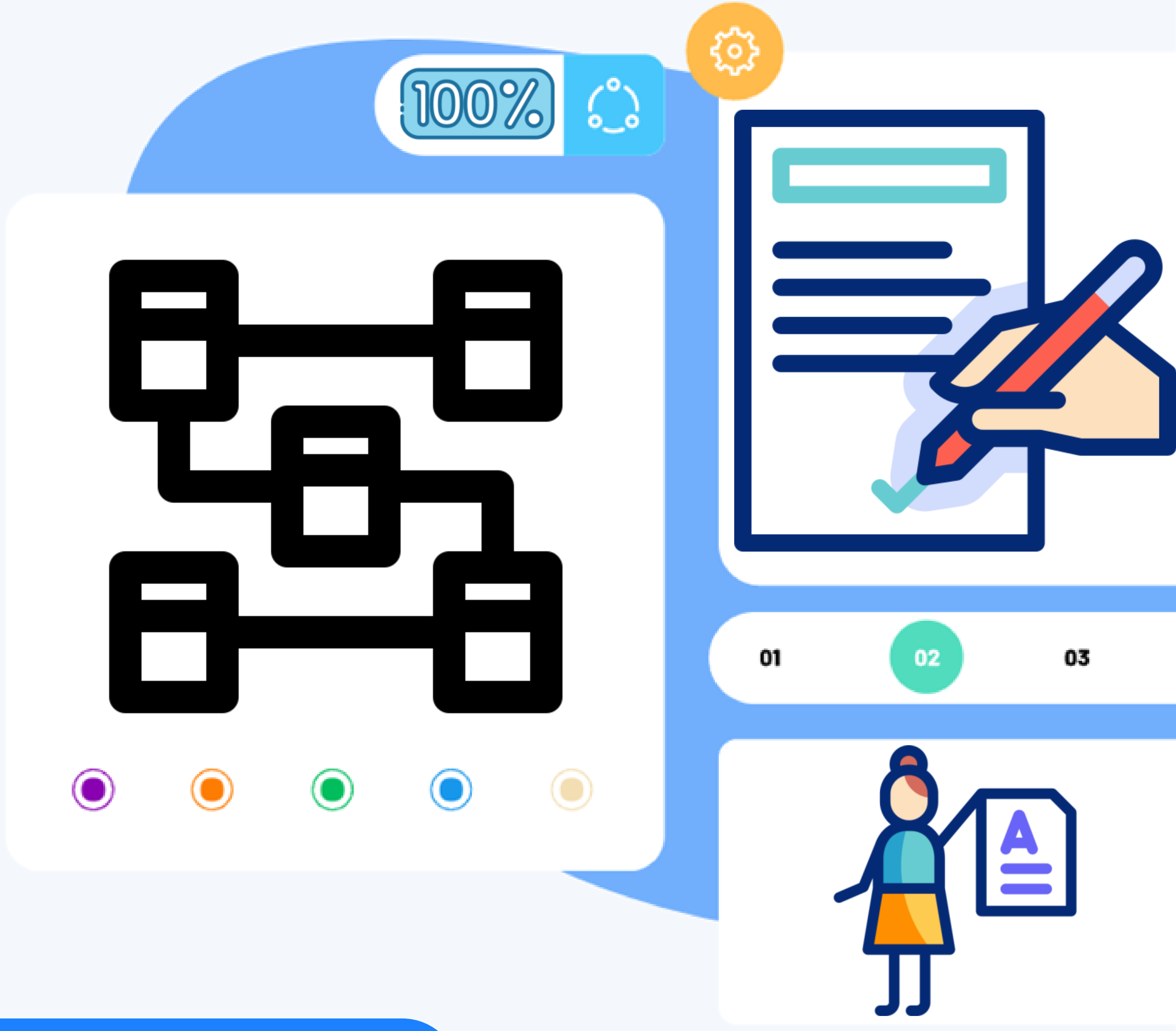
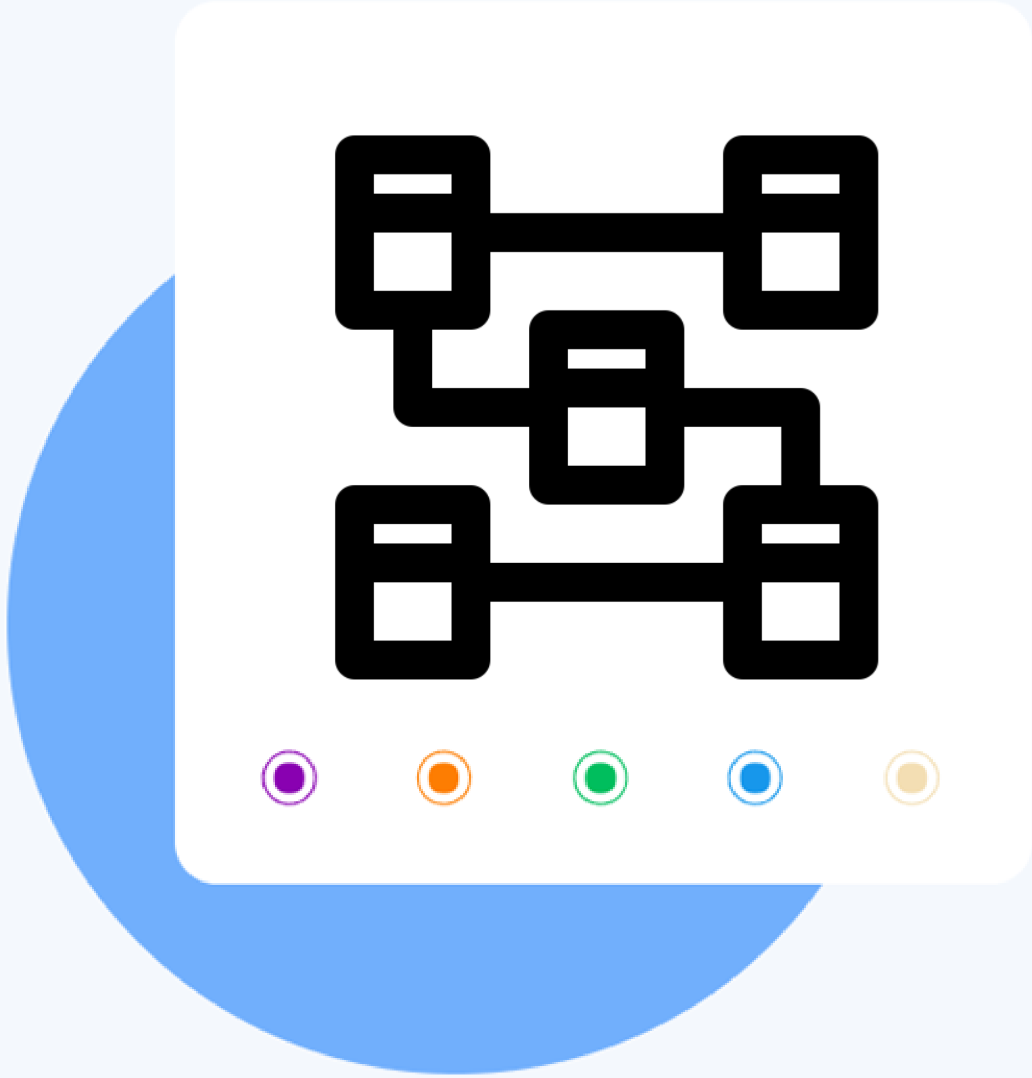
MODELS 2024 Educators Symposium.
September 2024. Linz, Austria.

Embedding-based Automated Assessment of Domain Models



Kua Chen[1], **Boqi Chen**[1], Yujing Yang[1], Gunter Mussbacher[1], Daniel Varro[1, 2]

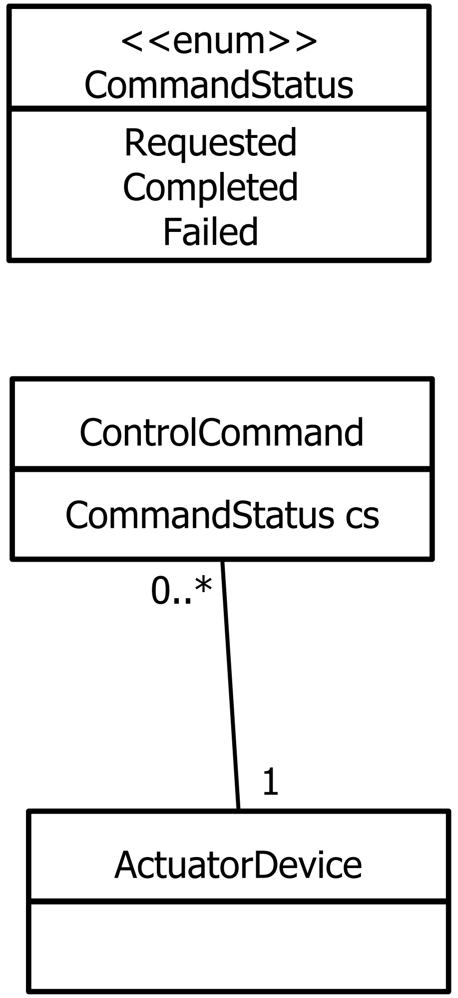
[1]Electrical and Computer Engineering, McGill University, Montreal, Canada
[2]Department of Computer and Information Science (IDA), Linköping University, Linköping, Sweden



01

Introduction

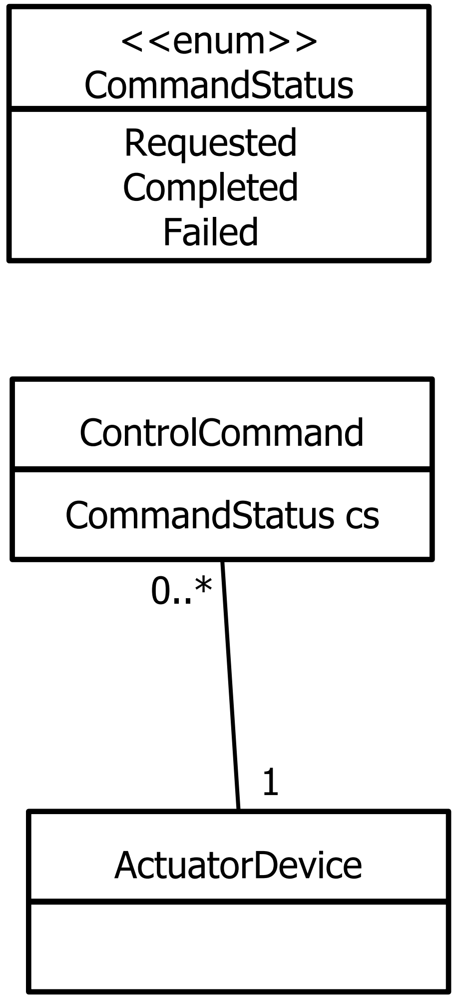
Domain Modeling



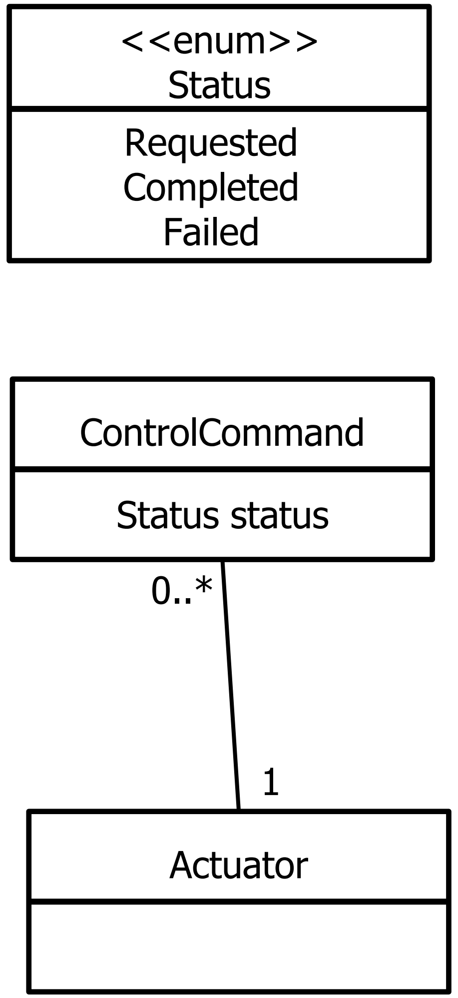
Domain modeling

- Captures relations between different entities of a domain
- Is a **core concept** for in software engineering practice and education
- An active research field for **automated generation**
- In **both cases**, a large amounts of assessments against a *reference solution* are required!

Domain Model Assessment



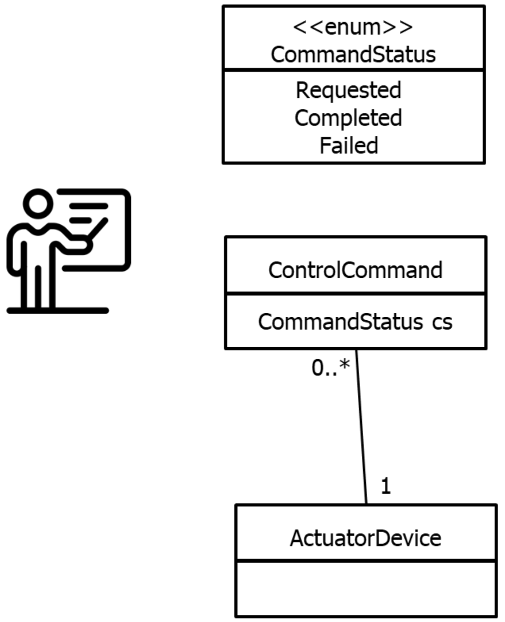
How good is the student solution?



Reference solution

Student solution

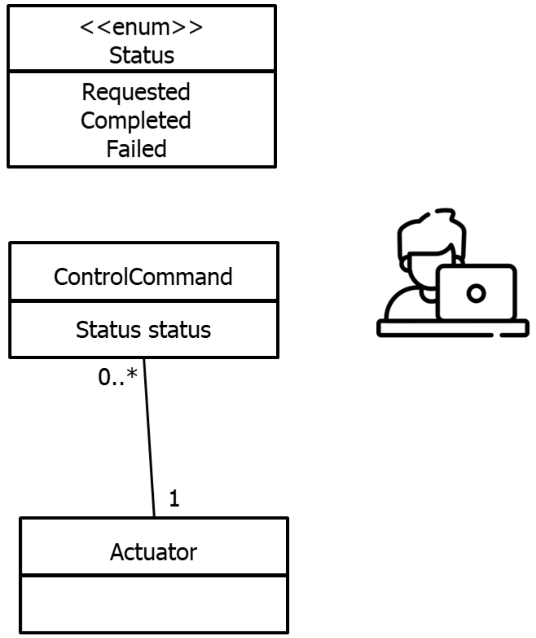
Domain Model Assessment



Reference solution

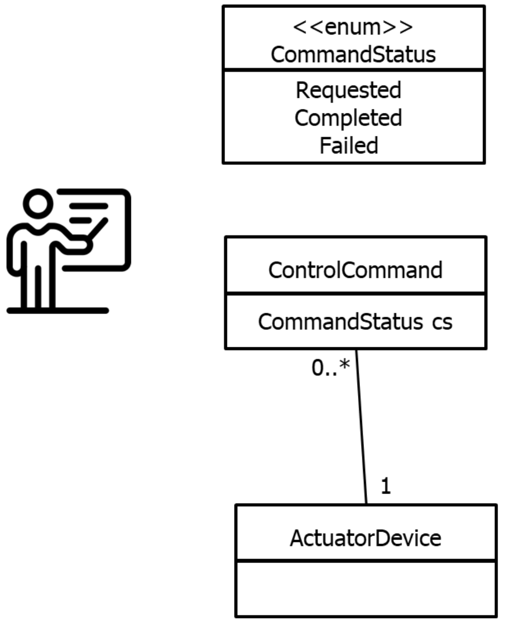
Rule-based method?

- Hard to generalize
- Manual effort
- Error-prone



Student solution

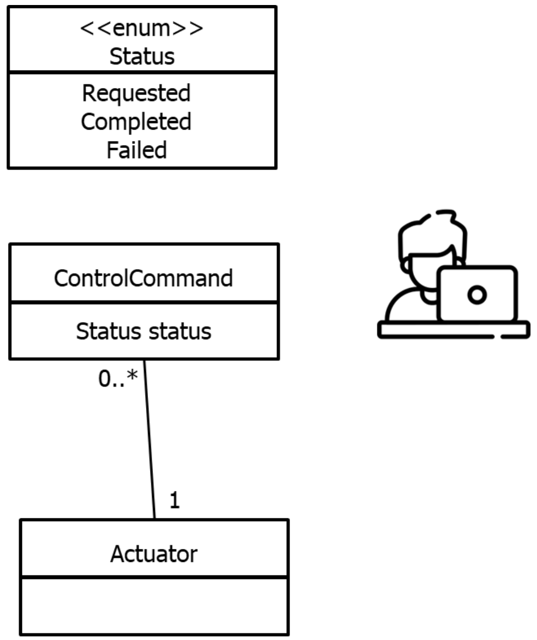
Domain Model Assessment



Reference solution

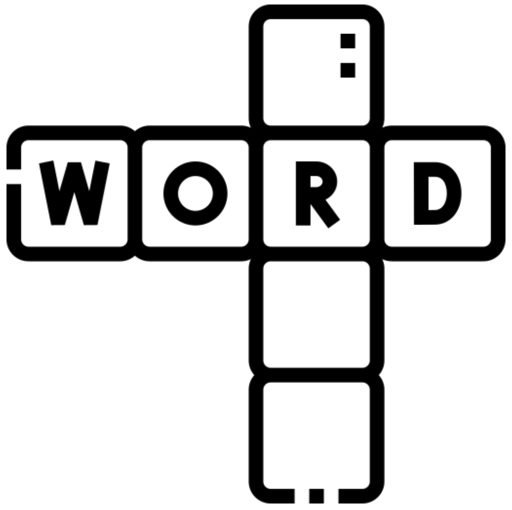
Large language models (LLMs)?

- Hallucination
- No explanation
- Can we trust it?



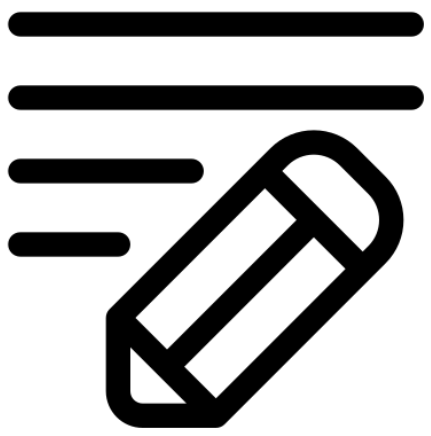
Student solution

Text Embeddings



Word Embedding: Skip-gram MDE

- Represent **words** in a **fixed-dimension vector**
- Predicting the surrounding words in a sentence
- Trained on modeling corpus
- e.g., embed(‘model’) = [0.15, 0.28, 0.123, …]

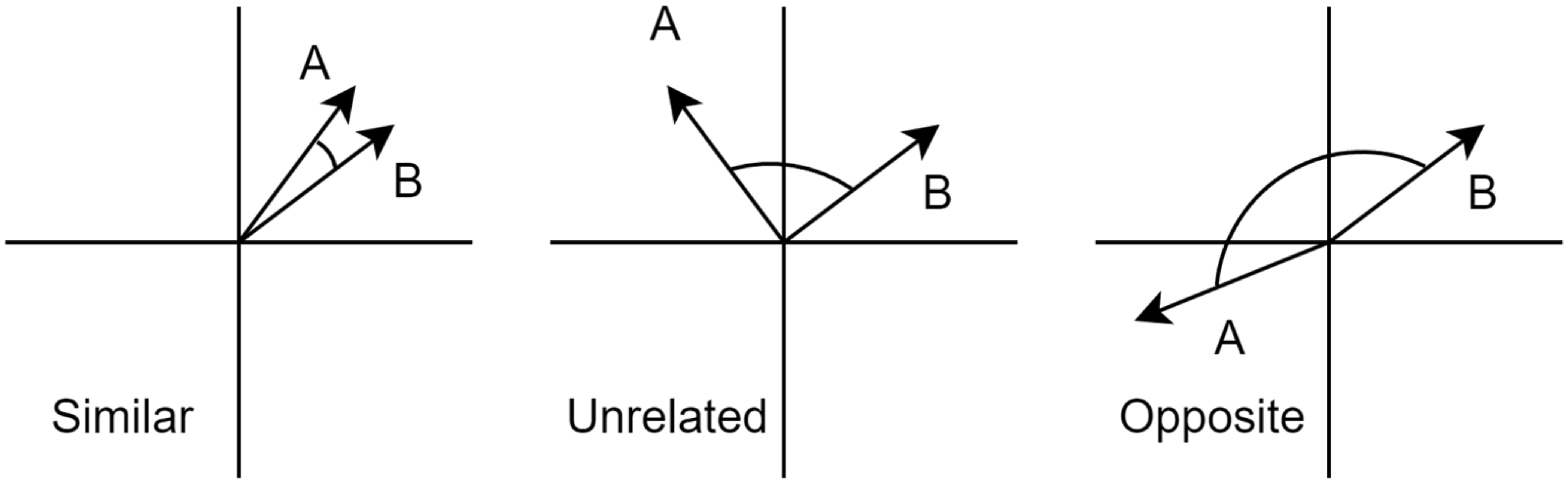


Sentence Embeddings: text-embedding-ada-002

- Represent a sequence of words in a fixed-dimension vector
- Captures the **relations of words** in the sentence

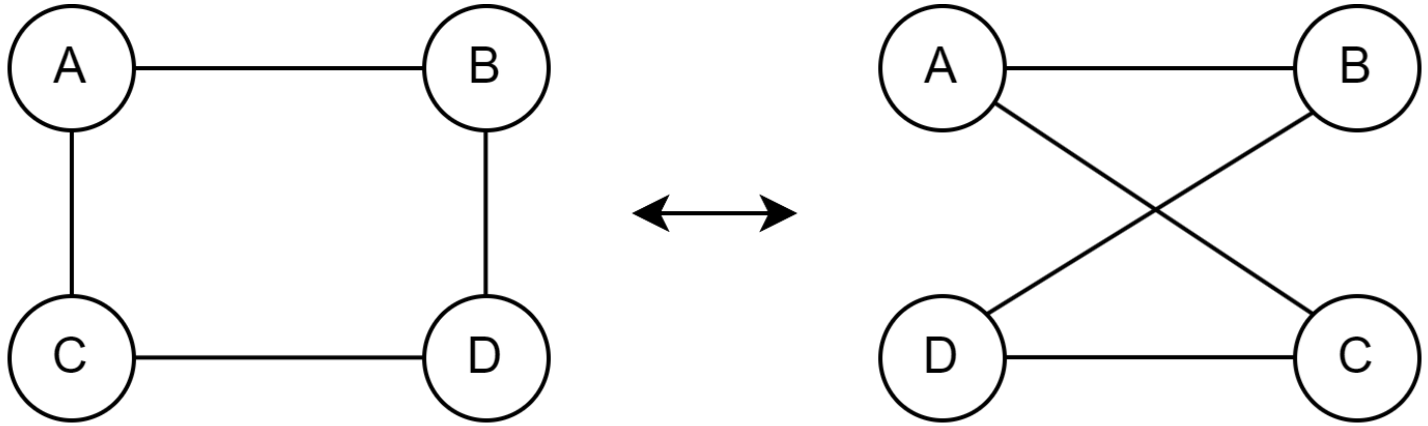
Cosine Similarity

- Cosine similarity measures the similarity between two vectors
- It can be used to measure the similarity between text embeddings



Graph Similarity Measures

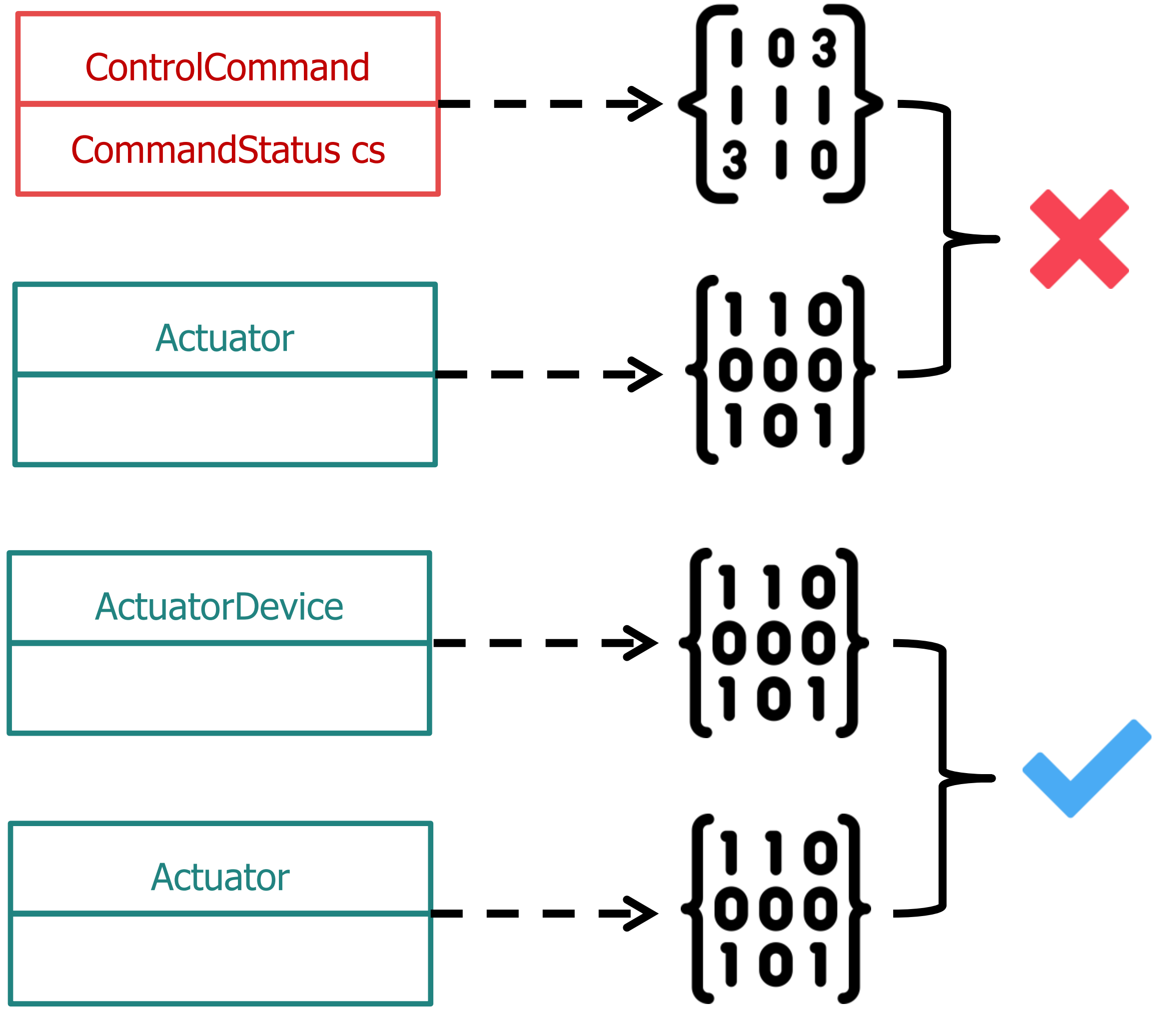
- **Graph isomorphism**
  - Determine if two graphs have the **same** underlying structure



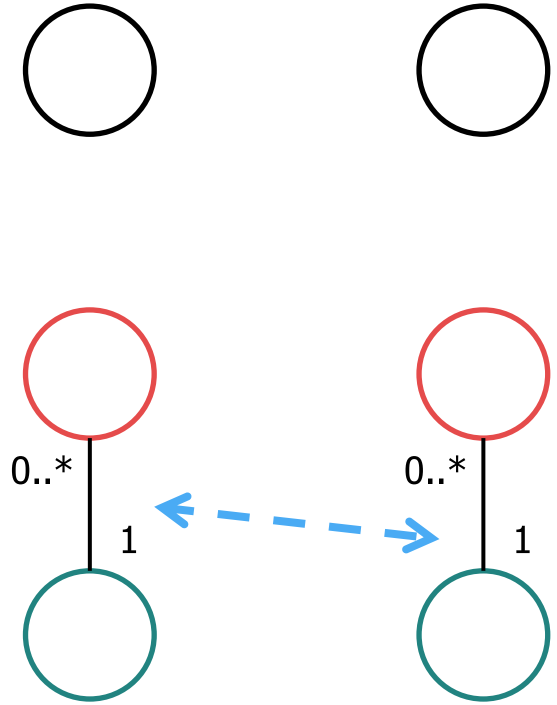
- **Graph edit distance (GED)**
  - The minimum cost of an **edit path** to transforms one graph **isomorphic** to the other
  - Edit path: a sequence of **edit operations** (inserting, deleting, and relabeling vertices or edges)

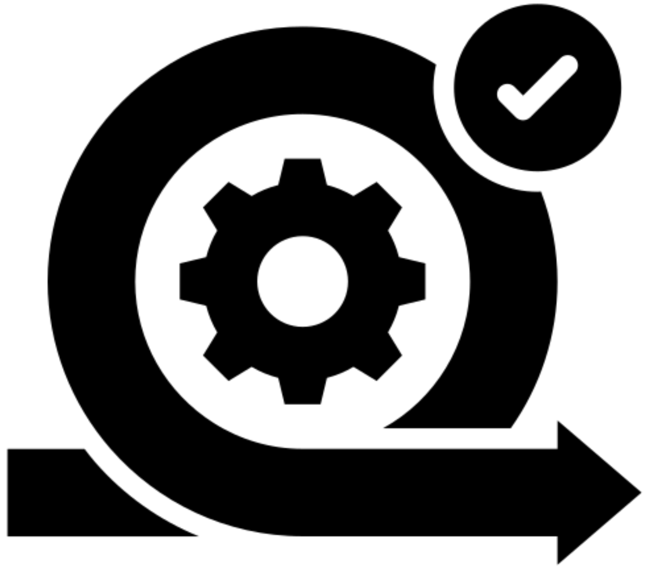
Our approach

Element matching: Text embedding



Relation matching: Graph matching

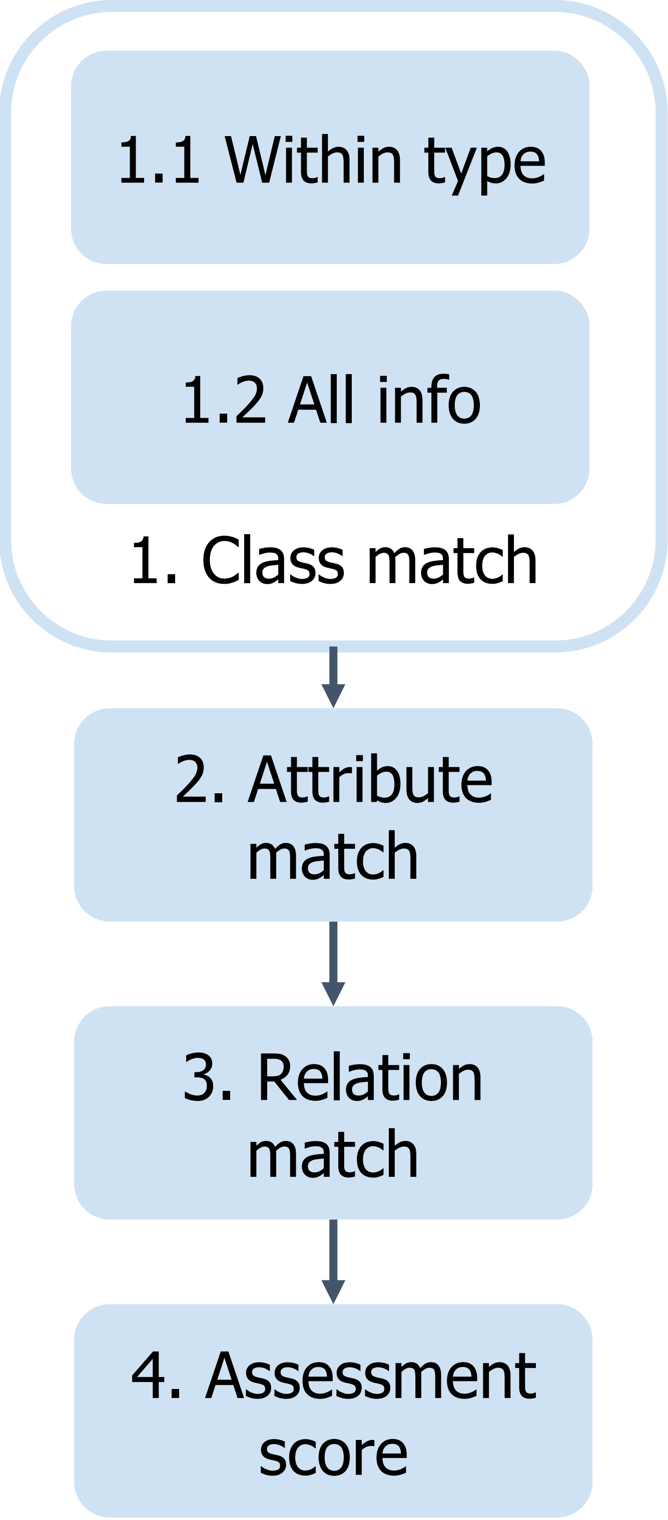




02

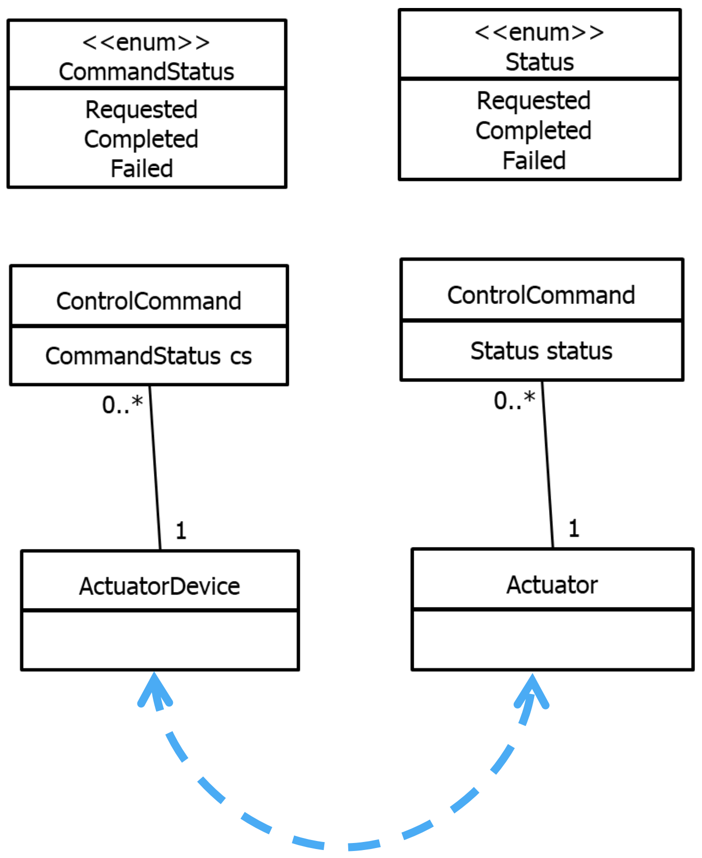
Method

Overview


1.1 Within type
1.2 All info
1. Class match
2. Attribute match
3. Relation match
4. Assessment score

Stage 1: Class Matching

- Match classes based on **class names** and **attributes** with **word embeddings**
- Match classes based on class names and attributes and **relations** with **sentence embeddings**



Compute embeddings

Word embeddings
emb("ActuatorDevice")
…

Pairwise similarities

sim("Actuator", "ActuatorDevice") = 0.99
…

Graph edit distance

Find optimum node matching given similarity

Class matches

(Actuator, ActuatorDevice)
…

1.1 Without relation

1.2 With relation

1. Class match

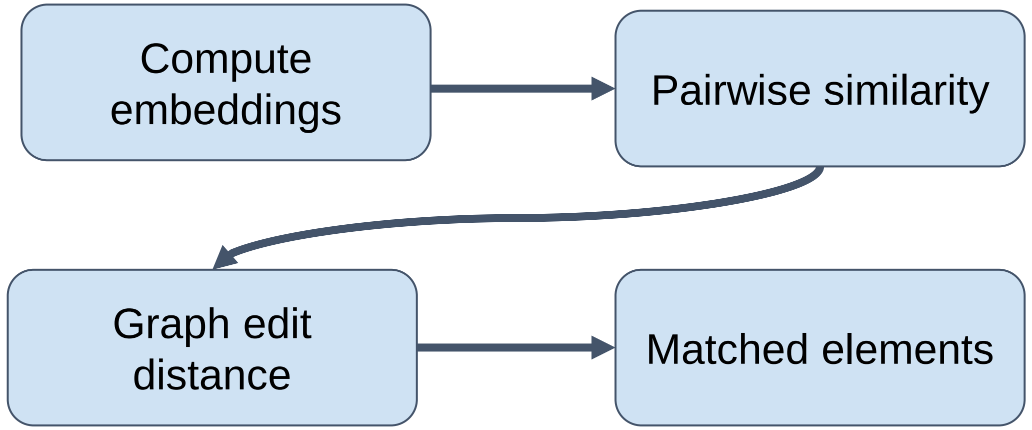
2. Attribute match

3. Relation match

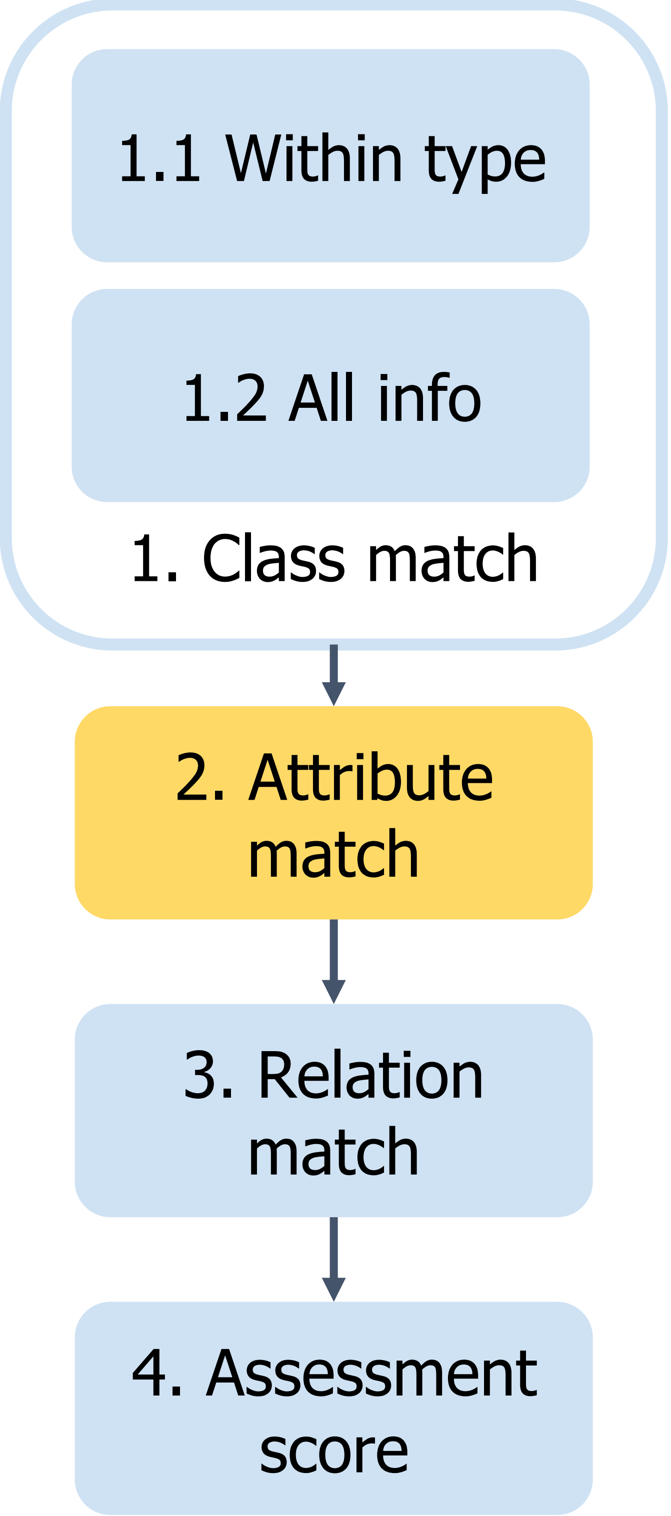
4. Assessment score

Stage 2: Attribute Matching

- **Similar logic to class matching**



- Stage 2.1 Attribute matching between matched classes
- Stage 2.2 Attribute matching between any classes
- Stage 2.3 Reference attribute to candidate class matching
- Stage 2.4 Reference class to candidate attribute matching



Stage 3: Relation Matching

- **Different logic** from class matching or attribute matching
- Relation (R) = multiplicity 1, class 1, relation type, multiplicity 2, class 2
- A **candidate relation** R1 is matched with the **reference** R2 if
  - $\text{class}^{R1} = \text{class}^{R2}$ (from the class match)
  - $\text{relation type}^{R1} = \text{relation type}^{R2}$
  - $\text{multiplicity}^{R1} = \text{multiplicity}^{R2}$
- An Example of perfect match:

* ControlCommand associate 1 ActuatorDevice
* ControlCommand associate 1 Actuator

1.1 Within type

1.2 All info

1. Class match

2. Attribute match

3. Relation match

4. Assessment score

Stage 4: Assessment score

- Each model element receives a matching score.
  - Perfectly matched: score of **1**
  - Partially matched: score of **0.5**
  - Not matched: score of **0**

- Calculate Precision, Recall, and F1 based on matches
- Final grade is a **weighted average** of F1 scores for classes, attributes and relations: $grade = \frac{w_c F_1^C + w_a F_1^A + w_r F_1^R}{w_c + w_a + w_r}$

1.1 Within type

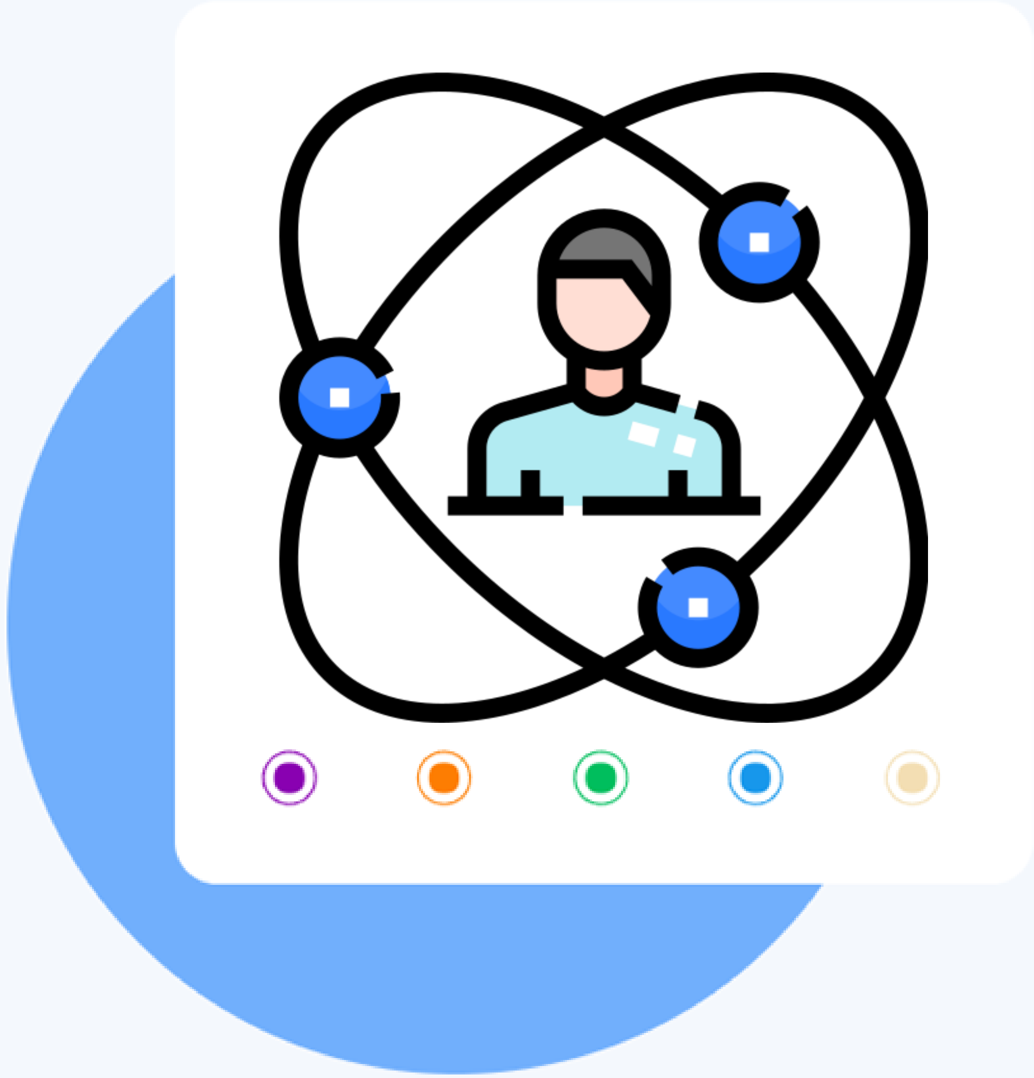
1.2 All info

1. Class match

2. Attribute match

3. Relation match

4. Assessment score



03

Experiment

Experimental Settings

- Modeling problem: smart home domain
- From undergraduate software course
  - 5 enumeration classes, 15 regular classes, 3 abstract classes
  - 13 enumeration literals, 13 attributes
  - 32 relations
- 20 student solutions

- Embedding
  - WordE4MDE library
  - OpenAI embed-ding model text-embedding-ada-002

Research Questions

RQ1: What is the performance of our algorithm in **matching** a candidate domain model to a reference model regarding classes, attributes, and relations?

RQ2: To what extent does the algorithm-generated **grade** compare with those produced by **human grading** or other automated approaches?

RQ1: Evaluation of Generated Matches

- Manually match model elements: human matches
- The matches are evaluated with precision / recall and F1
- For any modeling element *a* from the reference model, *b* from the candidate model (*a*, *b*) represents *a* matches *b*

a

(*a*, *b*)
- (*a*, *b*) → True positive
- (*a*, *c*) → False positive
- (*a*, *None*) → False positive

(*a*, *None*)
- (*a*, *b*) → False negative
- (*a*, *None*) → True negative

algorithm human

RQ1: Matching Performance

Evaluation on precision, recall and F1-score for each type of modeling elements (%)

Precision

Class	74.25
Attribute	72.74
Relation	**85.56**

Recall

Class	**93.32**
Attribute	78.61
Relation	76.28

F1

Class	**82.39**
Attribute	74.56
Relation	79.58

- Close to human matches but still room for improvements
- Better at identifying **classes** compared to **attributes**

RQ2: Grading Performance

At the end of the day, we need a “grade”

1

Compare with human

- Comparison with grading from one **author** of this paper
- Use metrics **Mean Absolute Error** and **Pearson Correlation**

2

Compare with other tools

- Comparison using an **external** benchmark
- Compare numerical grades

RQ2: Evaluation of Generated Score

Our approach gives: $grade = \frac{w_c F_1^C + w_a F_1^A + w_r F_1^R}{w_c + w_a + w_r}$

Set $w_c = 4, w_a = 1$ *and* $w_r = 1$ based on grading practice

Comparing generated grades from human grades

- Mean absolute error (**MAE**)
  - $\mathrm{MAE} = \frac{1}{\mathrm{n}} \sum_{i=1} n \, |Actual_i - Predicted_i|$
- Pearson **correlation** between two set of scores

TouchCore Comparison

GPT4-turbo
(few-shot prompting)

RQ2: Grading Performance - External Comparison

- Comparison on our benchmark and an external benchmark against human grading
    - Typical letter grade range: ~5%, e.g., A- ≈ 80% - 85%

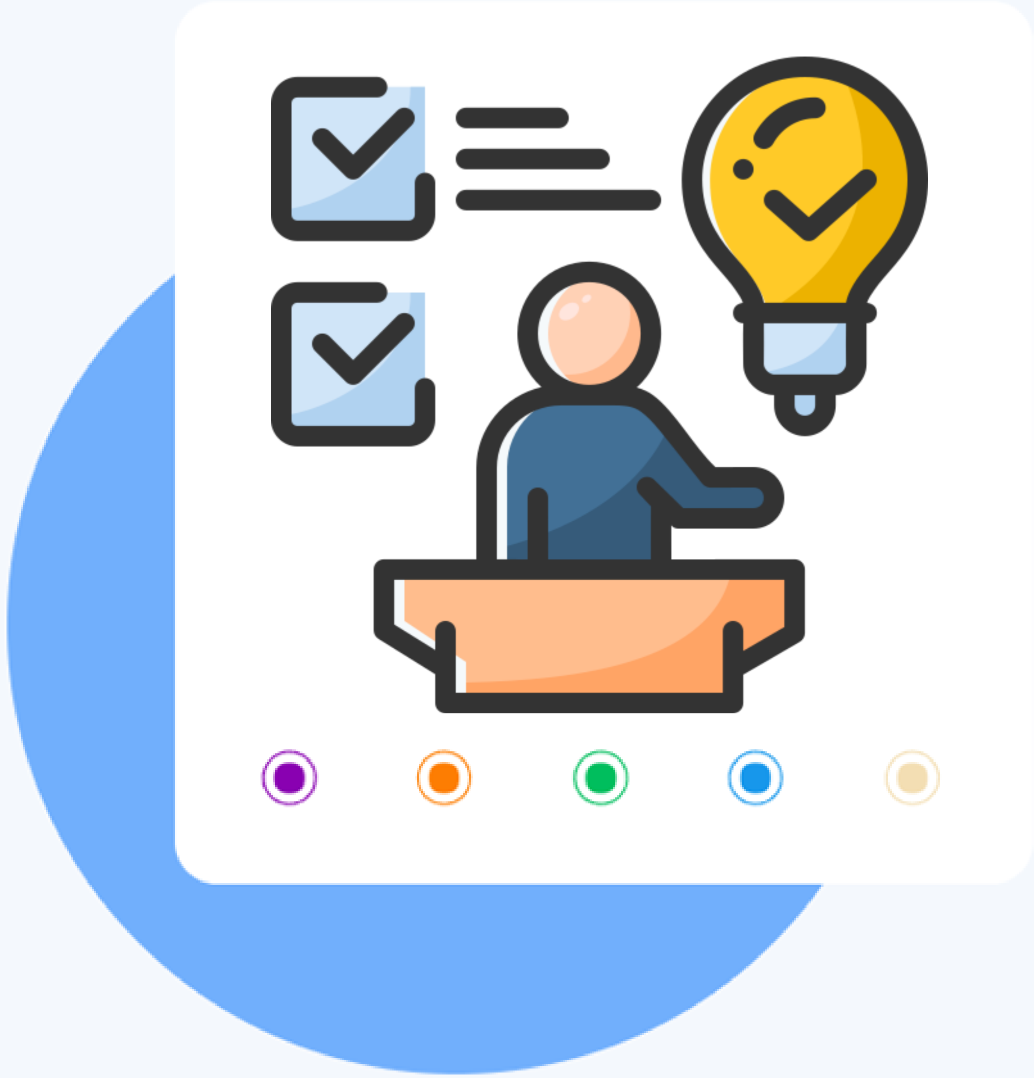
Methods	MAE	Correlation
Ours	0.0310	0.8714

Results on our benchmark

Methods	MAE
TouchCore	0.2524
GPT-4	0.0674
Ours	**0.0456**

Results on benchmark from Singh et. al.
(manual grades provided by modeling experts)

- Our approach **closely correlates** with grades given by human
- Our approach **outperforms** both rule-based and LLM baselines



06

Conclusion

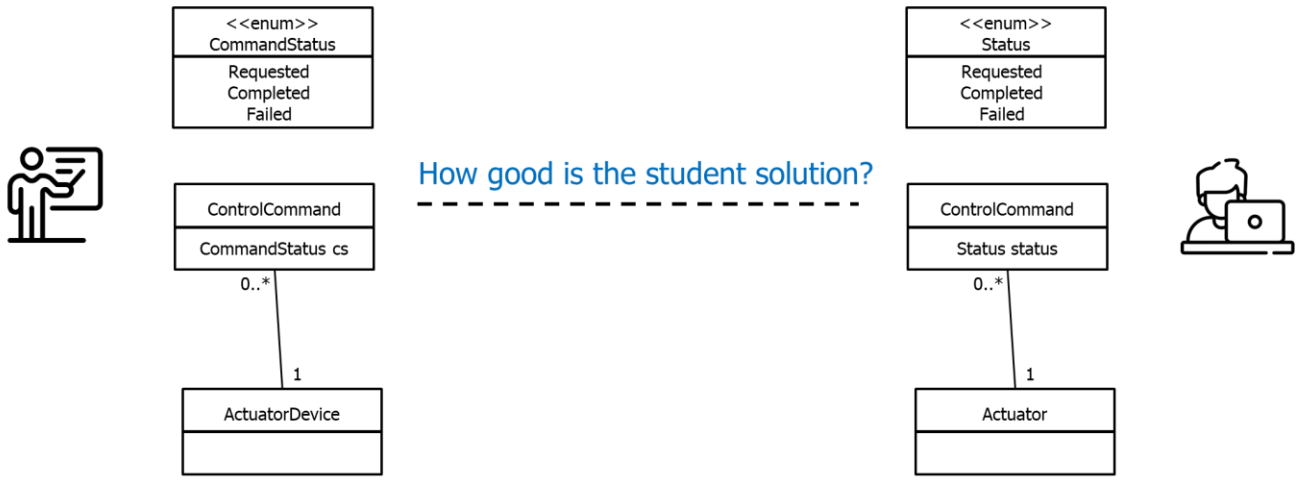
Conclusion

- Introduces a novel algorithm for automated assessment utilizing **text embeddings** and **graph matching techniques**

- Highly correlate with human grading, but there remains potential for **enhancement**

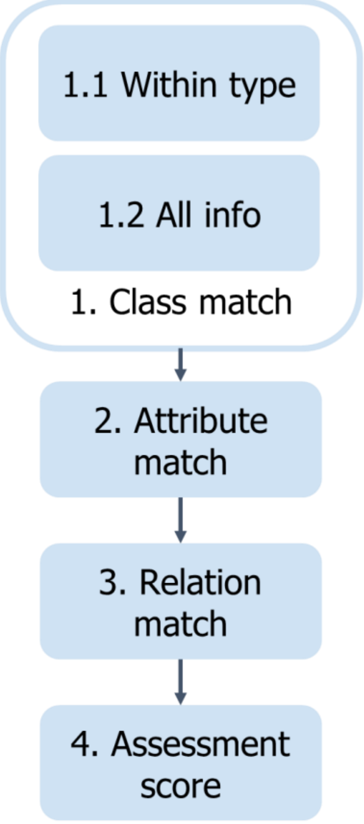
- Future directions
  - User-friendly interface
  - Generate human-readable feedback

Domain Model Assessment



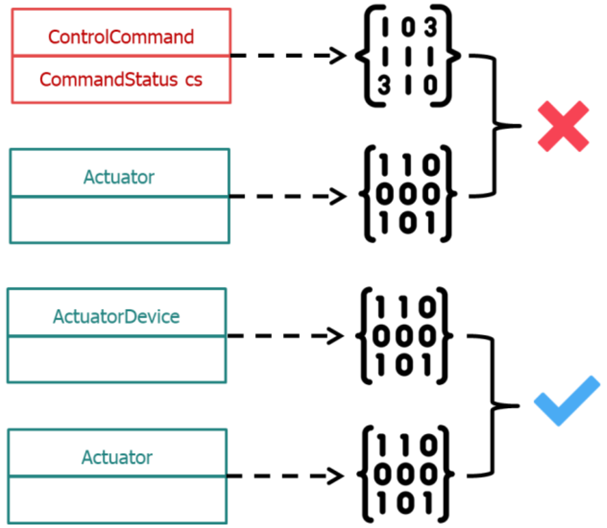
Reference solution **Student solution**

Overview

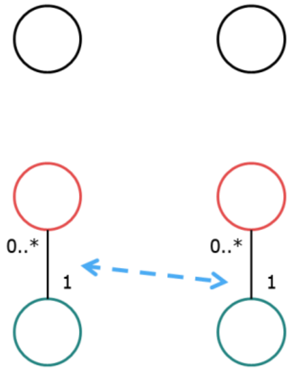


Our approach

Element matching:
Text embedding



Relation matching:
Graph matching



Research Questions

RQ1: What is the performance of our algorithm in **matching** a candidate domain model to a reference model regarding classes, attributes, and relations?

RQ2: To what extent does the algorithm-generated **grade** compare with those produced by **human grading** or other automated approaches?