# Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language

Parag Pravin Dakle[1], Serdar Kadıoğlu[1,2][0000−0002−4672−6830], Karthik Uppuluri[1], Regina Politi[1], Preethi Raghavan[1], SaiKrishna Rallabandi[1], and Ravisutha Srinivasamurthy[1]

[1] AI Center of Excellence, Fidelity Investments, Boston, USA
[2] Dept. Computer Science, Brown University, Providence, USA
{firstname.lastname}@fmr.com

**Abstract.** Solving combinatorial optimization problems involves a two-stage process that follows the model-and-run approach. First, a user is responsible for formulating the problem at hand as an optimization model, and then, given the model, a solver is responsible for finding the solution. While optimization technology has enjoyed tremendous theoretical and practical advances, the overall process has remained the same for decades. To date, transforming problem descriptions into optimization models remains a barrier to entry. To alleviate users from the cognitive task of modeling, we study named entity recognition to capture components of optimization models such as the objective, variables, and constraints from free-form natural language text, and coin this problem as Ner4Opt. We show how to solve Ner4Opt using classical techniques based on morphological and grammatical properties and modern methods leveraging pre-trained large language models and fine-tuning transformers architecture with optimization-specific corpora. For best performance, we present their hybridization combined with feature engineering and data augmentation to exploit the language of optimization problems. We improve over the state-of-the-art for annotated linear programming word problems, identify several next steps and discuss important open problems toward automated modeling.

**Keywords:** Optimization Modeling · Named Entity Recognition · Natural Language Processing

## 1 Introduction

Optimization technology spans a wide range of applications, and over the years, combinatorial optimization solvers have enjoyed significant speed-ups [29]. In parallel, several high-level modeling languages (e.g., [18,39,59]) are designed to improve the accessibility of this powerful technology. Still, the overall process of modeling and solving optimization problems remained the same for decades. The de facto approach is to follow the *model-and-run* strategy where the user is responsible for transforming the problem at hand as an optimization model, and then, given the model, a solver is responsible for finding the solution.

We envision an automated modeling assistant to help turn natural language into optimization formulations. To realize this future state, there is a necessary precursor: given the problem description of an optimization problem, finding key pieces of information to enable model formulation. This is exactly what we study in this paper; NER4OPT, the challenge of named entity recognition for extracting optimization-related information such as the objective, constraints, and variables from free-form natural language text. With this goal in mind, in this paper, we make the following contributions:

1. We formalize NER4OPT as an interdisciplinary problem at the intersection of Natural Language Processing and Combinatorial Optimization. We discuss how it differs from the standard named entity recognition (NER) (§2) in important ways stemming from the optimization context.
2. We start with baseline lexical solutions built on classical techniques commonly used in NER (§3.1) and then study the impact of recent advances in pre-trained large language models for building semantic solutions (§3.2).
3. We show how to combine the lexical and semantic models as a hybrid approach and propose several augmentation techniques including fine-tuning language models using optimization textbooks to achieve the best performance (§3.4).

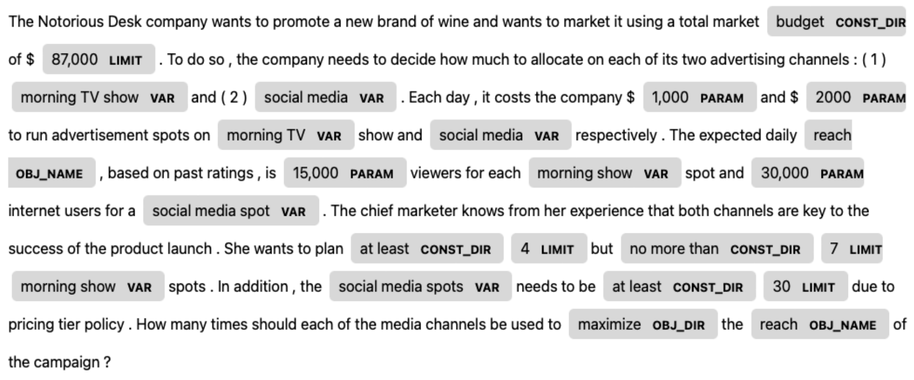
Our key finding is that generalization for NER4OPT is possible. We learn from annotated optimization problems emerging in advertising, investment, and sales optimization as the source domain that is then tested on production, science, and transportation optimization as the target domain.

For computational experiments (§4), we consider the recently introduced linear programming word problems as the benchmark (§4.1). We improve over the best-known solutions on this dataset [51]. To foster further research, we release our code and demo that extracts optimization-related entities from input text [3]. More importantly, we show that it is possible to train effectively for generalization not only to new problem instances of the same domain but also to new applications. Our work is necessary but not sufficient for automated modeling assistants, and accordingly, we discuss important next steps and remaining open problems.

## 2 Problem Description

Let us start with a formal definition of the NER4OPT problem which can be viewed as an instantiation of the classical NER problem [17,58,10] with its particular challenges emerging from the optimization context.

**Definition 1 (Named Entity Recognition for Optimization (NER4OPT)).** *Given a sequence of tokens $s = \langle w_1, w_2, \cdots, w_n \rangle$, the goal of* NER4OPT *is to output a list of tuples $\langle I_s, I_e, t \rangle$ each of which is a named entity specified in $s$. Here, $I_s \in [1, n]$ and $I_e \in [1, n]$ are the start and the end indexes of a named entity mention; $t$ is the entity type from a predefined category set of constructs related to optimization.*



The Notorious Desk company wants to promote a new brand of wine and wants to market it using a total market budget CONST_DIR of $ 87,000 LIMIT . To do so , the company needs to decide how much to allocate on each of its two advertising channels : ( 1 ) morning TV show VAR and ( 2 ) social media VAR . Each day , it costs the company $ 1,000 PARAM and $ 2000 PARAM to run advertisement spots on morning TV VAR show and social media VAR respectively . The expected daily reach OBJ_NAME , based on past ratings , is 15,000 PARAM viewers for each morning show VAR spot and 30,000 PARAM internet users for a social media spot VAR . The chief marketer knows from her experience that both channels are key to the success of the product launch . She wants to plan at least CONST_DIR 4 LIMIT but no more than CONST_DIR 7 LIMIT morning show VAR spots . In addition , the social media spots VAR needs to be at least CONST_DIR 30 LIMIT due to pricing tier policy . How many times should each of the media channels be used to maximize OBJ_DIR the reach OBJ_NAME of the campaign ?

**Fig. 1.** NER4OPT Example: Given the problem description in free-form natural language text, the goal is to extract key information about the variables, parameters, constraint direction, limits, objective, and optimization direction.

Figure 1 illustrates our problem definition with an example. Given the problem description, the goal of NER4OPT is to extract constraints, parameters, variables, and the objective, among others.

Regarding entities, when NER was first defined in MUC-6 [17], the task was to recognize names of people, organizations, locations, time, currency, and percentage expressions in the text. As shown in our example, the predefined optimization entities in this paper are constraint direction (CONST_DIR), limits (LIMIT), objective direction (OBJ_DIR), objective name (OBJ_NAME), parameter (PARAM), and variable (VAR).

Regarding downstream applications, NER acts as an essential pre-processing step for information retrieval, question answering, and machine translation. Here, we leverage it as the precursor to automated modeling where NER4OPT treats the input description as a word problem that describes decision variables, the objective, and constraints. However, this multi-sentence word problem exhibits a high level of ambiguity due to the variability of the linguistic patterns, problem structure, and application domain.

Regarding NLP tasks, the NER4OPT differs from the NER problem in several challenging ways. First, optimization technology is a general-purpose tool that can tackle a wide range of applications. Accordingly, when parsing problem descriptions, the solution for NER4OPT must be domain-agnostic and generalize to new instances and applications. Second, given the complexity of building optimization models, we only have access to limited training data. Unlike many NLP tasks, we cannot even depend on human annotators since it requires modeling expertise. Therefore, we must rely on large-scale domain knowledge and data augmentation methods to train robust models in low-resource settings. Finally, while most existing parsers operate at the sentence level, optimization descriptions span long text inputs to describe variables and constraints with a high degree of compositionality and ambiguity.

[3] https://huggingface.co/spaces/skadio/ner4opt

## 3 Our Approach

We follow three main directions to address the NER4OPT problem: classical NLP approaches (§3.1), followed by recent advances in modern language models (§3.2), and their hybridization thereof (§3.4) together with data augmentation techniques (§3.3) to achieve the best performance.

### 3.1 Classical NLP

A standard approach in the literature to solve the NER problem is feature engineering coupled with a structured prediction model such as linear chain Conditional Random Field (CRF) [30,44]. This is what we start with as our baseline.

**Overview of CRF** As shown in Figure 2, given an input sequence of tokens $\mathrm{x_i}$ and a set of engineered feature extraction functions $\mathrm{f_j}$ at each token position, a conditional random field models a probability distribution of labels $\mathrm{y_i}$ that can be assigned to appropriate segments in x.

$$score\left(y|x\right) = \sum_{j=1}^{m}\sum_{i=1}^{n} w_j f_j\left(x, i, y_i, y_{i-1}\right) \tag{1}$$

$$p(y|x) \;=\; \frac{\exp^{score(y|x)}}{\sum_{y'} \exp^{score(y'|x)}} \tag{2}$$

Given a set of training examples $D$, a CRF finds an optimal label assignment using maximum likelihood. Here, $w$ is the weight vector and $C$ is the regularization parameter.

$$D = \left[\left(x_1, y_1\right), \left(x_2, y_2\right), \left(x_3, y_3\right), \ldots, \left(x_d, y_d\right)\right] \quad i.i.d\ training\ examples \tag{3}$$

$$L(w, D) \;=\; -\sum_{k=1}^{d} log\left[p(y^k|x^k)\right] \tag{4}$$

$$w^* \;=\; \arg\min_{w}\; L(w, D) \;+\; C\,\frac{1}{2}||w||^2 \tag{5}$$

**Feature Extraction** In NLP, a feature extraction function explores the linguistic properties of a token or a group of tokens. For NER, different classes of properties, including grammatical (e.g., part-of-speech tagging and dependency relations), morphological (e.g., prefix, suffix, and word shape), vocabulary (e.g., gazetteers) and syntactic (noun phrases and prepositional phrases) are often used. For details on features used for NER, we refer to [52]. In addition to commonly used feature extraction functions, we engineered other features inspired by the structural characteristics of the optimization problems, as detailed next.

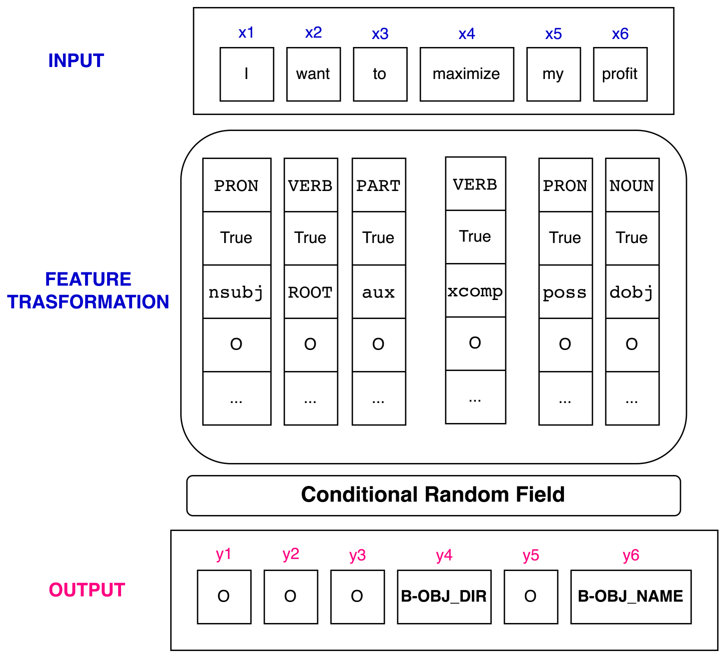

INPUT
x1 x2 x3 x4 x5 x6
I want to maximize my profit
FEATURE TRASFORMATION
PRON VERB PART VERB PRON NOUN
True True True True True True
nsubj ROOT aux xcomp poss dobj
O O O O O O
... ... ... ... ... ...
Conditional Random Field
OUTPUT
y1 y2 y3 y4 y5 y6
O O O B-OBJ_DIR O B-OBJ_NAME

**Fig. 2.** NER4OPT CRF Example: Given the input sentence, feature extraction and transformation of each token is fed into the conditional random field to find the output of recognized entities.

**Gazetteer Features:** Gazetteers serve as lookup tables and are utilized as noisy priors to entity labels. These are especially useful when the entity class has frequent keywords and phrases. These key phrases are extracted from the training data. In our optimization setting, canonical keywords include `maximize OBJ_DIR` and `minimize OBJ_DIR`, and similarly, `at least CONST_DIR` and `at most CONST_DIR`.

**Syntactic Features:** In linguistics, a *conjunct* is a group of tokens joined together by conjunction or appropriate punctuation. The (VAR) and (OBJ_NAME) entities are associated with unique syntactical properties in the form of conjuncts. We observe four patterns for the (VAR) entity. First, these are often conjuncting noun chunks, e.g., "`a factory produces rice VAR and corn VAR`". These entities also appear as conjuncting prepositional chunks, e.g., "`there are two types of cars: cars with automatic gear VAR and cars with manual gear VAR`". The other patterns include conjuncts connected by a hyphen or a quote. For the (OBJ_NAME), we observe that (OBJ_DIR) appears in the context of defining the objective of the problem. Moreover, frequently, an (OBJ_DIR) is followed by a noun chunk denoting the (OBJ_NAME) often qualified by an adjective, verb, or prepositional phrase. To succinctly capture these feature extraction heuristics, we design an automaton as depicted in Figure 3.

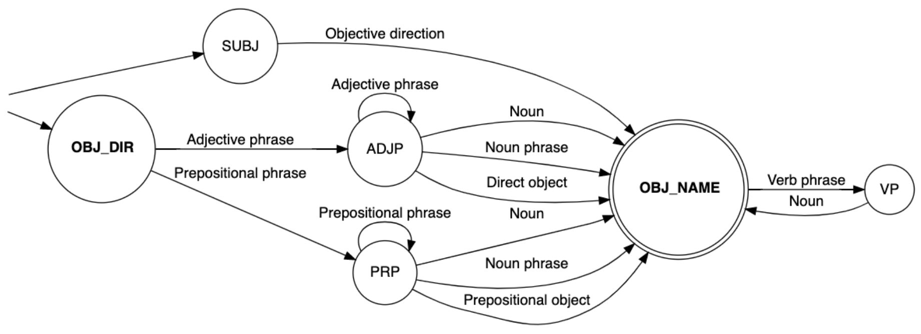


**Fig. 3.** Regular automaton (sketch) to capture features for (OBJ_NAME) extraction.

The regular membership with respect to this automaton in Figure 3 enables us to extract the (OBJ_NAME). For example, "`profit SUBJ to be maximized OBJ_DIR`, and similarly, "`maximize OBJ_DIR the total monthly ADJP profit NOUN`" are valid in this language, and the `profit` is extracted as the objective.

**Contextual Features:** All the previous hand-crafted features operate at token level. In addition, we extract left and right contextual features around each token with window size, $w$. The parameter $w$ is learned from the training data based on the longest entity phrase. These feature extraction functions act as noisy priors to each entity label. The CRF model relies on many such features and their respective contexts; hence, a few false positive features will not affect the model's overall performance. We studied other features engineering methods, including constituent parsing, quantized word embeddings, and word-frequency-based features that are omitted here for brevity.

### 3.2 Modern NLP

So far, our solution for Ner4Opt only considered classical methods based on feature extraction and manual feature engineering. This helps us establish a baseline performance. The challenger to this baseline is motivated by the recent advances in NLP, offering advantages over traditional techniques. Specifically, deep neural networks alleviate the need for manual feature extraction. At a high level, Equation 1 continues to apply whereby feature vectors $f(x)$ now correspond to dense embeddings retrieved from large language models. This not only saves a significant amount of time in creating features but offers more robust behavior. Moreover, the nonlinearity in the activation functions enables learning complex features and dependencies from the labeled training data.

In practice, Ner problems require modeling long-range text dependencies. When operating on the long-range, recurrent architectures are known to struggle with vanishing and exploding gradients [42]. As a remedy, most recent works rely on the Transformers architecture [60] that solve the long-range problem by replacing the recurrent component with the *attention* mechanism. There are many variants of this architecture, and here, we consider three distinct flavors based on RoBERTa [34] to generate the features vectors $f(x)$ used in CRF.

1. XLM-RB: The XLM-RoBERTa [11] is a self-supervised language model that follows the RoBERTa architecture with multilingual training. This is the state-of-the-art method [51] on the benchmark dataset we consider. One of our goals is to improve this existing approach and its large version, XLM-RL.
2. XLM-RL+: Our unique contribution that extends XLM-RL with fine-tuning over corpora related to optimization texts. We explain this in detail below.
3. ROBERTA: Another large language model that uses the same transformers architecture as BERT [13] and improves it with more robust training [34]. It achieves state-of-the-art results on well-known NLP benchmarks such as GLUE [61], RACE [31] and SQuAD [50,49]. As such, we consider it here and employ its large version.

**[XLM-RL+] Fine-Tuning on Optimization Textbooks:** Language models such as BERT, RoBERTa, and GPT [45] are pre-trained on non-domain specific texts where the goal of pre-training is to obtain good downstream performance on a diverse set of *language-oriented tasks* (e.g., sentiment analysis). The training is carried out in a self-supervised fashion via masked language modeling, next sentence prediction [13], and causal language modeling [45]. For *domain-specific* tasks (e.g., sentiment analysis in finance), performance can be improved further using domain-specific corpora to fine-tune pre-trained models [22,1,32,5].

We use a similar approach for fine-tuning XLM-RL leading to XLM-RL+. For that purpose, we source three publicly available optimization textbooks. The first is the well-known convex optimization book by S. Boyd [8]. The second one is about linear programming and game theory [57]. And finally, we consider the course notes on optimization [20] from the Open Optimization Platform[4] that shares educational content. We extract textual data from the PDFs of these textbooks, and then, fine-tune XLM-RL via masked language modeling. More precisely, we mask 15% of the words at random and replace 80% of the masked words with the MASK token, 10% with random words, and the remaining 10% with the original word. Finally, the model is trained in self-supervised fashion to predict the masked words.

## 3.3 Data Augmentation

In parallel to modeling strategies, we also consider two methods for data augmentation to improve performance. First, we show how to find infrequent patterns to over-sample, and second, we introduce a simple yet effective technique, coined L2 augmentation, to disambiguate the objective variable from other variables.

**Dealing with Infrequent Patterns:** Over-sampling is an effective technique, especially when dealing with class imbalance. While the distribution of entity classes might not be imbalanced, the lexical features might exhibit popular traits with a few infrequent features. For example, the objective direction is almost always `maximize` or `minimize`. Yet, in a few cases, it is given as an adjective, e.g., "`cost to be minimal ADJ`". The challenge is to find a way to surface such infrequent cases without manual inspection so that we can over-sample the dataset.

[4] https://github.com/open-optimization

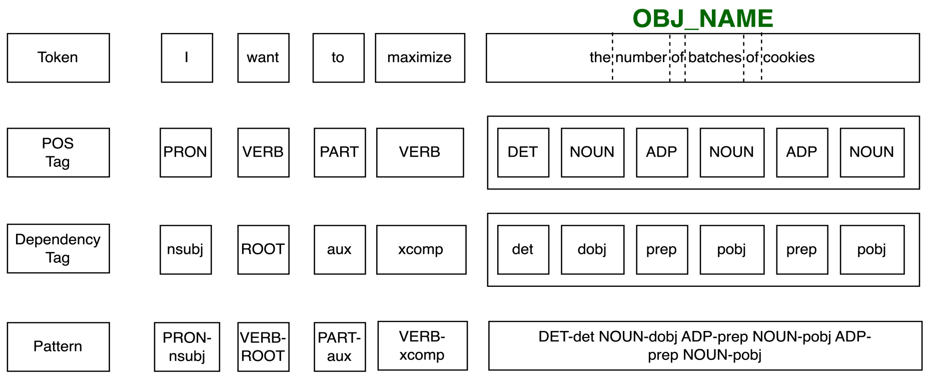


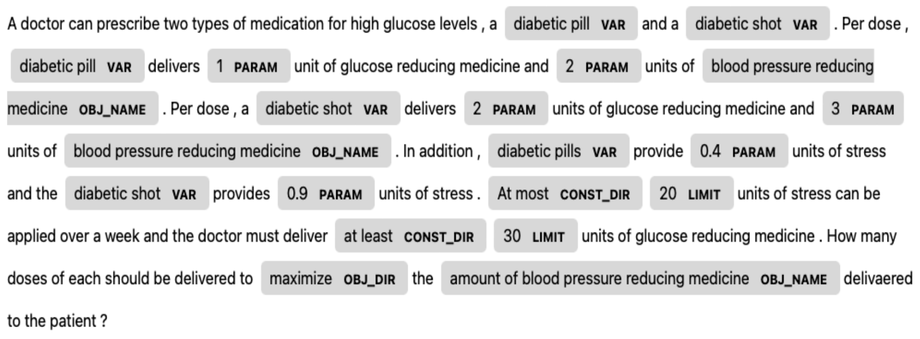
**Fig. 4.** Example pattern for the given objective name entity as the union of its part-of-speech and dependency tags.

We propose a simple approach as shown in Figure 4. First, we extract part-of-speech and dependency tags for each token in a given sentence and consider their union as a *pattern*. We then build the set of unique patterns and map them to entity labels in the training data to find their occurrence counters. Consequently, problem descriptions with infrequent patterns are duplicated.

**Dealing with Disambiguation:** There is a severe ambiguity between the objective variable and other variables. After all, the objective is yet another variable, only with an optimization direction. The critical question is how to train a model to differentiate between the two effectively. Consider the scenario in Figure 5. In this case, the objective is blood pressure reducing medicine, and there is no distinctive feature that separates it from other variables such as diabetic pill and diabetic shot. Even for the human annotator or the optimization expert, the objective remains unknown until the last sentence. Despite that, the model must label the objective correctly as early as the second sentence in its first occurrence. This is precisely the long-range text dependency aspect of Ner4Opt. To combat this, we append the beginning of each problem description with the last two sentences and refer to this method as L2 augmentation.

## 3.4 Hybrid Modeling

Finally, we consider a hybrid approach that combines our proposed methods. Classical methods for feature engineering and modern techniques for feature learning have their strengths and weakness. While feature engineering can sometimes be brittle, feature learning struggles when there is long-range dependency and no semantic theme, as in the (OBJ_NAME). On the other hand, handcrafted features, such as our gazetteer, syntactic and contextual features, and knowledge injection, such as our purpose-built automaton, allow us to build apriori information. Our hybrid model uses the classical CRF approach boosted by feature engineering plus an additional feature provided by the prediction of a transformers-based model fine-tuned on optimization corpora with over-sampling and L2 data augmentation.



A doctor can prescribe two types of medication for high glucose levels , a diabetic pill VAR and a diabetic shot VAR . Per dose , diabetic pill VAR delivers 1 PARAM unit of glucose reducing medicine and 2 PARAM units of blood pressure reducing medicine OBJ_NAME . Per dose , a diabetic shot VAR delivers 2 PARAM units of glucose reducing medicine and 3 PARAM units of blood pressure reducing medicine OBJ_NAME . In addition , diabetic pills VAR provide 0.4 PARAM units of stress and the diabetic shot VAR provides 0.9 PARAM units of stress . At most CONST_DIR 20 LIMIT units of stress can be applied over a week and the doctor must deliver at least CONST_DIR 30 LIMIT units of glucose reducing medicine . How many doses of each should be delivered to maximize OBJ_DIR the amount of blood pressure reducing medicine OBJ_NAME delivaered to the patient ?

**Fig. 5.** The challenge of long-range text dependency in NER4OPT. Notice the disambiguation problem between the objective variable and other variables and the importance of the last sentences in capturing the goal of the problem description.

## 4 Experiments

To demonstrate the effectiveness of our approach when solving the NER4OPT problem in practice, we consider the following specific questions:

**Q1:** What is the baseline performance of classical methods (§3.1) and does feature engineering help?

**Q2:** How do modern NLP methods (§3.2) perform, how do we fare against the best-known solutions on the same dataset, and do we improve the state-of-the-art for NER4OPT?

**Q3:** Does the hybrid model (§3.4) that combines feature engineering with feature learning and augmentation perform better than its counterparts in isolation?

Let us start with an overview of the dataset, the experimental setup, and evaluation metrics and present numerical results with discussions and error analysis.

### 4.1 NER4OPT Dataset

We use linear programming word problems that are released as part of the NeurIPS'22 natural language for optimization challenge[5]. We are indebted to the organizers for contributing such a rich dataset to the community. Our formal definition of NER4OPT corresponds to Task–I from this challenge[6]. This dataset was first introduced in [51], which uses the XLM-RB model to solve the entity recognition problem. It contains 1101 linear programming word problems of the form:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{a}_i^\top \mathbf{x} \leq \mathbf{b}_i, \quad i = 1, ..., m \tag{6}$$

where $\mathbf{c}$ represents the parameters of the objective, $\mathbf{a}_i$ the $i$-th constraint, and $\mathbf{b}_i$ is the right-hand-side limit. The goal of the linear problems is to find $\mathbf{x}$ that minimizes the objective value.

[5] https://github.com/nl4opt

[6] https://github.com/nl4opt/nl4opt-subtask1-baseline

| Statistic | Value |
|---|---|
| Dataset size | 1101 |
| Train set size | 713 |
| Dev set size | 99 |
| Test set size (not available) | 289 |
| Number of entity types | 6 |
| Number of VAR entities | 5299 |
| Number of PARAM entities | 4113 |
| Number of LIMIT entities | 2064 |
| Number of CONST_DIR entities | 1877 |
| Number of OBJ_DIR entities | 813 |
| Number of OBJ_NAME entities | 2391 |

**Table 1.** The benchmark dataset with 1101 samples annotated with six entities.

Table 1 shows dataset statistics. The problems in the dataset belong to six domains grouped into two: the source domain comprising of problems from advertising, investment, and sales, and the target domain consists of problems from production, science, and transportation. As in our problem description (§2), it contains annotations for six entity types: variable (VAR), parameter (PARAM), limit (LIMIT), constraint direction (CONST_DIR), objective direction (OBJ_DIR) and objective name (OBJ_NAME). The test set is not public, hence we focus on train and dev sets and cannot directly compare with Task-I.

The training set consists of samples only from the source domain, whereas the dev and test sets consist of samples from both source and target domains in a 1:3 source-to-target domain ratio. According to [51], 15 annotators created the problem descriptions and the labels while 4 additional NLP/OR experts annotated more than 10% of the entire dataset to compute inter-annotator agreement. Average pairwise micro-averaged F1 score was used to measure the agreement and a score of 97.7% was reported. Figure 1 presents an annotated input sample.

## 4.2 Comparisons

We compare the classical, modern, and hybrid models with the following variants:

1. Classical: Our classical method (§3.1) based on grammatical and morphological features.
2. Classical+: Our classical method plus our hand-crafted gazetteer, syntactic, and contextual features.
3. Xlm-rb: The state-of-the-art method on this dataset from [51] that we reran thanks to authors' code. We also consider its large version, Xlm-rl.
4. RoBERTa: Transformers model with good default performance across several language tasks for comparison. We use its large model variant.
5. Xlm-rl+: Our approach (§3.2) to fine-tune Xlm-rl with optimization books.
6. Hybrid: Our hybrid approach (§3.4) that combines Classical+ with Xlm-rl+ and data augmentation.

### 4.3 Experimental Setup

We use the train set for learning the model weights and the dev set for testing the performance for all the methods considered. We conduct limited parameter tuning to avoid overfitting. We leverage HuggingFace [62] and SimpleTransformers [48] for transformer models, spaCy [21] for part-of-speech and dependency tagging, and sklearn-crf[7] for the CRF model.

**Hyperparameters for CRF** There are four hyper-parameters for CRF: $c1$ & $c2$ controlling the amount of regularization, the context window size $w$, and the optimizer. We use gradient descent with the L-BFGS method [69] as our optimizer and random cross-validation search to find the best values for $c1$ & $c2$ from the continuous exponential distribution of scale 0.5 for c1 and 0.05 for c2. In addition, the window size is set to 6 tokens as the longest entity observed in the training data.

**Hyperparameters for Transformers** We perform limited tuning for all the transformers models to avoid over-fitting. As mentioned in § 4.1, dev and test sets have samples from both source and target domains. Therefore, any over-fitting of the training data will hurt the model's generalizability. For learning rate, we use the range {4E-5, . . . , 1E-1} with a step size of 1E-2. For the maximum sequence length, we experiment with {256, 512}. In addition, for the l2 regularization coefficient, we use the range {1E-3, . . . , 1E-1} with a step size of 1E-2. Finally, we run the training procedure for a maximum of 25 epochs, with an early stopping callback function set to stop training by monitoring the loss delta set to 1E-3 and patience of 5 epochs.

### 4.4 Evaluation Metrics

We evaluate all methods using the micro-averaged F1 score as suggested in the NeurIPS'22 competition and in the existing results [51]. The score is computed as follows:

$$\mathbf{F1} = \frac{2 \times \mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{7}$$

where $\mathcal{P}$ and $\mathcal{R}$ are the average precision and average recall of all entity types, respectively. The computation of true positives, false positives, and false negatives for precision and recall is done as follows:

- A predicted span is considered as *True Positive* if the span and the predicted entity type are present in the ground truth annotations.
- A predicted span is considered as *False Positive* if the span is present in the ground truth annotations but the predicted entity type is incorrect or the predicted span is not present in the ground truth annotations.
- A span is considered as *False Negative* if the span is present in the ground truth but is absent in the predicted spans.

[7] https://github.com/TeamHG-Memex/sklearn-crfsuite

| METHOD | CONST_DIR | | LIMIT | | OBJ_DIR | | OBJ_NAME | | PARAM | | VAR | | Average Micro F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{P}$ | $\mathcal{R}$ | |
| CLASSICAL | 0.956 | 0.854 | 0.904 | **0.954** | 0.979 | 0.929 | 0.649 | 0.353 | 0.958 | 0.916 | 0.795 | 0.714 | 0.816 |
| CLASSICAL+ | **0.960** | 0.858 | 0.931 | 0.942 | **0.990** | 0.970 | 0.726 | 0.544 | 0.953 | 0.935 | 0.823 | 0.787 | 0.853 |
| XLM-RB [51] | 0.887 | 0.897 | 0.965 | 0.950 | 0.949 | 0.999 | 0.617 | 0.469 | 0.960 | 0.969 | 0.909 | 0.932 | 0.888 |
| XLM-RL | 0.930 | 0.897 | 0.979 | 0.938 | 0.979 | 0.989 | 0.606 | 0.512 | 0.963 | 0.985 | 0.899 | 0.938 | 0.893 |
| ROBERTA | 0.895 | **0.902** | 0.984 | 0.950 | **0.990** | **1.000** | 0.668 | 0.597 | 0.965 | 0.983 | 0.916 | 0.940 | 0.904 |
| XLM-RL+ | 0.901 | 0.897 | **0.987** | 0.953 | 0.989 | 0.999 | 0.665 | 0.583 | **0.971** | **0.989** | 0.918 | 0.946 | 0.907 |
| HYBRID | 0.946 | 0.890 | 0.980 | 0.942 | **0.990** | **1.000** | **0.730** | **0.668** | 0.957 | 0.983 | **0.935** | **0.953** | **0.919** |

**Table 2.** Numerical results that compare classical, modern, and hybrid models for precision, $\mathcal{P}$, and recall, $\mathcal{R}$ for each named entity together with average micro F1 score.

## 4.5 Numerical Results

Table 2 presents our results that compare different classical, modern, and hybrid methods for solving the NER4OPT on the linear programming word problems dataset. We report the performance metrics for precision, $\mathcal{P}$, and recall, $\mathcal{R}$, for each entity class together with the micro F1 score averaged across classes.

**[Q1] Performance of Classical NLP** The CLASSICAL method based on grammatical and morphological features achieves an average micro F1 of 0.816. This result establishes our performance lower bound. From there, CLASSICAL+ jumps to 0.853 by leveraging our hand-crafted gazetteer, syntactic, and contextual features. The gazetteer features focus on (OBJ_DIR) and (CONST_DIR). Accordingly, for (OBJ_DIR), the $\mathcal{P}$ and $\mathcal{R}$ increase by 0.011 and 0.041, and for (CONST_DIR), both metrics increases slightly by 0.004. The syntactic features focus on (VAR) and (OBJ_NAME) entity types. Accordingly, for (VAR), $\mathcal{P}$ and $\mathcal{R}$ increase by 0.028 and 0.077, and for (OBJ_NAME), $\mathcal{P}$ and $\mathcal{R}$ increase by 0.077 and 0.191. While the F1 score 0.816 is relatively lower compared to other methods, both classical approaches report 0.90+ $\mathcal{P}$ and 0.85+ $\mathcal{R}$ on all entity types except (OBJ_NAME) and (VAR). These two classes stand out as the difficult labels, as we discussed earlier, due to ambiguity and long-range dependency.

**[Q2] Performance of Modern NLP and the State-of-the-Art** The state-of-the-art method for NER4OPT on this dataset from [51] is based on the modern transformers architecture, specifically XLM-RB. XLM-RB improves over the classical results from an F1 of 0.816 to 0.888. Switching the underlying transformer architecture from the base model to its large version, XLM-RL, or to a different model as ROBERTA improves the results further. Interestingly, ROBERTA outperforms XLM-RB, hinting that the multilingual pre-training objective of XLM-RB [11] is not beneficial in our NER4OPT task.

When comparing modern methods that use deep contextual embeddings with their classical counterpart, we observe that recent techniques perform better; 0.853 vs. 0.907. That said, it is worth noting that while it is possible to improve the overall average F1 score when compared to classical methods, modern methods do not improve $\mathcal{P}$ and $\mathcal{R}$ in every class.

Our approach Xlm-Rl+ that combines Xlm-Rl with fine-tuning on optimization corpora improves the F1 score of 0.893 to 0.907. It is encouraging to realize this performance improvement even when fine-tuning with only a few textbooks over the pre-trained model built with large corpora. Our Xlm-Rl+ stands out as the best-performing modern method in parts with small margins.

**[Q3] Performance of Hybrid Modeling** Finally, we consider the case for combining classical and modern techniques. Overall, the best performance is achieved with our Hybrid model with an F1 score of **0.919**. This result significantly outperforms the baseline classical approach from the F1 score of 0.816 and improves over the best-known results from the F1 score of 0.888. The Hybrid model benefits from data augmentation via over-sampling to address infrequent patterns and L2 augmentation to combat long-range dependency. Beyond average results, upon closer inspection of $\mathcal{P}$ and $\mathcal{R}$ for each entity class, we find that Hybrid offers the best result in half of the scores. The largest improvement is realized for the (OBJ_NAME) entity type, which is the most challenging label to predict. To summarize, our experiments demonstrate that integrating feature engineering with feature learning coupled with fine tuning and augmentation stands out as an attractive mechanism for Ner4Opt.

## 4.6 Post-Mortem Analysis

A critical post-mortem error analysis is to inspect where the methods fail for Ner4Opt. Our initial findings reveal conflicting token spans in the annotation of entities. Let's highlight a few examples to illustrate the issue:

$\Longrightarrow$ `How many of each type of donut should be bought in order to maximize OBJ_DIR the total monthly profit OBJ_NAME?`

$\Longrightarrow$ `How many of each type of transportation should the company schedule to move their lumber to minimize OBJ_DIR the total cost OBJ_NAME?`

$\Longrightarrow$ `How many of each should the pharmaceutical manufacturing plant make to minimize OBJ_DIR the total number of minutes needed OBJ_NAME?`

In the first example from the training data, `profit` is annotated as the objective omitting the preceding adjective phrase `total monthly`. Contrarily, in the second example, this time from the dev data, `total cost` is annotated as the objective, considering the adjective as part of the entity span. On the other hand, in the last example, again from the dev data, `number of minutes` is annotated as the objective omitting the `total`.

This inconsistency is especially evident in (OBJ_NAME) entity, which turns out to be the hardest entity to predict. Similar inconsistencies exist in other classes as well. For example, in (VAR) entity, the prepositional phrase preceding a noun is sometimes tagged, sometimes ignored. Likewise, in (LIMIT) and (PARAM) non-alpha-numeric characters (e.g., $, %) preceding or succeeding the term is tagged inconsistently.

This is known as aleatoric uncertainty and is difficult to address [15]. We expect the performance of any method, classical, modern or hybrid, to saturate eventually due to inherent labeling issues. At times, even human annotators cannot agree on the exact span of annotations, making Ner4Opt challenging.

## 5 Related Work

The unique aspect of our paper is its interdisciplinary nature. At the intersection of NLP and Optimization, our main contribution is to formalize the Ner4Opt problem and offer an initial attempt at its solution. We consider classical, modern, and hybrid techniques commonly employed in the Ner literature.

Similar to our approach, early Ner systems were built via templates and hand-crafted rules. For example, in [53], corporate identity was extracted from financial texts using heuristics. As in our case, such rule-based approaches require domain experts to formulate templates. We design gazetteer, syntactic, and contextual features for optimization problems. It is also common to employ representative models such as Hidden Markov Models [37,68,66], and as in here, Conditional Random Fields [30,44]. A drawback of these methods is poor generalization, as observed in our experiments for the average scores.

Complementary to these are modern approaches, and in particular, masked language modeling employed in BERT [14], autoregressive training utilized in GPT [46], permutation-based training employed in XLNet [65]. We build on the idea of pre-trained language models to perform practical tasks without additional training [46,9]. We leverage pre-trained transformer models such as Xlm-Rb [12] and RoBERTa [35]. Additionally, we fine-tune these large language models using optimization textbooks. To the best of our knowledge, this is the first attempt to improve large language models with optimization verbiage. A recent survey of advances in modern Ner can be found in [63].

Apart from these, we exploit domain expertise in optimization and propose regular automaton to capture heuristics for the objective variable succinctly. This automaton can be further specialized for specific downstream optimization problems. Similarly, we propose the L2 data augmentation aimed at capturing the context of the objective earlier in the text.

As benchmark, we consider the linear programming word problems dataset from [51]. This work is the closest to our paper in spirit. Compared to our work, [51] goes a step further and attempts to build an interactive decision support system to assist modelers in formulating optimization problems from text. Unfortunately, [51] does not formally define the entity recognition problem. It is only mentioned briefly as one of the black boxes in the overall system architecture without details on how to solve it.

We introduce Ner4Opt as a standalone problem and an important building block of modeling assistants. We then provide an in-depth study of its solution approaches ranging from baselines to advanced hybrids. While [51] depends solely on off-the-shelf pre-trained models, we attempt domain-specific fine-tuning. Our results improve the best-known solutions from [51] considerably.

A growing body of research is dedicated to integrating machine learning and optimization. These include general algorithm configuration procedures [26,23,24], variable selection [6,25,33,36,24], branching constraint selection [64], cut selection [56], node selection[54,19], and theoretical results for tree-search configuration [2,3]. Compared to these efforts, the integration of natural language processing and optimization remains much more limited. Our paper is one of the first attempts in that direction. For optimization technology, the Ner4Opt is immediately relevant to pave the road for modeling assistants.

For NLP, Ner4Opt offers unique challenges as noted in Section (§2) such as multi-sentence dependency with high-level of ambiguity, low data regime with high-cost of annotation, and inherent aleatoric uncertainty. Linguistically, Ner4Opt is somewhat counter-intuitive. In the classical NLP setting, entities in the same class refer to similar objects in the real world, e.g., person, place, and organization, and they share grammatical properties. Contrarily, in Ner4Opt, the objects tagged in the same entity can wildly differ and be completely unrelated, e.g., the number of trucks, the number of coconuts, and the time spent brewing coffee might all be variables or even objectives. Given the challenging nature of Ner4Opt, both communities would benefit from closer integration.

The need for significant expertise to formulate models as a barrier to entry is a common concern shared by many in the community. In that regard, our paper is closely related to learning constraint models. These include learning models using generate-and-test [28], from examples [47], from spreadsheets [27], from tables [41], from solutions as in model seeker [4], and from non-solutions [7,43]. Related to this are visualization frameworks, explanations, and user hints as part of human-computer interaction [38,16,55,40]. Our work differs substantially from all of these previous works, as we are working with free form natural language text. Let us note that analogous attempts have already succeeded in other domains, e.g., turning text into Sql queries [67].

## 6 Conclusions

We envision a future in which non-technical users are empowered with optimization techniques so that they can naturally interact in multi-modal settings via text, and even voice. This requires significant advances at the intersection of multiple domains, and our paper is an initial attempt toward automated modeling assistants. Still, more work is needed toward the integration of Ner4Opt into high-level modeling frameworks. Our call to action for researchers and practitioners is to help us break the low annotated data regime to achieve revolutionary breakthroughs as realized in large language models. The optimization community is neatly suited for such success with a wide range of applications that have already equipped with exact model annotations in several problem domains.

## References

1. Araci, D.: Finbert: Financial sentiment analysis with pre-trained language models. arXiv preprint arXiv:1908.10063 (2019)
2. Balcan, M., Prasad, S., Sandholm, T., Vitercik, E.: Sample complexity of tree search configuration: Cutting planes and beyond. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. pp. 4015–4027 (2021), https://proceedings.neurips.cc/paper/2021/hash/210b7ec74fc9cec6fb8388dbbdaf23f7-Abstract.html
3. Balcan, M., Prasad, S., Sandholm, T., Vitercik, E.: Improved sample complexity bounds for branch-and-cut. In: Solnon, C. (ed.) 28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel. LIPIcs, vol. 235, pp. 3:1–3:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.CP.2022.3, https://doi.org/10.4230/LIPIcs.CP.2022.3
4. Beldiceanu, N., Simonis, H.: A model seeker: Extracting global constraint models from positive examples. In: Milano, M. (ed.) Principles and Practice of Constraint Programming. pp. 141–157. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. 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). pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1371, https://aclanthology.org/D19-1371
6. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: A methodological tour d'horizon. European Journal of Operational Research **290**(2), 405–421 (2021). https://doi.org/https://doi.org/10.1016/j.ejor.2020.07.063, https://www.sciencedirect.com/science/article/pii/S0377221720306895
7. Bessiere, C., Coletta, R., Freuder, E.C., O'Sullivan, B.: Leveraging the learning power of examples in automated constraint acquisition. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming – CP 2004. pp. 123–137. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
8. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
9. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
10. Chinchor, N., Robinson, P.: Appendix E: MUC-7 named entity task definition (version 3.5). In: Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998 (1998), https://aclanthology.org/M98-1028
11. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019)
12. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019)

13. 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)
14. 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)
15. Fisch, A., Jia, R., Schuster, T.: Uncertainty estimation for natural language processing. In: COLING (2022), https://sites.google.com/view/uncertainty-nlp
16. Goodwin, S., Mears, C., Dwyer, T., de la Banda, M.G., Tack, G., Wallace, M.: What do constraint programming users want to see? exploring the role of visualisation in profiling of models and search. IEEE Trans. Vis. Comput. Graph. **23**(1), 281–290 (2017). https://doi.org/10.1109/TVCG.2016.2598545, https://doi.org/10.1109/TVCG.2016.2598545
17. Grishman, R., Sundheim, B.: Message Understanding Conference- 6: A brief history. In: COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics (1996), https://aclanthology.org/C96-1079
18. Guns, T.: On learning and branching: a survey. The 18th workshop on Constraint Modelling and Reformulation (2019)
19. He, H., Daume III, H., Eisner, J.M.: Learning to search in branch and bound algorithms. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), https://proceedings.neurips.cc/paper/2014/file/757f843a169cc678064d9530d12a1881-Paper.pdf
20. Hildebrand, R., Poirrier, L., Bish, D., Moran, D.: Mathematical programming and operations research (2022), https://github.com/open-optimization/open-optimization-or-book
21. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spacy: Industrial-strength natural language processing in python (2020)
22. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (2018)
23. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. J. Artif. Int. Res. **36**(1), 267–306 (sep 2009)
24. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: Lee, J.H. (ed.) Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6876, pp. 454–469. Springer (2011). https://doi.org/10.1007/978-3-642-23786-7_35, https://doi.org/10.1007/978-3-642-23786-7_35
25. Kadioglu, S., Malitsky, Y., Sellmann, M.: Non-model-based search guidance for set partitioning problems. In: Hoffmann, J., Selman, B. (eds.) Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada. AAAI Press (2012), http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5082
26. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: ISAC - instance-specific algorithm configuration. In: Coelho, H., Studer, R., Wooldridge, M.J. (eds.) ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 751–756. IOS Press (2010). https://doi.org/10.3233/978-1-60750-606-5-751, https://doi.org/10.3233/978-1-60750-606-5-751

27. Kolb, S., Paramonov, S., Guns, T., Raedt, L.D.: Learning constraints in spreadsheets and tabular data. Mach. Learn. **106**(9-10), 1441–1468 (2017). https://doi.org/10.1007/s10994-017-5640-x, https://doi.org/10.1007/s10994-017-5640-x
28. Kumar, M., Kolb, S., Guns, T.: Learning constraint programming models from data using generate-and-aggregate. In: Solnon, C. (ed.) 28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel. LIPIcs, vol. 235, pp. 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.CP.2022.29, https://doi.org/10.4230/LIPIcs.CP.2022.29
29. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP optimizer for scheduling - 20+ years of scheduling with constraints at IBM/ILOG. Constraints An Int. J. **23**(2), 210–250 (2018). https://doi.org/10.1007/s10601-018-9281-x, https://doi.org/10.1007/s10601-018-9281-x
30. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Brodley, C.E., Danyluk, A.P. (eds.) Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001. pp. 282–289. Morgan Kaufmann (2001)
31. Lai, G., Xie, Q., Liu, H., Yang, Y., Hovy, E.: Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683 (2017)
32. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics (sep 2019). https://doi.org/10.1093/bioinformatics/btz682, https://doi.org/10.1093%2Fbioinformatics%2Fbtz682
33. Liberto, G.M.D., Kadioglu, S., Leo, K., Malitsky, Y.: DASH: dynamic approach for switching heuristics. Eur. J. Oper. Res. **248**(3), 943–953 (2016). https://doi.org/10.1016/j.ejor.2015.08.018, https://doi.org/10.1016/j.ejor.2015.08.018
34. 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)
35. 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)
36. Lodi, A., Zarpellon, G.: On learning and branching: a survey. Top **25**(2), 207–236 (2017)
37. Morwal, S., Jahan, N., Chopra, D.: Named entity recognition using hidden markov model (hmm). International Journal on Natural Language Computing (IJNLC) Vol **1** (2012)
38. do Nascimento, H.A.D., Eades, P.: User hints: a framework for interactive optimization. Future Gener. Comput. Syst. **21**(7), 1171–1191 (2005). https://doi.org/10.1016/j.future.2004.04.005, https://doi.org/10.1016/j.future.2004.04.005
39. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: Bessière, C. (ed.) Principles and Practice of Constraint Programming – CP 2007. pp. 529–543. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

40. O'Callaghan, B., O'Sullivan, B., Freuder, E.C.: Generating corrective explanations for interactive constraint satisfaction. In: van Beek, P. (ed.) Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3709, pp. 445–459. Springer (2005). https://doi.org/10.1007/11564751_34, https://doi.org/10.1007/11564751_34
41. Paramonov, S., Kolb, S., Guns, T., Raedt, L.D.: Tacle: Learning constraints in tabular data. In: Lim, E., Winslett, M., Sanderson, M., Fu, A.W., Sun, J., Culpepper, J.S., Lo, E., Ho, J.C., Donato, D., Agrawal, R., Zheng, Y., Castillo, C., Sun, A., Tseng, V.S., Li, C. (eds.) Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017. pp. 2511–2514. ACM (2017). https://doi.org/10.1145/3132847.3133193, https://doi.org/10.1145/3132847.3133193
42. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International conference on machine learning. pp. 1310–1318. PMLR (2013)
43. Pawlak, T.P., Krawiec, K.: Automatic synthesis of constraints from examples using mixed integer linear programming. European Journal of Operational Research **261**(3), 1141–1157 (2017). https://doi.org/https://doi.org/10.1016/j.ejor.2017.02.034, https://www.sciencedirect.com/science/article/pii/S037722171730156X
44. Quattoni, A., Collins, M., Darrell, T.: Conditional random fields for object recognition. Advances in neural information processing systems **17** (2004)
45. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
46. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
47. Raedt, L.D., Passerini, A., Teso, S.: Learning constraints from examples. In: AAAI Conference on Artificial Intelligence (2018)
48. Rajapakse, T.C.: Simple transformers. https://github.com/ThilinaRajapakse/simpletransformers (2019)
49. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for squad. arXiv preprint arXiv:1806.03822 (2018)
50. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)
51. Ramamonjison, R., Li, H., Yu, T.T., He, S., Rengan, V., Banitalebi-Dehkordi, A., Zhou, Z., Zhang, Y.: Augmenting operations research with auto-formulation of optimization models from problem descriptions (2022). https://doi.org/10.48550/ARXIV.2209.15565, https://arxiv.org/abs/2209.15565
52. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009). pp. 147–155 (2009)
53. Rau, L.F.: Extracting company names from text. In: Proceedings the Seventh IEEE Conference on Artificial Intelligence Application. pp. 29–30. IEEE Computer Society (1991)
54. Sabharwal, A., Samulowitz, H., Reddy, C.: Guiding combinatorial optimization with uct. In: Beldiceanu, N., Jussien, N., Pinson, É. (eds.) Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems. pp. 356–361. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

55. Simonis, H., Davern, P., Feldman, J., Mehta, D., Quesada, L., Carlsson, M.: A generic visualization platform for CP. In: Cohen, D. (ed.) Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6308, pp. 460–474. Springer (2010). https://doi.org/10.1007/978-3-642-15396-9_37, https://doi.org/10.1007/978-3-642-15396-9_37
56. Tang, Y., Agrawal, S., Faenza, Y.: Reinforcement learning for integer programming: Learning to cut. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 9367–9376. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/tang20a.html
57. Thie, P.R., Keough, G.E.: An introduction to linear programming and game theory. John Wiley & Sons (2011)
58. Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In: COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002) (2002), https://aclanthology.org/W02-2024
59. Van Hentenryck, P.: The OPL Optimization Programming Language. MIT Press, Cambridge, MA, USA (1999)
60. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
61. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461 (2018)
62. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
63. Yadav, V., Bethard, S.: A survey on recent advances in named entity recognition from deep learning models. arXiv preprint arXiv:1910.11470 (2019)
64. Yang, Y., Boland, N., Dilkina, B., Savelsbergh, M.: Learning generalized strong branching for set covering, set packing, and 0–1 knapsack problems. European Journal of Operational Research **301**(3), 828–840 (2022). https://doi.org/https://doi.org/10.1016/j.ejor.2021.11.050, https://www.sciencedirect.com/science/article/pii/S0377221721010018
65. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems **32** (2019)
66. Zhao, S.: Named entity recognition in biomedical texts using an hmm model. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP). pp. 87–90 (2004)
67. Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103 (2017)
68. Zhou, G., Su, J.: Named entity recognition using an hmm-based chunk tagger. In: Proceedings of the 40th annual meeting of the association for computational linguistics. pp. 473–480 (2002)
69. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on mathematical software (TOMS) **23**(4), 550–560 (1997)