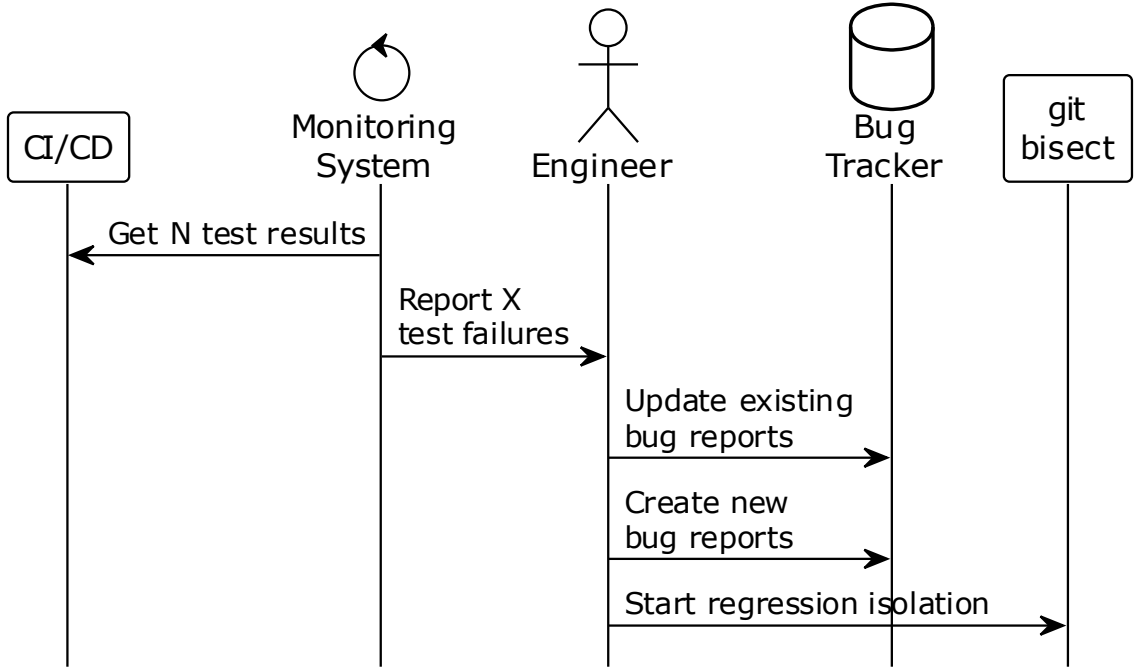
# AI-based failure aggregation

Łukasz Towarek

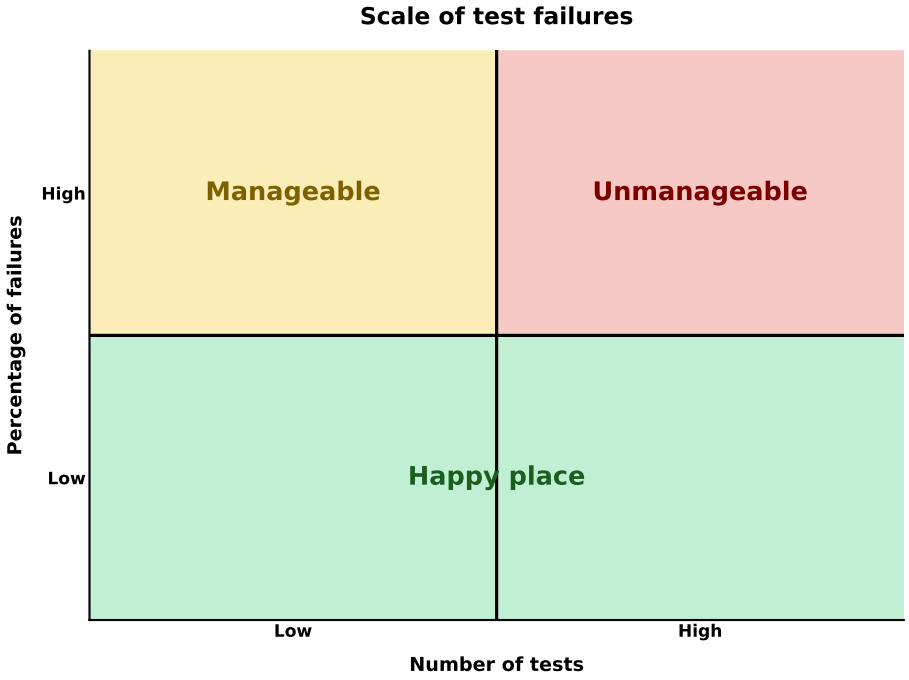
FOSDEM 2026

# Problem statement

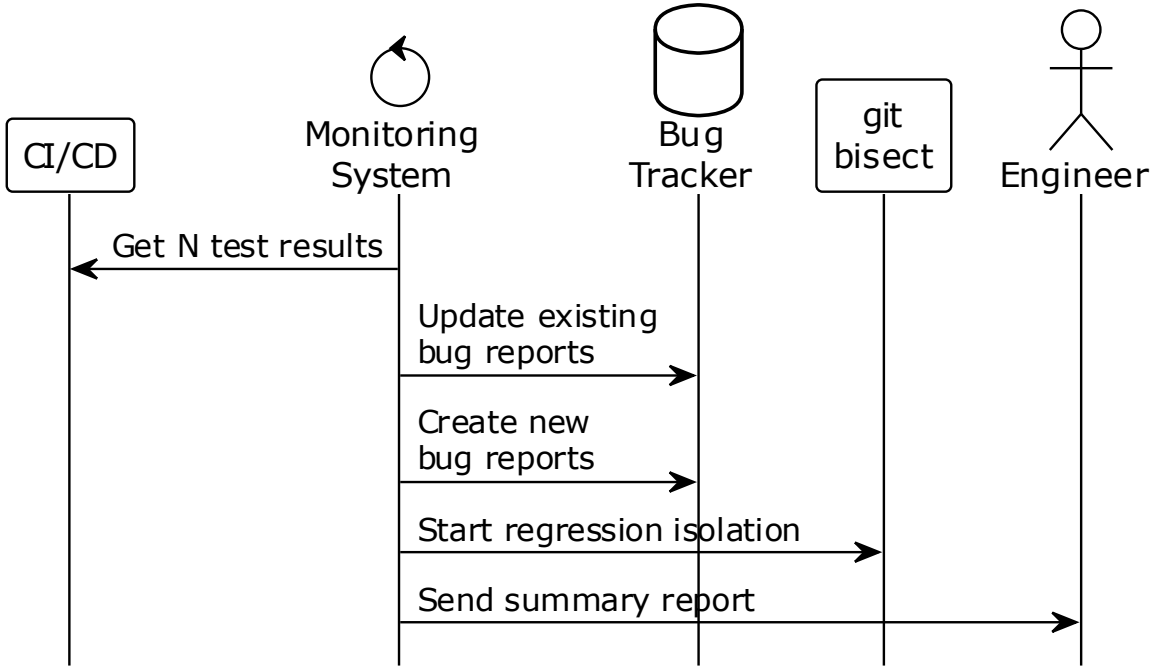
# Original workflow



# Scale


Scale of test failures
Manageable
Unmanageable
Happy place
High
Low
Percentage of failures
Low
High
Number of tests

# Desired workflow



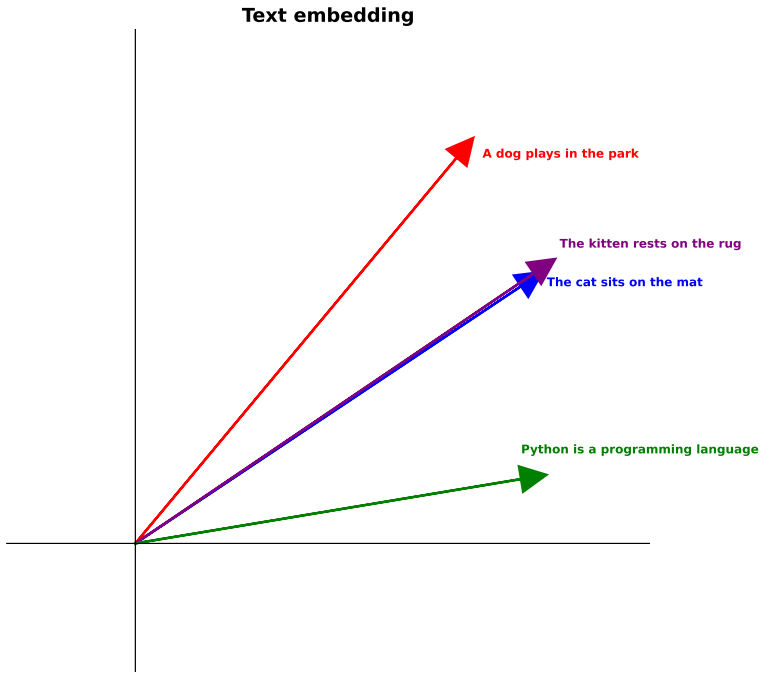
# Potential solutions

- **Goal:**
  - Fully automated agent
- **Problem:**
  - How to determine if a failure is a new issue or an already reported bug?

- **Potential solutions:**
  - No aggregation:
    - Duplicates
    - Waste of resources
  - Aggregation per test case:
    - Duplicates across test cases
    - Missed nested regressions
  - Direct logs comparison:
    - Complex text cleaning
  - Train new ML model:
    - Time-consuming and error-prone preparation of a dataset
    - Expensive
- **There’s another way**

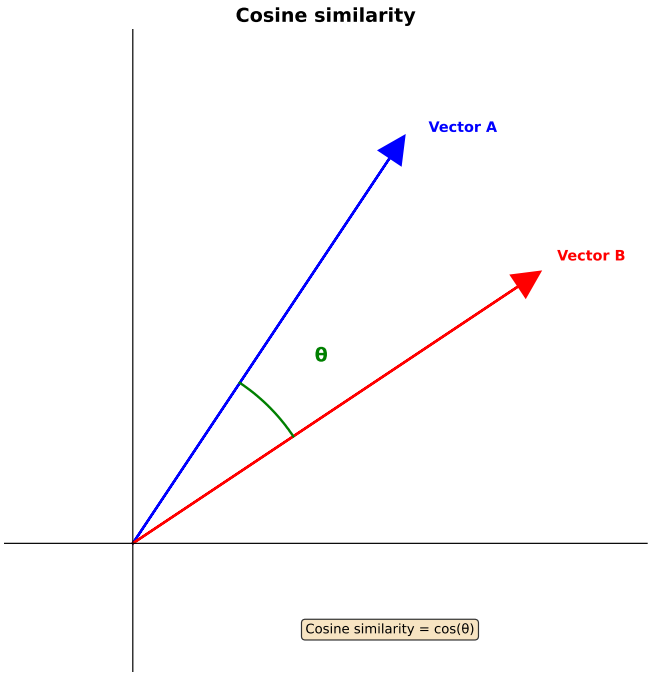
# Solution - theory

# Text embedding



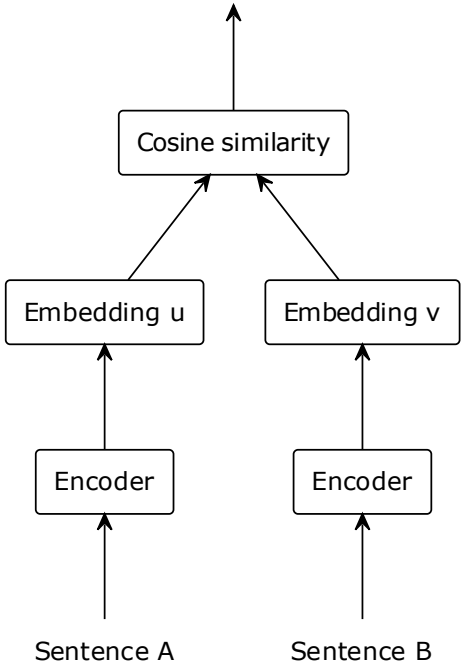

- **Text embedding:**
  - Numerical vectors
  - Multi-dimensional space
  - Similar meaning → similar vectors

# Vector similarity search




- **Cosine similarity:**
  - 1 – highly similar
  - 0 – unrelated

# Bi-Encoder


Cosine similarity
Embedding u
Embedding v
Encoder
Encoder
Sentence A
Sentence B

# Solution - implementation

# Sentence Transformers

PyPI

Pre-trained model

Embeddings (384D)

Similarity

```python
from sentence_transformers import SentenceTransformer

# 1. Load a pretrained Sentence Transformer model
model = SentenceTransformer("all-MiniLM-L6-v2")

# The sentences to encode
sentences = [
    "The weather is lovely today.",
    "It's so sunny outside!",
    "He drove to the stadium.",
]

# 2. Calculate embeddings by calling model.encode()
embeddings = model.encode(sentences)
print(embeddings.shape)
# [3, 384]

# 3. Calculate the embedding similarities
similarities = model.similarity(embeddings, embeddings)
print(similarities)
# tensor([[1.0000, 0.6660, 0.1046],
#         [0.6660, 1.0000, 0.1411],
#         [0.1046, 0.1411, 1.0000]])
```

# Text Embeddings Inference (TEI)

## Docker

```
model=Qwen/Qwen3-Embedding-0.6B
volume=$PWD/data # share a volume with the Docker container to avoid downloading weights every run

docker run --gpus all -p 8080:80 -v $volume:/data --pull always ghcr.io/huggingface/text-embeddings-inf
```

And then you can make requests like

```
curl 127.0.0.1:8080/embed \
    -X POST \
    -d '{"inputs":"What is Deep Learning?"}' \
    -H 'Content-Type: application/json'
```

# Multilingual Text Embedding Benchmark (MTEB)

| Rank (Borda) | Model | Memory Usage (MB) | Number of Parameters (B) | Embedding Dimensions | Max Tokens | Mean (Task) |
|---|---|---|---|---|---|---|
| 1 | KaLM-Embedding-Gemma3-12B-2511 | 44884 | 11.76 | 3840 | 32768 | 72.32 |
| 2 | llama-embed-nemotron-8b | 28629 | 7.505 | 4096 | 32768 | 69.46 |
| 3 | Qwen3-Embedding-8B | 14433 | 7.567 | 4096 | 32768 | 70.58 |
| 110 | all-MiniLM-L12-v2 | 127 | 0.033 | 384 | 256 | 42.28 |

# pgvector

Enable the extension (do this once in each database where you want to use it)

```
CREATE EXTENSION vector;
```

Create a vector column with 3 dimensions

```
CREATE TABLE items (id bigserial PRIMARY KEY, embedding vector(3));
```

Insert vectors

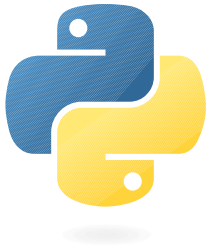
```
INSERT INTO items (embedding) VALUES ('[1,2,3]'), ('[4,5,6]');
```

Get the nearest neighbors by L2 distance

```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

Also supports inner product ( `<#>` ), cosine distance ( `<=>` ), and L1 distance ( `<+>` )

# Workflow



Get logs →

Generate embedding →

Check if new →

Create bug report →

ATLASSIAN
Jira

git bisect + revert →

# Potential improvements

- **Analyze more logs:**
  - Bigger input text
    - Truncate
    - Bigger model
  - Multiple log files
    - Merge logs
    - Select more important log
  - Embed only error signatures
- **Fine-tuning:**
  - Support domain-specific patterns e.g. numeric errors
- **Failure correlation:**
  - Link failures with different error messages, but related root cause

# Summary

# Text embedding for failure aggregation

Improve efficiency by minimizing noise
Low entry barrier (Open-source + Pre-Trained models)

**Text embedding is a low-effort way to turn CI noise into signal**

# Thank you

# Credits

- https://sbert.net
- https://github.com/huggingface/text-embeddings-inference
- https://huggingface.co/spaces/mteb/leaderboard
- https://github.com/pgvector/pgvector

- https://plantuml.com
- https://matplotlib.org