

CP-Model-Zoo: A Natural Language Query System for Constraint Programming Models

Augustin Crespín, Ioannis Kostis, H       Verhaeghe, Pierre Schaus
UCLouvain, Louvain-la-Neuve, Belgium

augustin.crespin@student.uclouvain.be, ioannis.kostis@uclouvain.be, helene.verhaeghe@uclouvain.be, pierre.schaus@uclouvain.be

LLM-Solve Workshop

11 August 2025



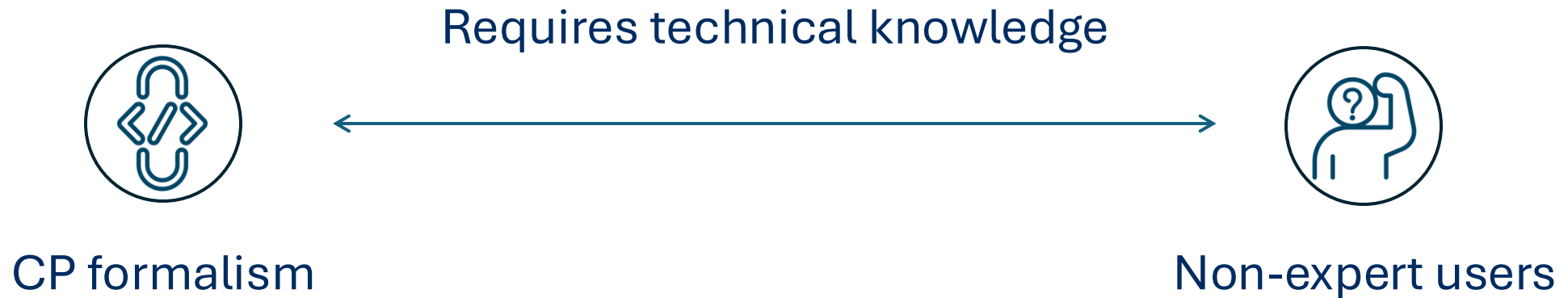
Holy Grail of Programming

“Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.”

Eugene C. Freuder, “In pursuit of the Holy Grail”, 1997

Constraint Programming

“The user states the problem, the computer solves it.”



How to make CP accessible for everyone?

Can we make CP modeling as easy as
describing your problem in plain English?

Large Language Models

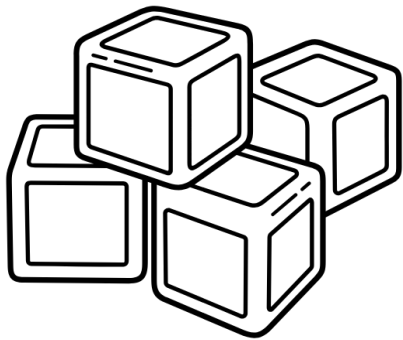


An example

ChatGPT ▾

Se connecter

Inscription gratuite



You are given 13 words: BUOY, CAVE, CELT, FLUB, FORK, HEMP, JUDY, JUNK, LIMN, QUIP, SWAG, VISA, and WISH. These words collectively contain 24 unique letters. The goal is to assign each of these 24 letters to one of the six faces of four different cubes such that each letter appears on exactly one face of one cube. The assignment must ensure that for each word, its four letters are distributed across the four different cubes. Write a MiniZinc model for solving this problem.

Here's a **MiniZinc model** to solve the cube-word-letter assignment problem.

Problem Summary:

- You have 13 four-letter words.
- Together, the words use **24 unique letters**.
- You must distribute these letters across **4 cubes**, each with 6 faces (so $4 \times 6 = 24$ letters total, perfect match).
- **Constraint:** For each word, **no two letters can be on the same cube**. That is, each word's letters must be on **4 different cubes**.

An example

🔥 MiniZinc Model:

minizinc

📄 Copier ✎ Modifier

```
% Define the letters
set of int: LETTERS = 1..24;
set of int: CUBES = 1..4;

% Each letter is assigned to one cube
array[LETTERS] of var CUBES: cube_of_letter;

% Define the 13 words, each as indices to the 24 unique letters
% You must assign these based on the 13 input words
% e.g., if A=1, B=2, ..., then BUOY might be [2, 21, 15, 25] etc.
% For now, let's define a placeholder mapping and word list
% You must replace this with actual mapping

% Example mapping of 24 unique letters
% Suppose the unique letters in order are:
% A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, Y
% Map letters to 1..24
% Then words are represented using indices
```

```
% List of words represented as arrays of indices into LETTERS
array[1..13, 1..4] of int: words = [
  % Replace with actual letter indices
  % Each row is a word of 4 letters
  | 1, 2, 3, 4
  | 5, 6, 7, 8      BUOY      O = 3
  | 9,10,11,12     CAVE      C = 5
  |13,14,15,16     CELT      C = 9
  |17,18,19,20     FLUB
  |21,22,23,24     FORK
  | 1, 5, 9,13     HEMP
  | 2, 6,10,14     JUDY
  | 3, 7,11,15     JUNK
  | 4, 8,12,16     LIMN      L = 3
  |17,21, 2, 6     QUIP
  | 3, 7,22,10     SWAG
  |11,15,19,23     VISA      V = 3
  |               WISH
];

% Constraint: each cube has exactly 6 letters
constraint
  forall(c in CUBES)(
    sum([bool2int(cube_of_letter[l] = c) | l in LETTERS]) = 6
  );

% Constraint: for each word, all 4 letters are on different cubes
constraint
  forall(w in 1..13)(
    alldifferent([cube_of_letter[words[w, i]] | i in 1..4])
  );

solve satisfy;
```

Typical errors

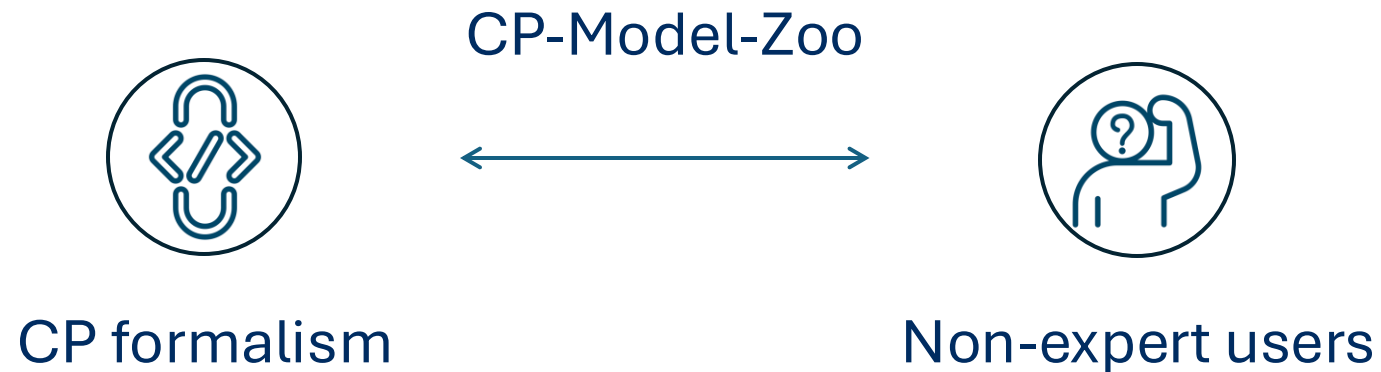
- Compilation errors (i.e., errors directly linked to the modelling language)
- Data errors (i.e., wrongly mapped input data)
- Modelling error (i.e., lacking some constraints, wrong constraints)
- Unefficient models (i.e., underuse of advanced modeling techniques such as symmetry breaking)

Not that much of a problem for experts...

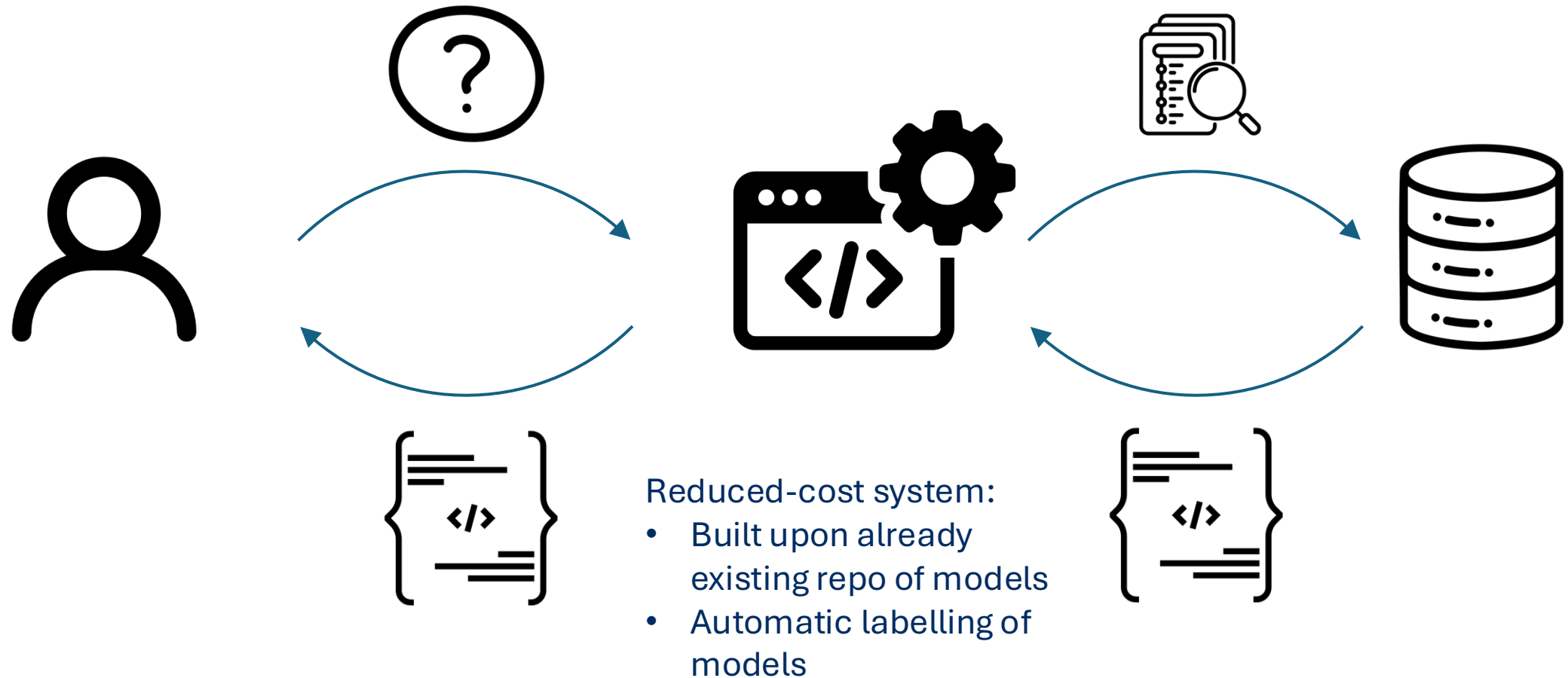
But very much for learners...

Declarative & Constraint Programming

“The user states the problem, the computer solves it.”

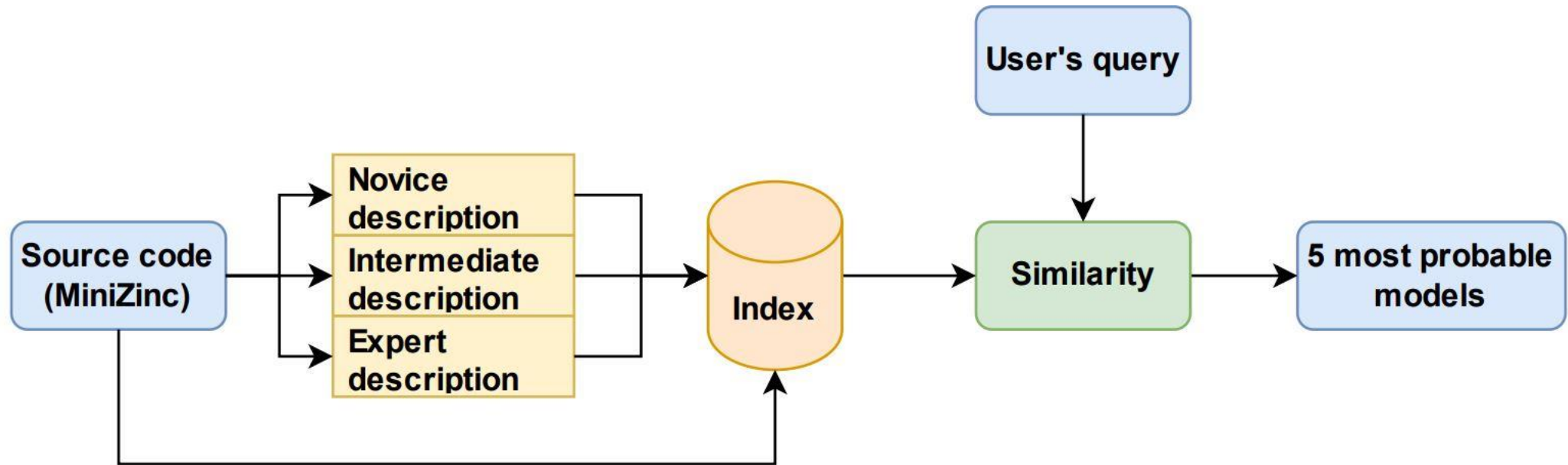


CP-Model-Zoo

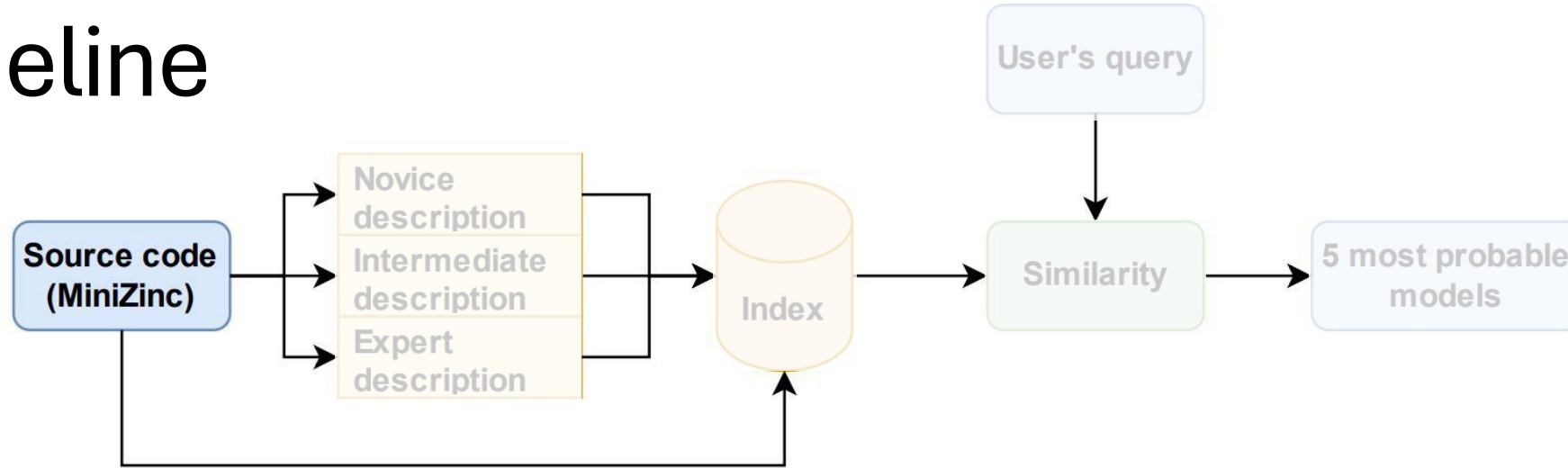


Our method

CP-Model-Zoo Pipeline



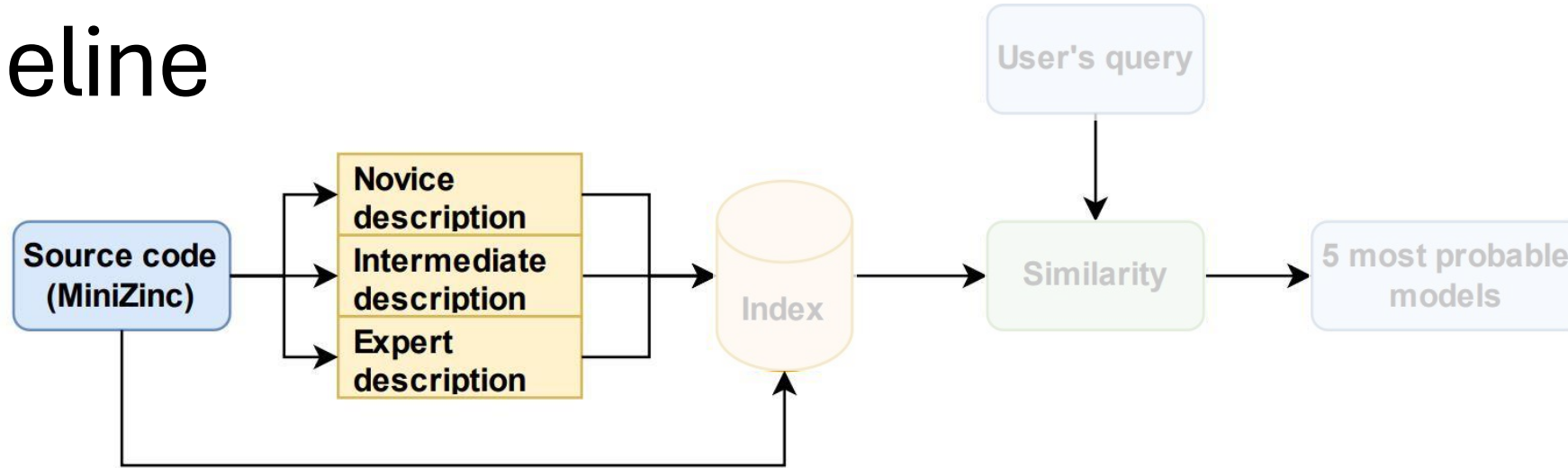
Pipeline



MiniZinc examples repository
CSPLib

} 67 problems

Pipeline



Novice: Very simple explanations in non-specific terms.

Intermediate: Domain-specific terminology.

Expert: Formal mathematical formulations

Generated automatically using LLM!

Prompts to Generate Descriptions - Intermediate

You are experienced in constraint programming and familiar with MiniZinc.

You are provided with one or more MiniZinc models representing a classic constraint programming problem. Your task is to identify the problem and explain it in clear, intermediate-level language. Assume the reader has some technical background but is not an expert.

If there are several models for the same problem, do not explain each one separately. Instead, focus on explaining the overall problem. In your answer please explain:

- The name of the problem.
- A concise description of what the problem is about.
- An explanation of the main decision variables and what they represent.
- A description of the key constraints in plain language (avoid heavy mathematical notation).
- An explanation of the problem's objective (what is being minimized or maximized).

In your answer, do not include any introductory phrases (such as 'Here is the explanation of the problem') Here is the source code of the model(s):

```
-----  
{source_code}
```

```
-----
```

Knapsack problem – Intermediate description

The problem is known as the 0/1 Knapsack Problem.

This problem is about selecting a subset of items to include in a knapsack of limited capacity, with the goal of maximizing the total value of the selected items. Each item has a size and a profit, and the knapsack has a limited capacity.

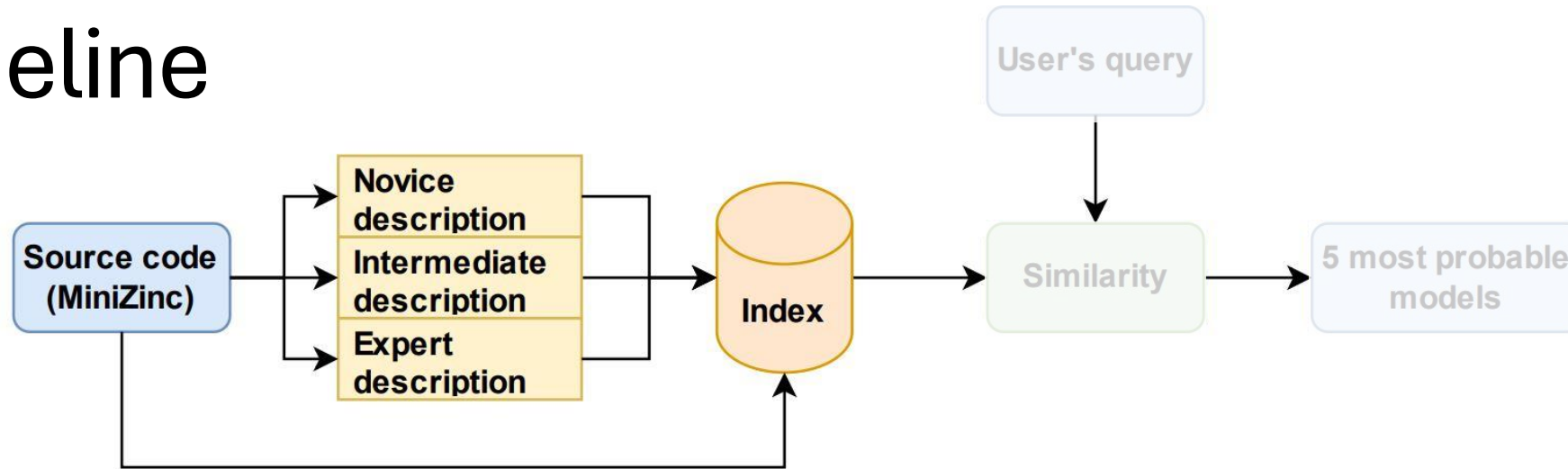
The main decision variables are the x_i variables, which represent whether each item is included in the knapsack (1) or not (0).

The key constraints are:

- Each item can only be included once (or not at all), which is ensured by the x_i variables being binary.
- The total size of the included items must not exceed the knapsack's capacity.

The objective is to maximize the total profit of the included items. In other words, the problem is to find the optimal selection of items to put in the knapsack, given the capacity constraint, to maximize the total value of the selected items.

Pipeline



Source Code + Synthetic Descriptions of varying expertise levels

Pre-computation of embeddings

Embedding

Words semantically close = Close in the embedding space

Embedding

Words semantically close = Close in the embedding space

Dense embedding: learned rather than defined

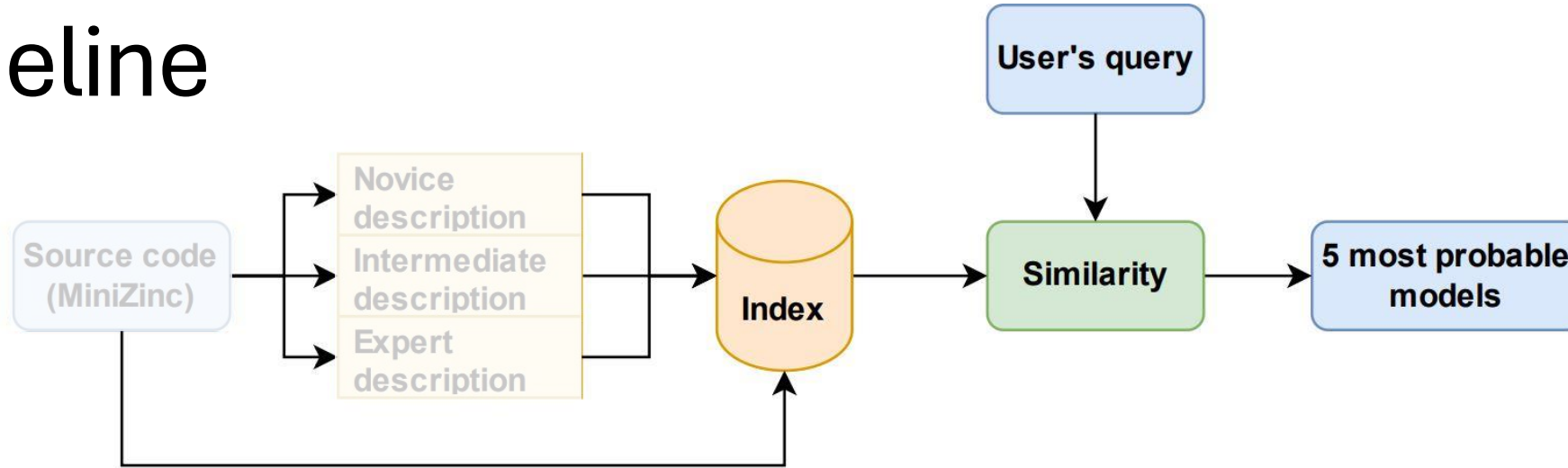
Word2Vec: binary prediction task

Given a *target word* t , is a *candidate word* c likely to occur in its context?

$$P(\text{positive}|t, c) = \sigma(t \cdot c) \text{ with } \sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

$$LL(\theta) = \sum_{t, c \in +} \log(P(+|t, c)) + \sum_{t, c \in -} \log(P(-|t, c))$$

Pipeline



The top-k retrieved items are the $s_i \in D$ with the highest similarity scores

$$\operatorname{argmax}_{s_i \in D}^k \operatorname{sim}(e_q, e_i)$$

e_i = precomputed embedding of a string in the database

e_q = embedding vector of the query, computed at inference time

Similarity measure: cosine similarity

= Finding a relevant document d in a collection D of documents that best matches a query q

$$\text{cosine similarity}(p, q) = \cos(q, p) = \frac{q \cdot p}{||q|| \cdot ||p||} = \frac{\sum_{i=1}^{|V|} q_i p_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} p_i^2}}$$

Experiments

Metric of Evaluation: Mean Reciprocal Rank

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

Q = set of queries

rank_i = rank position of the correct source-code model for the i -th query.

$\text{MRR} = 1 \iff$ corresponding source-code model was ranked first for each query

Metric of Evaluation

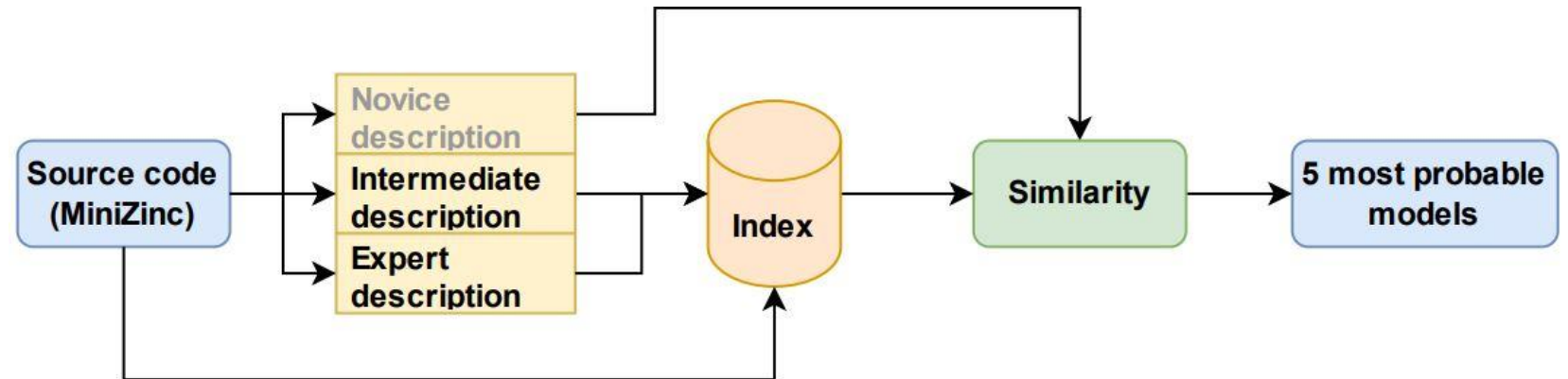
$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

Query	Rank of Relevant Result	Reciprocal Rank
Q1	1	1.0
Q2	3	0.333
Q3	2	0.5
Q4	no relevant result	0.0

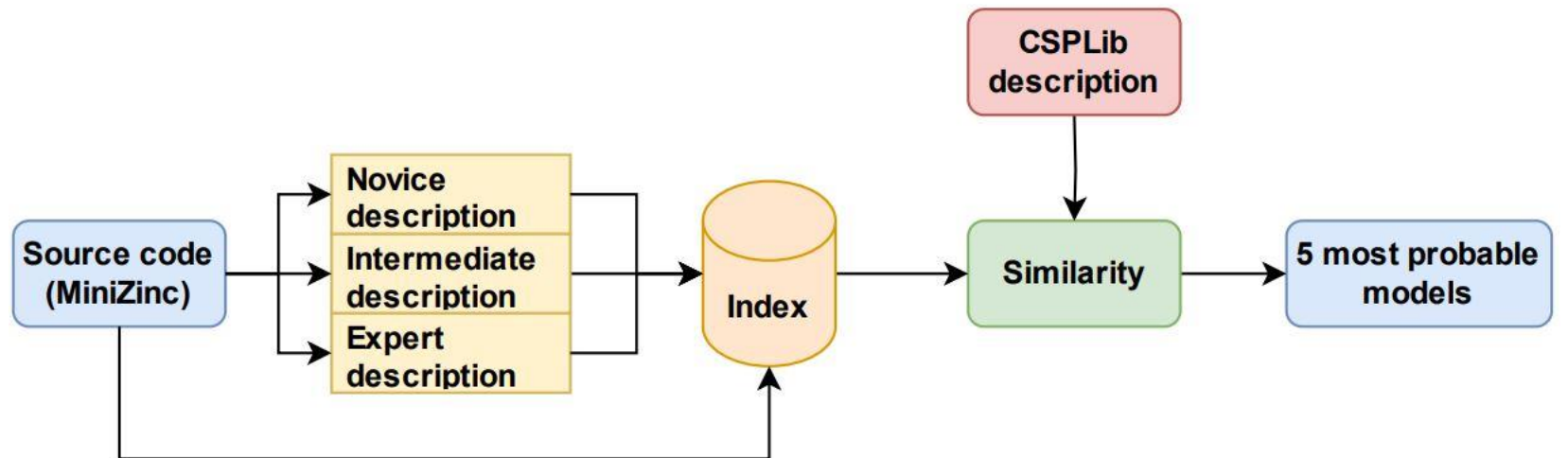
$$\text{MRR} = (1,0+0,333+0,5+0,0)/4 = 0,45825$$

Experiments

Leave-One-Out



CSPLib Description



Results

		Source of Embedding							
		SC	SC + D1	SC + D2	SC + D3	SC + D1&2	SC + D1&3	SC + D2&3	SC + D1&2&3
Query	D1	0.9851	-	1.0	0.9925	-	-	1.0	-
	D2	0.9254	0.9378	-	0.9577	-	0.9478	-	-
	D3	0.9527	0.9639	0.9925	-	0.9776	-	-	-
	CSPLib	0.8736	0.8898	0.9051	0.8676	0.8796	0.8704	0.8829	0.8634

Web Application

Live Demo of the Web Application

<https://cp-model-zoo.info.ucl.ac.be/>

Conclusion & Future Work

Improvement to the application

- Supports other modeling languages
- Addition of more models
- Automatic retrieval of new models from repositories

Future work could explore

- Alternative ways to evaluate the system's practical usefulness
- Problem categorization or taxonomy as a new feature
- Adaptation of the retrieved models to the needs of user (e.g., using the retrieved model as a correct base for an LLM to adapt to the actual variation of the problem at hand)

Thank you for listening!

Any questions? Suggestions?

<https://cp-model-zoo.info.ucl.ac.be/>

<https://hverhaeghe.bitbucket.io/>