Named Entity-Based Enrichment with BART

NL4Opt Subtask 2, NeurIPS 2022 (nl4opt.github.io)

Neeraj Gangwar (gangwar2@illinois.edu)

Nickvash Kani (kani@illinois.edu)

Dec 7, 2022

University of Illinois at Urbana-Champaign

Task Overview

The goal of this task is to formulate the meaning representation of a linear programming word problem, given the problem description, labeled semantic entities, and the order mapping of variables.

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'
}
```

Output Formulation

JSON 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'
}
```

Order Mapping

farm 1: 0, farm 2: 1

$\Longleftrightarrow$

Algebraic Formulation

Objective:

$$\max x + y$$

Constraints:

$$50x + 70y \geq 3000$$

$$300x + 200y \geq 15000$$

$\Updownarrow$

Canonical Formulation

Objective:

0	1
1	1

Always maximize. Invert the sign in the case of minimization.

Constraints:

0	1	
-50	-70	-3000
-300	-200	-15000

Upperbounds. Invert the signs in the case of lowerbound.

Dataset

- The dataset for this subtask consists of 1101 linear programming word problems.
- The train, dev, and test splits contain 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.

Our Approach

Our approach is built on top of the baseline method, which uses BART with Copy Mechanism[1], 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

Add tags around the named entities in the input (XML-like tagging with start and end tags). For example, the problem shown earlier will be augmented 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?
```

[1]transformers.BartForConditionalGeneration (v4.3.0)

Model Output Format

The model output is in XML format, consisting of objective and constraint declarations.


```
<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>
```

Training Details

1. Used BART-large[2] as the pre-trained model and fine-tuned it on the subtask 2 dataset.
2. The model was trained on one A100 GPU with 40GB VRAM.

Metrics

The model was evaluated based on declaration-level accuracy.

Results

Our (submitted) 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 initialized with different seeds. The fine-tuning was very sensitive to hyperparameter values including seeds with the large version of BART.

[2] huggingface.co/facebook/bart-large

We ran our approach with pre-trained models BART-base and BART-large and 10 different seeds (5 common and 5 different seed values)[3]. Below are the accuracy values achieved by the model on the validation set with greedy decoding.

BART-large[4]	85.9	89.0	61.3	84.4	56.2	75.4	74.9	75.1	62.1	35.9
BART-base[5]	80.8	81.5	76.2	80.8	79.2	81.5	81.0	79.7	81.2	80.8

The standard deviation values for BART-large and BART-base are 15.47 and 1.52, respectively.

For these runs, only pre-trained models were switched between BART-base and BART-large. Other hyperparameters were kept the same.

[3] Ran on one V100 GPU with 32GB VRAM.

[4] huggingface.co/facebook/bart-large

[5] huggingface.co/facebook/bart-base

Challenges

As we used the baseline model as a starting point for our approach, we faced two major challenges with the baseline code.

Evaluation Batch Size

The validation part of the training failed if the evaluation batch size was increased from 1. This issue affected the speed of training, and the fix helped speed up our training pipeline.

Reproducibility

Even after fixing the seeds in the code, the results were different for different runs with the same seed values. The reason for this was the `scatter_add_` method that does not have a deterministic implementation in PyTorch 1.12 or earlier versions. We used PyTorch 1.14 (the dev version) and added `torch.use_deterministic_algorithms(True)` statement in addition to fixing the seeds. Note that this statement resulted in errors with versions 1.12 or before, due to not having a deterministic implementation of `scatter_add_`. It works with PyTorch 1.13 and later versions.

Following Shin et al. (2021)[6], we tried converting the output to a more natural text looking format.

```
<DECLARATION>The objective is amount of time which is defined as the sum of 1 times
farm 2, 1 times farm 1. minimize the objective.</DECLARATION><DECLARATION>Constraint
of type linear with direction at least is that sum of 70 times farm 2, 50 times farm 1
is GREATER_OR_EQUAL to 3000</DECLARATION><DECLARATION> ... </DECLARATION>
```

In our initial experiments, this approach did not show an improvement compared to the approaches that used XML-looking model output.

[6] Shin et al., “Constrained Language Models Yield Few-Shot Semantic Parsers”.

Code at /mlpgroup/nl4opt-eq-generation

Thank you for listening! Questions?