Explanation Generation Using ASP and Language Models: A Case Study in Smart Home Scheduling

Van Nguyen[1] and Tran Cao Son[1]

New Mexico State University
vnguyen,tson@cs.nmsu.edu

Abstract. The rapid growth of Artificial Intelligence (AI) applications gives rise to explainable AI research whose quest is to answer questions related to *why* and *how* a decision is made by an AI application. Besides the problem of identifying the answer to the question, an equally important problem is that of presenting the answer to the end-users who are often not familiar with the applications' terminologies and lingo. In this paper, we describe a process that supports the generation of natural language explanations via a case study in the smart home scheduling application. In this process, answer set programming (ASP) is used to identify contents of explanations and to generate dataset for the training of language model, which is used for the generation of natural language explanations.

Keywords: Explainable AI · Natural Language Generation · Answer Set Programming.

1 Introduction

In recent years, AI applications become more and more popular to nonexpert users and have been used in many areas that affect everyone's daily life. To protect individuals from harmful decisions made by AI applications, the right to (an) explanation regulation has been imposed (e.g., the EU GDPR law). Following this regulation, an user has the right to be given an explanation for an output of the AI algorithm. This issue has led to the rapid development of *Explainable AI* (see, e.g., surveys on explainable AI in [6, 12]) whose goal is to explain to users the decision of AI algorithms (or, more generally, computer softwares). To make the matter more complicated, the explanation should be understood by laymen. It is therefore desirable to present explanations in natural language to the users. In this paper, we describe a case study in generating natural language explanations for the smart home scheduling application ([9]).

2 Background: Smart Home Scheduling Problem and Language Models

We present a brief review of the *Smart Home Scheduling Problem* (SHSP) and some basics of language models. We assume that the readers are familiar with the basics of ASP.

2.1 Smart Home Scheduling Problems

In this section, we review the *Smart Home Scheduling Problem* (SHSP) with probabilistic user preferences, which is formulated in [9].

Definition 1 (Smart Home Scheduling Problem) *A* smart home scheduling problem *P is a tuple $\langle A, E, T, C, L, D \rangle$, where*

- *A is a set of **appliances** (or **devices**), usually written as the set of integers $\{1, \ldots, |A|\}$.*
- *$E = (e_1, e_2, ..., e_{|A|})$ is a vector of positive real numbers, where each e_i represents the **energy consumption** of device i.*
- *T is a set of **time slots**, usually written as the set of integers $\{1, \ldots, |T|\}$.*
- *$C = (c_1, c_2, ..., c_{|T|})$ and $L = (l_1, l_2, ..., l_{|T|})$ are vectors of non-negative real numbers, where c_i and l_i represent the **cost** of 1 kWh and the maximum **permissible load** of all the devices at time i, respectively.*
- *D is an $|A| \times |A|$ matrix, called **dependency matrix**, each cell $D(i, j)$ represents a hard constraint between devices i and j. The relations/constraints can be one of the following types: (i) **before** (resp. **after**) means that the device imust be turned on before (resp. after) device j; (ii) **parallel** (resp. **not-parallel**) means that the device i must run in parallel (resp. must not run in parallel) with the device j; and (iii) **nil** if the usage of the device i is independent from that of the device j.*

Without the loss of generality, it is assumpted that each device in P is turned on exactly once within $|T|$ time slots. A user preference for a scheduling problem is defined as follows.

Definition 2 (Probabilistic User Preference) *A* probabilistic user preference *over a scheduling problem $P = \langle A, E, T, C, L, D \rangle$ is a tuple $\mathscr{C} = \langle N, \alpha, \beta, \lambda \rangle$, where*

- *N is an $|A| \times |T|$ matrix, called **preference matrix**, where each cell $N(i, j)$ is a Normal distribution $\mathscr{N}(\mu_{ij}, \sigma_{ij})$ representing the probability distribution of the user's preference in turning the device i on at time slot j.*
- *α, called the **cumulative satisfaction threshold**, is a number representing the minimum acquired cumulative preference required by a user from a schedule.*
- *β is a number in the interval $[0, 1]$ representing the **probability threshold**, which indicates the threshold of the probability that α will be achieved given a schedule in order for a user to accept that schedule.*
- *λ is a number indicating the **cost threshold** that a user could accept.*

Scheduling problems with probabilistic preferences are defined next.

Definition 3 (Smart Home Scheduling Problem with Probabilistic User Preferences) *A* Smart Home Scheduling Problem *(SHSP) with probabilistic user preferences (or* p-scheduling problem, *for short) is a pair $(P, \mathscr{C})$, where P is a scheduling problem and $\mathscr{C}$ is a probabilistic preference over P.*

For brevity in this paper, SHSPs are referred to SHSPs with probabilistic user preferences.

2.2 Language Model

Models that assign probabilities to sequences of words are called language models (LM) [7]. Mathematically, a language model assigns probabilities to word sequences. The simplest model that assigns probabilities to sentences and sequences of words is the n-gram model. An n-gram is a sequence of n words. n-gram models can be used to estimate the probability of the last word of an n-gram given the previous words, and also to assign probabilities to entire sequences.

3 Preparing For Explanation: Content Generation

Given a SHSP problem $(P, \mathscr{C})$, we assume that our AI system (e.g., the scheduler described in [9]) generates a schedule for the devices in P that satisfies $\mathscr{C}$ (or optimal with respect to $\mathscr{C}$). For simplicity of the presentation, we will assume that the schedule consists of atoms of the form $on(d,t)$ (resp. $off(d,t)$), which says that device d should be turned on (resp. off) at timeslot t.

We identify the problem of explaining a schedule H to a user with the problem of explaining to a user that H meets the user's preferences, expressed in $\mathscr{C}$. In particularly, we focus on answering the two questions that a homeowner is normally interested in:

- **Q1**: "Why is a device d or a set of devices $\{d_1, \ldots, d_n\}$ scheduled to turn on at time t?", and
- **Q2**: "Why is device d scheduled at time t_1 instead of time t_2?".

In order to explain **Q1** to the user, the system needs to present the user with the preference indicating that d (or the set $\{d_1, \ldots, d_n\}$) should be turned on at t. On the other hand, it is best to explain **Q2** by either presenting some constraints in $\mathscr{C}$ that would be violated or indicating the non-optimality of a schedule if d is scheduled at t_2.

Given $\mathscr{C}$ and a schedule H, there are several ways to extract the information to explain **Q1**-**Q2** to a user. The ASP program π, shown in Listing 1.1, could be used for this purpose.

Table 1 shows the atoms, their matrix representations and their meanings in the program π.

Table 1. The atoms in the problem description

Atoms	Represent matrix	Meanings
$cost(T,D)$	cost C	the electrical cost at time T is D dollars per hour
$dev(ID,TYPE,KW)$	energy usage E	device ID consumes KW energy
$before(A,B)$, $parallel(A,B)$, $after(A,B)$, $nparallel(A,B)$	dependency D	order of the execution of device A and B
$pref(ID,T,\mu,\sigma)$	preference N	the user preference of executing device ID at time T is represented by the normal distribution $\mathscr{N}(\mu, \sigma)$

Program π identifies conditions in $\mathscr{C}$ that are satisfied (or not satisfied) by H. For convenience, we will write $\pi(\mathscr{C}, H)$ to denote the program π with the two inputs $\mathscr{C}$ and

H. Observe that *max_e*, *cost_threshold*, *alpha* and *beta* are constants in the program π (given in $\mathscr{C}$) representing parameters such as the maximal energy consumption acceptable by the user (*max_e*), etc. Lines 5-10 are for dependency between devices, Lines 12-13 for power safety, Lines 15-18 for user preference condition, and Lines 20-21 for the cost. The code block from line 23 to 28 computes the user preference condition. We encode the user preference condition as discussed in [9] .

Observe that $\pi(\mathscr{C}, H)$ is not a program that generates the schedule H. It is a positive program and has a unique answer set which contains information related to the preferences that are satisfied by H (e.g., $condibefore(d_1, d_2, t_1, t_2)$ is true if there exists a preference $before(d_1, d_2)$ and H contains $on(d_1, t_1)$ and $on(d_2, t_2)$ such that $t_1 < t_2$); or whether the cost threshold is satisfied (e.g., $costthr(s, cost\_threshold)$); etc. While it is sufficient to answer Question 1 by presenting the satisfied constraints related to the device d (or the set of devices $\{d_1, \ldots, D_n\}$) and the energy threshold is satisfied, Question 2 requires the identification of the differences between two potential schedules. In this paper, we will focus on the differences between the two answer sets of $\pi(\mathscr{C}, H)$ and $\pi(\mathscr{C}, H')$.

4 A Language Model For SHSP

To create a LM for SHSP, we use of Transformer [13] because models trained by transformers are shown to have competitive performances in natural language processing and understanding [4, 2, 8, 11], and computer vision [10, 3, 5].

We take inspiration from [1] to create data for training of our language model using ASP. We generated data from all possible atoms in the program π. Each training example is generated from an atom or a set of atoms, and is a string of format

$$[\texttt{CLS}]\texttt{part1}[\texttt{SEP}]\texttt{part2}[\texttt{SEP}], \tag{1}$$

where `CLS` token indicates the result of classification tasks, which we do not use in our case; `SEP` token indicates a separation; `part1` is a description of an atom or a list of atoms which carries a meaning; and `part2` is the natural language sentence for the meaning of the atom or the list of atoms described in `part1`.

For instance, a training example generated for the atom $on(1, 20)$ is the string "$on(1, 20)$ has parameters 1 20 turn on<s>Device 1 is on at timeslot 20".

The string "<s>" represents the `SEP` token. A training example generated for a list of atoms can be "weight(1,1,5,5) weight(2,1,7,5) energythr(12,2,31)<s>At time slot 1, 2 devices turn on. The total energy is 12 kw.".

Table 2 shows the atoms in the program π and their corresponding natural language expressions. For simplicity of the presentation, we show only the atom $condibefore$, representing the $before$ dependency, in Table 2.

We use an ASP program to generate the atoms as in Table 2, and map them with their meaning in natural language to create synthetic data for training our language model. Given the `part1`, the fine-tuned LM will generate the meaning of the atom(s) described in the `part1`.

To evaluate the usability of GPT-2[SHSP], we evaluate it by computing its BLEU score (automatic) as well as conducting a small scale user study.

Table 2. The atoms of the program π and their meaning in natural language

Atoms	Meanings
$on(D,T)$	Device D is on at timeslot T.
$weight(D,T,E,C)$	Device D at timeslot T consumes E kWh and cost at that timeslot is C cents per hour.
$costthr(C,THR)$	The total cost of the schedule is C and is less than THR.
$energythr(T,E,THR)$	Energy consumption at timeslot T is E and is less than THR.
$condibefore(D1,D2,T1,T2)$	Device $D1$ turns on at $T1$ before device $D2$ at $T2$.
$condipref(PREF,SS)$	The estimated preference of the schedule is $PREF$ and SS.

BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality). Following the rough interpretation of BLEU score[1], a score that is larger than 0.3 can be considered understandable to good translations. In evaluating GPT-2[SHSP], we consider the training data as the reference translations and the generated text from GPT-2[SHSP] as the machine-translated text. The average score for our LM is 0.33, which shows a relatively acceptable level of translations in comparison with the synthetic data.

We also perform a small scale user study, in which, we sent out a survey to a group of seven students in a class and asked them to evaluate the explanations in terms of "how well they can be understood?". The survey is available online [2]. Overall, the reponses [3] shows that the participants could grasp the meaning of the texts generated by GPT-2[SHSP], and there is no complaint in grammar or misuse of sequences of words in the answers.

In summary, the BLEU score and the conducted experiments show that the explanations generated using GPT-2[SHSP] are meaningful and can be understood by human users.

5 Conclusions

In this paper, we investigate the use of language models in generating natural language explanations in AI applications and consider the smart home scheduling problem as a case study. We show that we can address an important problem in training a language model, the problem of creating training data, by using ASP. By fine-tuning an existing language model, we receive a language model that can generate explanations for the smart home scheduling problem.

[1] https://cloud.google.com/translate/automl/docs/evaluate
[2] https://ida3rxjl4si.typeform.com/to/GYJX9Yc9
[3] https://tinyurl.com/h82zp3a4

References

1. Banerjee, P., Baral, C., Luo, M., Mitra, A., Pal, K., Son, T.C., Varshney, N.: Can transformers reason about effects of actions? (2020)
2. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 1–42 (2018)
7. Keselj, V.: Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6, $115.00 (2009)
8. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
9. Nguyen, V., Yeoh, W., Son, T.C., Kreinovich, V., Le, T.: A scheduler for smart homes with probabilistic user preferences. In: International Conference on Principles and Practice of Multi-Agent Systems. pp. 138–152. Springer (2019)
10. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: International Conference on Machine Learning. pp. 4055–4064. PMLR (2018)
11. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
12. Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.R.: Explainable AI: interpreting, explaining and visualizing deep learning, vol. 11700. Springer Nature (2019)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)

A Program for Computing Contents of Explanations

Note: This program is included only for review.

Listing 1.1. Program π: Computing What to Say?

```
max_energy(max_e).
device(D)    :- dev(D,_,_). timeslot(X) :- cost(X,_).
weight(D,T,E,M) :- on(D,T), dev(D,_,E), cost(T,M).

condibefore(D1,D2,T1,T2):-before(D1,D2),on(D1,T1),on(D2,T2),T1<T2.
condiafter(D1,D2,T1,T2) :-after(D1,D2), on(D1,T1),on(D2,T2),T1>T2.
condiparallel(D1,D2,T1)     :- parallel(D1,D2),on(D1,T1),
                               on(D2,T2),T1 = T2.
```

```
condinparallel(D1,D2,T1,T2):- nparallel(D1,D2),on(D1,T1),
                              on(D2,T2),T1 != T2.

energythr(S,T,TH) :- S = #sum {E,D: weight(D,T,E,M)},
                     timeslot(T), max_energy(TH), S < TH.

schedule_pref(PR,S) :- PR = #sum{P,D,T:pref(D,T,P,STD),on(D,T)},
                       S = #sum{STD,D,T:pref(D,T,P,STD),on(D,T)}.
user_pref(P,S,@pref_con(P,S,alpha,beta)) :- schedule_pref(P,S).
condiuser(PREF,SS) :- user_pref(PREF,SS,C), C = 1.

costthr(S, cost_threshold) :- S = #sum {E*M,D,T: weight(D,T,E,M)},
                              S < cost_threshold.

#script (python)
import clingo
def pref_con(pref,ss,alpha,beta):
  cdf_value = cdf((alpha - pref) / ss)
  return cdf_value * 100 <= 100 - beta
#end.
```