker must pick CONST_DIR at least LIMIT 3000 strawberries and LIMIT 15000 raspberries. He visits two farms. For each OBJ_NAME hour at VAR farm 1 he spends, he can pick PARAM 50 strawberries and PARAM 300 raspberries. For each OBJ_NAME hour at VAR farm 2 he spends, he can catch PARAM 70 strawberries and PARAM 200 raspberries. How many OBJ_NAME hours should he spend at each farm to OBJ_DIR minimize the OBJ_NAME amount of time he spends at both farms?

ORDER MAPPING:
farm 1: 0, farm 2: 1

OUTPUT

OBJECTIVE

```
{
  'type': 'objvar',
  'direction': 'minimize',
  'name': 'amount of time',
  'vars': ['farm 2', 'farm 1']
}
```

CONSTRAINTS

```
{
  'type': 'linear',
  'direction': 'at least',
  'limit': '3000',
  'terms': {'farm 2': '70', 'farm 1': '50'},
  'operator': 'GREATER_OR_EQUAL'
},
{
  'type': 'linear',
  'direction': 'at least',
  'limit': '15000',
  'terms': {'farm 1': '300', 'farm 2': '200'},
  'operator': 'GREATER_OR_EQUAL'
}
```

Dataset

- The dataset for this subtask consists of 1101 linear programming word problems, divided into the train, dev, and test splits consisting of 713, 99, and 289 problems, respectively.
- These problems are from advertising, investment, sales, production, science, and transportation domains. The training split contains problems only from the first three domains, whereas the dev and test splits contain problems from all six domains.

Output Formulation

ALGEBRAIC FORMULATION

OBJECTIVE:

$$\max x + y$$

CONSTRAINTS:

$$50x + 70y \geq 3000$$
$$300x + 200y \geq 15000$$

$\Longleftrightarrow$

CANONICAL FORMULATION

OBJECTIVE: [1, 1] (1)
0 1

CONSTRAINTS: [−50, −70, −3000] ; [−300, −200, −15000] (2)
0 1

1. Always maximize. For minimization, invert the signs.
2. The limits are upperbounds. For lowerbounds, invert the signs.

Our Approach

Our approach is built on top of the baseline method, which uses BART with Copy Mechanism for generation, with two modifications:

- The input is enriched to incorporate the named-entity information.
- The model outputs the objective and constraints at once.

Named Entity-Based Enrichment

Our approach adds tags around the named entities in the input (XML-like tagging with start and end tags). For example, the problem shown in the Introduction section would be enriched as shown below:

```
A berry picker must pick <CONST_DIR> at least </CONST_DIR> <LIMIT>
3000 </LIMIT> strawberries and <LIMIT> 15000 </LIMIT> raspberries. He
visits two farms. For each <OBJ_NAME> hour </OBJ_NAME> at <VAR> farm 1
</VAR> he spends, ... the <OBJ_NAME> amount of time </OBJ_NAME> he
spends at both farms?
```

Model Output Format

```
<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of time
</OBJ_NAME> [is] <VAR> farm 2 </VAR> [TIMES] <PARAM> ONE </PARAM><VAR>
farm 1 </VAR> [TIMES] <PARAM> ONE
</PARAM></DECLARATION><DECLARATION><CONST_DIR> at least
</CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 3000
</LIMIT><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> farm
2 </VAR> [TIMES] <PARAM> 70 </PARAM><VAR> farm 1 </VAR> [TIMES]
<PARAM> 50 </PARAM></DECLARATION><DECLARATION> ... </DECLARATION>
```

Experiments

Training Details

- We used BART-large as the pre-trained model and fine-tuned it on the NL4Opt subtask 2 dataset.
- We used one A100 GPU with 40GB VRAM to train the model for our final submission.
- The implementation was based on Transformers 4.3.0 and used PyTorch 1.14.0 (the development version).
- We used deterministic implementations of PyTorch methods along with a fixed seed to ensure reproducibility.

Metrics

The model is evaluated based on declaration-level accuracy.

$$\text{Accuracy} = 1 - \frac{\sum_{i=1}^{N} \text{FP}_i + \text{FN}_i}{\sum_{i=1}^{N} D_i}$$

where N is the number of examples. For i^{th} example, D_i is the number of ground truth declarations, FP_i is the number of non-matched predicted declarations, and FN_i is the number of excess ground truth declarations.

Results

Our best model achieved an accuracy of 0.874 with greedy decoding and 0.882 with a beam size of 5 on the validation set. On the test set, it achieved an accuracy of 0.899 with a beam size of 5.

Sensitivity to Optimization

The model achieved varied accuracy values when the training was initialized with different seeds. The fine-tuning was very sensitive to hyperparameter values including seeds with the large version of BART.

We ran our approach with pre-trained models BART-base and BART-large for 10 different seeds (5 common and 5 different seed values). Below are the accuracy values achieved by the model on the validation set with greedy decoding. We used one V100 GPU with 32GB VRAM for training these models.

BART-large	85.9	89.0	61.3	84.4	56.2	75.4	74.9	75.1	62.1	35.9
BART-base	80.8	81.5	76.2	80.8	79.2	81.5	81.0	79.7	81.2	80.8

It can be observed from the above table that BART-large has a standard deviation of 15.47 compared to 1.52 for BART-base.

References

[1] Mike Lewis et al. "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[2] Rindranirina Ramamonjison et al. "Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions". In: *arXiv preprint arXiv:2209.15565* (2022).

[3] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Code at github.com/mlpgroup/nl4opt-eq-generation