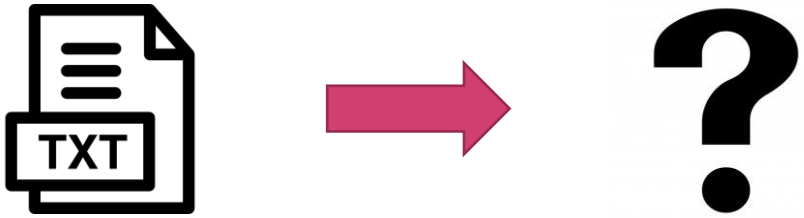
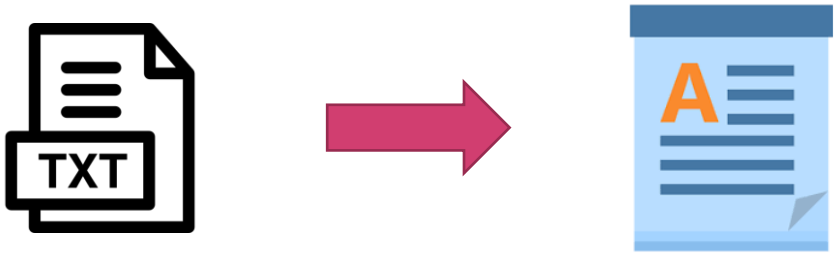
Recommendation System for Emulation environments using OpenSearch

Dakila Serasinghe
University of Freiburg

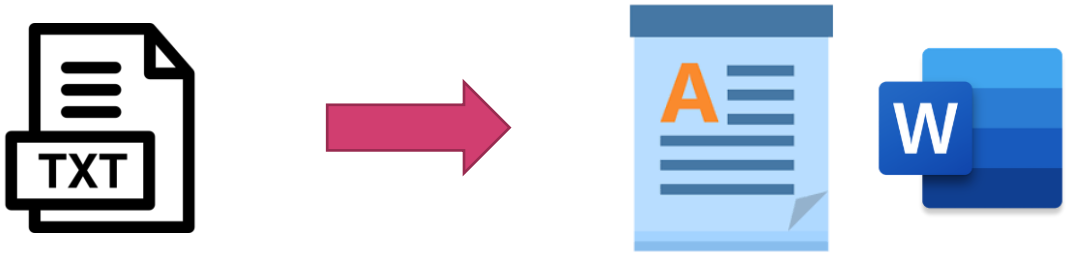
Introduction



Introduction



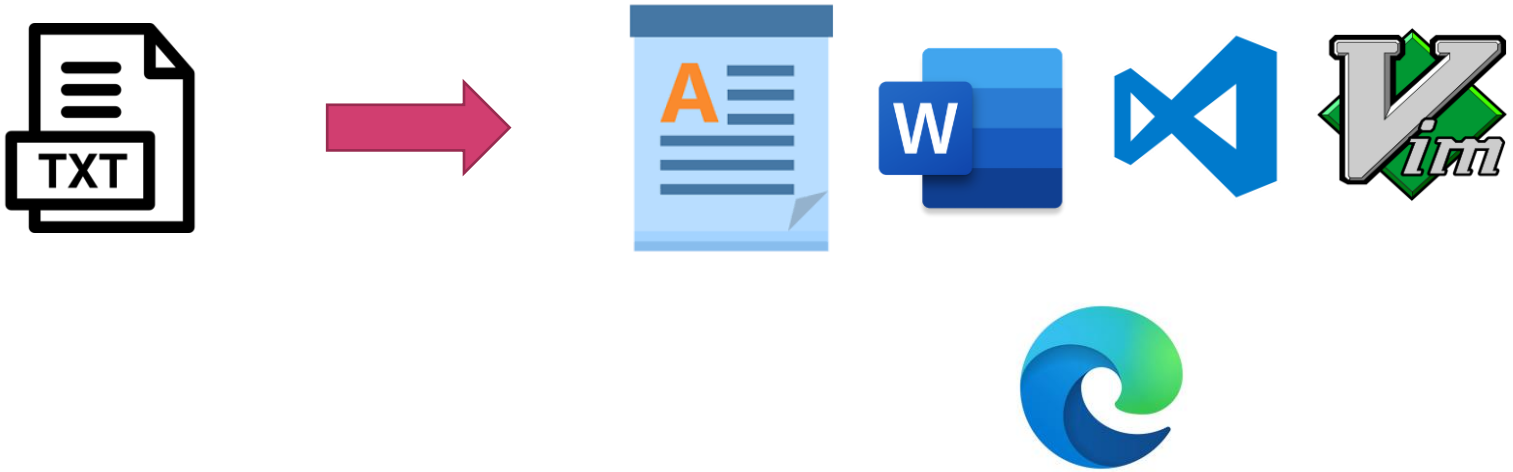
Introduction



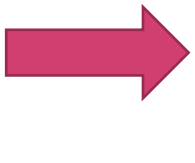
Introduction

Introduction

Introduction



Introduction



Introduction

TORQUE
GAME ENGINE

Introduction

- Archives contain (old) digital objects with obsolete file formats.
- Problem: Which file format need Which software application for rendering?
- Complexity with file types and Software dependencies make it a hard problem to choose relatable rendering environment.
- Manually choosing a suitable Environments/Applications is no longer a “Practical” solution.
- Need to Automate the process.

Introduction

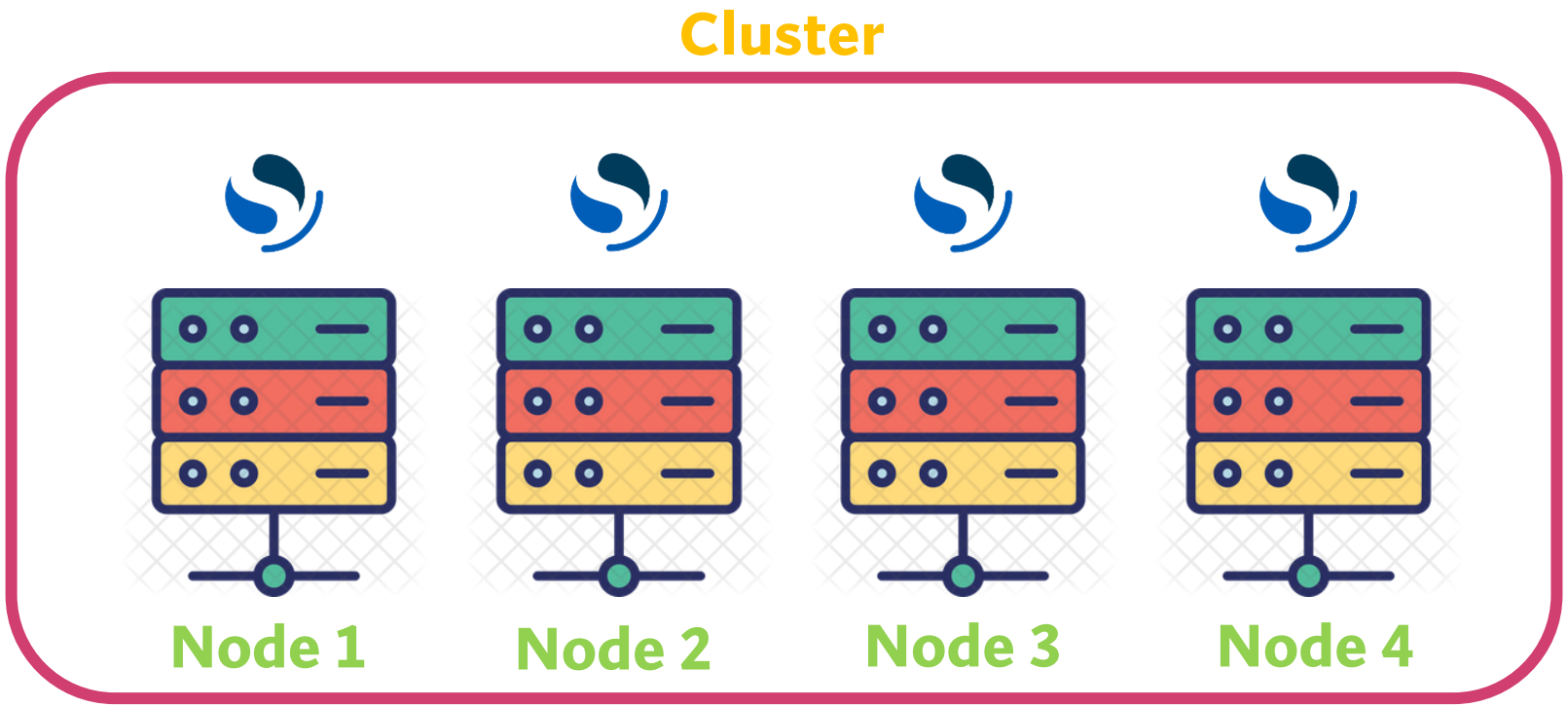
- Recommend an emulation environment -> Search Problem -> Search Engine
- Open Search
  - Off-the-Shelf solution.
  - Open-Source.
  - No Coding required.
  - No errors.

- "Automating the Selection of Emulated Rendering Environments for Born-Digital Data-Sets" – Julian Giessl, Rafael Gieshke, Klaus Rechert and Euan Cochrane

OpenSearch

- A distributed, community-driven,Apache 2.0 licensed, 100% open-source search and analytics engine.
- User can perform full-text searches: search by field, search multiple indices boost fields, rank results by score etc.
- Used for:
  - Real-time Application Monitoring
  - Log Analytics
  - Website Search

OpenSearch



Indexing Data

- Two APIs exist: the Index API and _bulk API

```
POST https://localhost:9200/_bulk

{“index”:{“_index”: “applications”, “_type”: “application”, “_id”: 1}}
{“app”: “word”, “fformat”: [“doc”, “txt”]}

{“index”:{“_index”: “applications”, “_type”: “application”, “_id”: 2}}
{“app”: “excel”, “fformat”: [“xlx”]}

{“index”:{“_index”: “applications”, “_type”: “application”, “_id”: 3}}
{“app”: “win10”, “fformat”: [“.exe”, “.dll”, “.cpl”, “.drv”, “.scr”]}

{“index”:{“_index”: “applications”, “_type”: “application”, “_id”: 4}}
{“app”: “win9”, “fformat”: [“.exe”, “.dll”, “.cpl”, “.drv”, “.scr”, “txt”]}
```

Querying Data

- Many ways of data querying available: term level, full-text, boolean

```
POST https://localhost:9200/_search

{
    "query": {
      "match": {
        "fformat": "txt"
      }
    }
}
```

Simple querying

```
POST https://localhost:9200/_search

{
    "query": {
      "multi_match": {
        "query": "exe+txt",
        "fields": ["fformat^2, "app"]
      }
    }
}
```

Advanced querying

Methodology

Approach 1: (Dual Mapping)

Search for supported applications for a file format and then search for the Environment which has that application involved.

Environment 1:	Application 1 Application 2 Application 3
Environment 2:	Application 1 Application 4

Application 1:	File Format 1 File Format 2
Application 2 :	File Format 3 File Format 4 File Format 1
Application 3 :	File Format 3
Application 4 :	File Format 4 File Format 5

Methodology

Approach 2: (Direct Mapping)

Search for supported environment for a file format.

Environment 1: Application 1
Application 2
Application 3

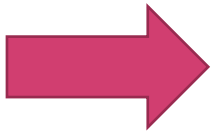
Environment 2: Application 1
Application 4

Application 1: File Format 1
File Format 2

Application 2 : File Format 3
File Format 4
File Format 1

Application 3 : File Format 3

Application 4 : File Format 4
File Format 5



Environment 1: File Format 1
File Format 2
File Format 3
File Format 4

Environment 2: File Format 1
File Format 2
File Format 4
File Format 5

Environments

Reference environment 1:

Reference environment 2:

Reference environment 3:

Reference environment 4:

Environments

- Random Environments are generated by randomly picking 107 environments from unique file formats.

- How to find all the file formats related to an application?
  - Web scraping from Wikidata.

https://www.wikidata.org/wiki/Q11261 —scrap→ app_to_fformat_map() → Readable/writable File formats

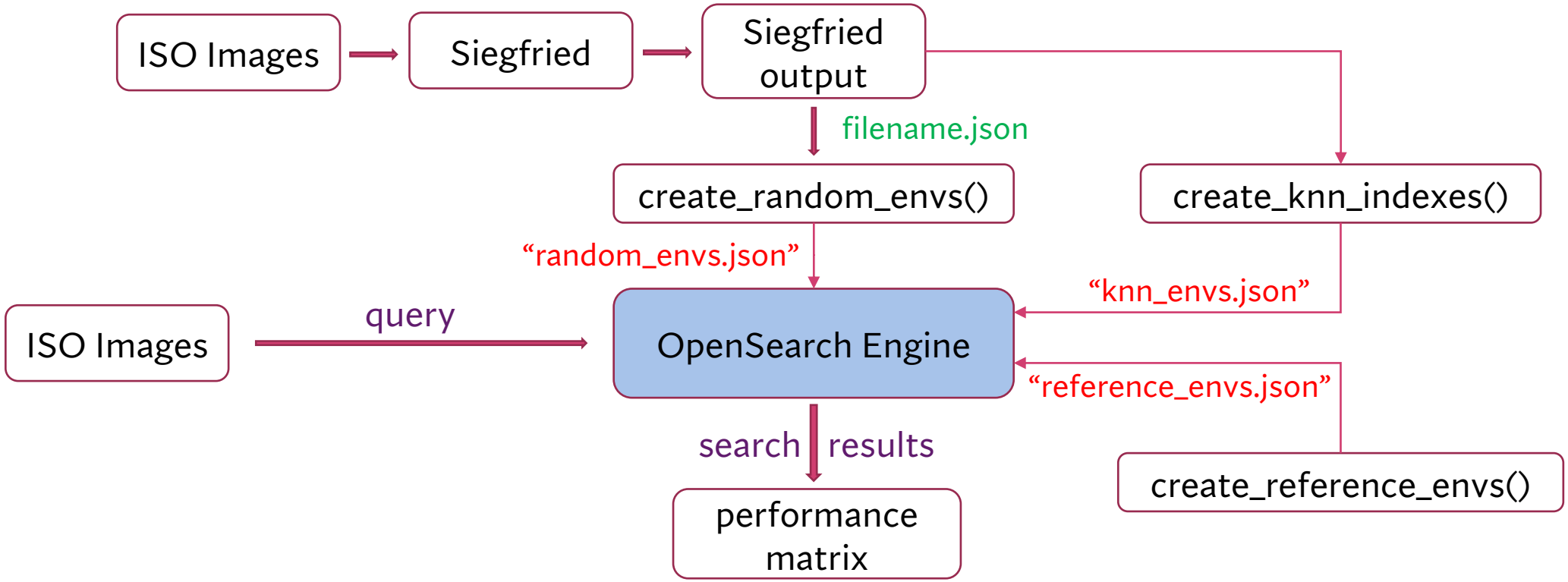
app2fformats.json

```
"Q11261": {
    "read_formats": [
        "Q686498",
        "Q3033641",
        "Q467454",
        "Q1145976",
        "Q26208225"],
    "write_formats": [
        "Q686498",
        "Q3033641",
        "Q26208225"]
}
```

Implementation

1. Create reference environments (4 fixed environments)
2. Create random environments (16 random environments)
3. Ingest environment indexes into “OpenSearch” engine
4. Query for digital objects

Implementation



k-NN Search plugin

- K-nearest neighbors, enable users to search for the k-nearest neighbors to

a query point across the index of vectors.

- User can specify the space (the distance function) to measure the distance between two points.

Distance functions

spaceType	Distance Function	OpenSearch Score
l2	$Distance(X, Y) = \sum_{i=1}^{n} (X_i - Y_i)^2$	1 / (1 + Distance Function)
l1	$Distance(X, Y) = \sum_{i=1}^{n} (X_i - Y_i)$	1 / (1 + Distance Function)
linf	$Distance(X, Y) = Max(X_i - Y_i)$	1 / (1 + Distance Function)
cosinesimil	$1 - \frac{A \cdot B}{\lVert A \rVert \cdot \lVert B \rVert} = 1 - \frac{\sum_{i=1}^{n}(A_i \cdot B_i)}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$ where $\lVert A \rVert$ and $\lVert B \rVert$ represent normalized vectors.	1 / (1 + Distance Function)
innerproduct	$Distance(X, Y) = -A \cdot B$	if (Distance Function >= 0) 1 / (1 + Distance Function) else - Distance Function + 1

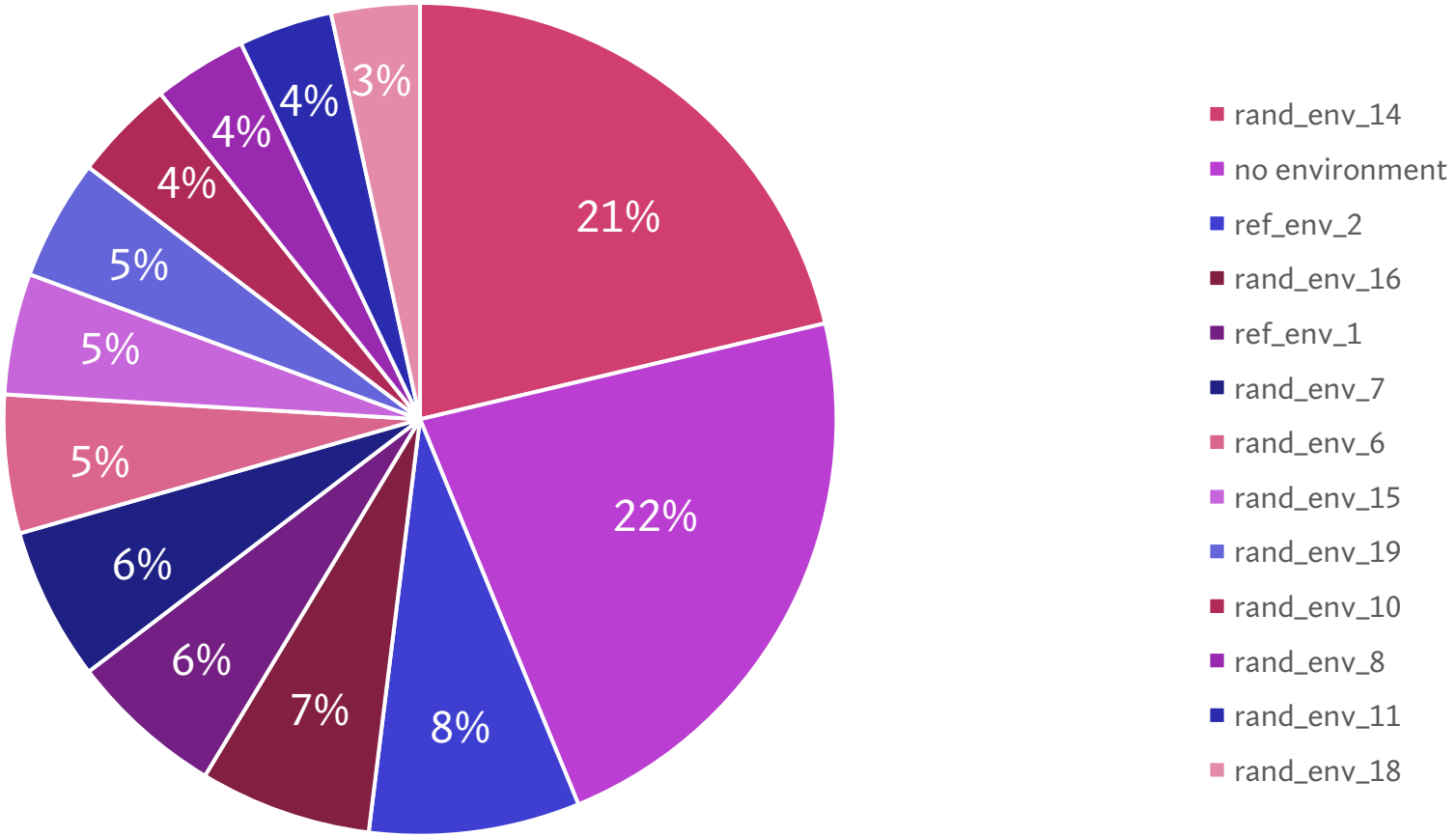
k-NN Vector Index

- Assume the entire environment space supports 8 unique file formats
- Environment 1 only could support 5 of them. (F0, F2, F3, F4, F7)
- Environment 2 supports 6 formats. (F0, F1, F2, F4, F5, F6)
- Query image has 4 file formats. (F1, F2, F4, F7)

	F0	F1	F2	F3	F4	F5	F6	F7
E0	1	0	1	1	1	0	0	1
E1	1	1	1	0	1	1	1	0
Query	0	1	1	0	1	0	0	1

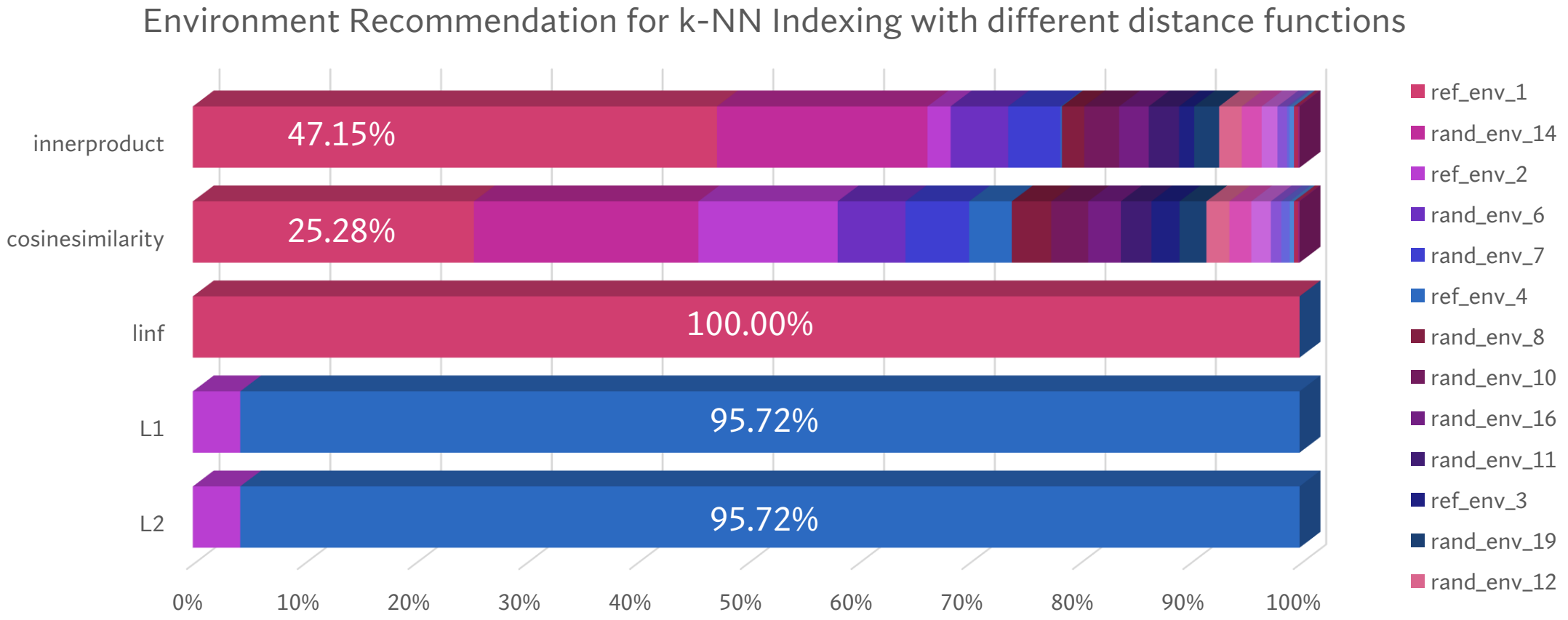
- k-NN index vectors has approx. 990-1200 unique file formats

Standard indexed Data Results – BM25

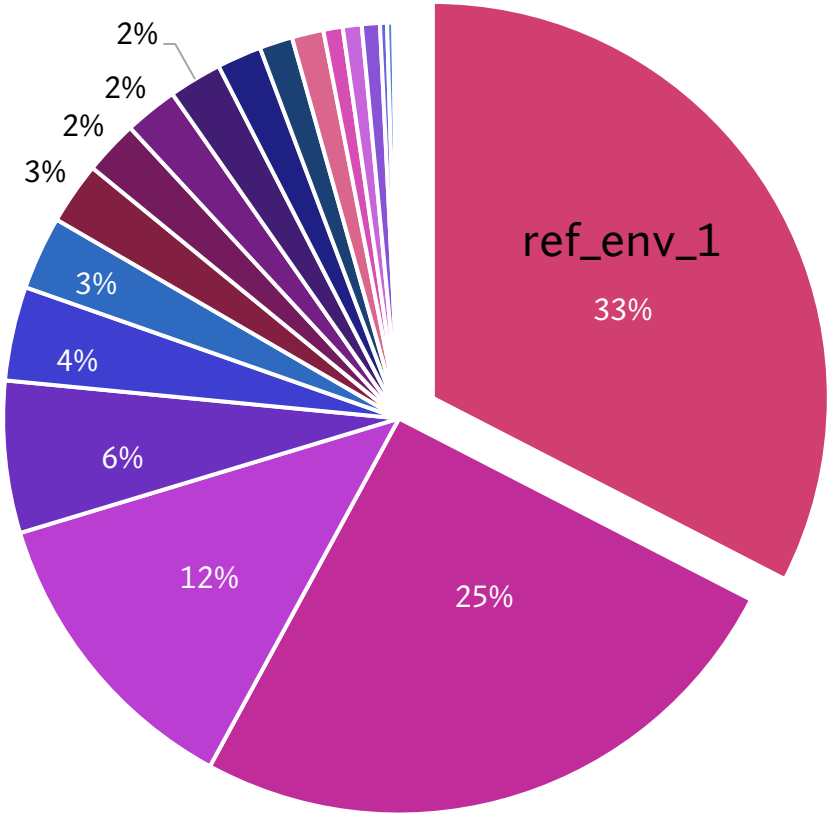


Environment Recommendation for Standard OpenSearch Indexing

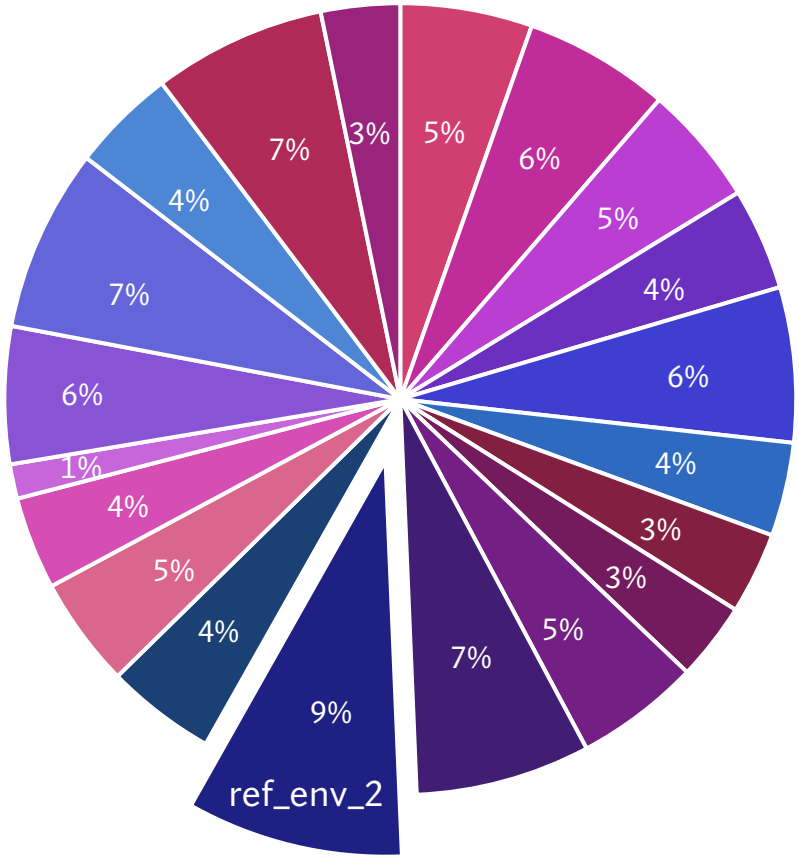
k-NN Indexed Data Results



k-NN(innerprod) vs Standard (50 runs)

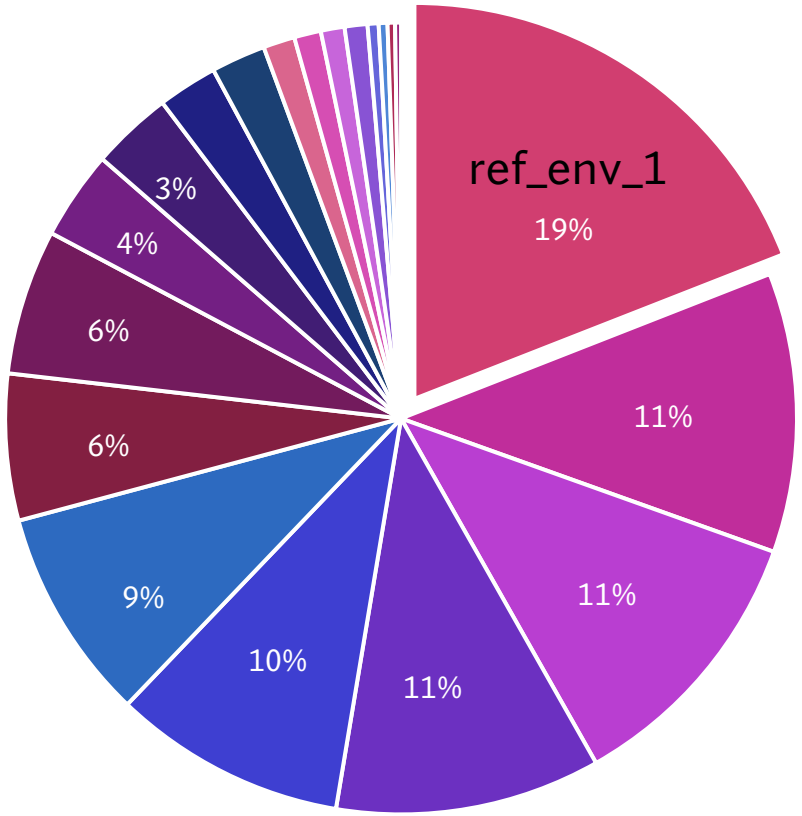


k-NN similarity (innerproduct)

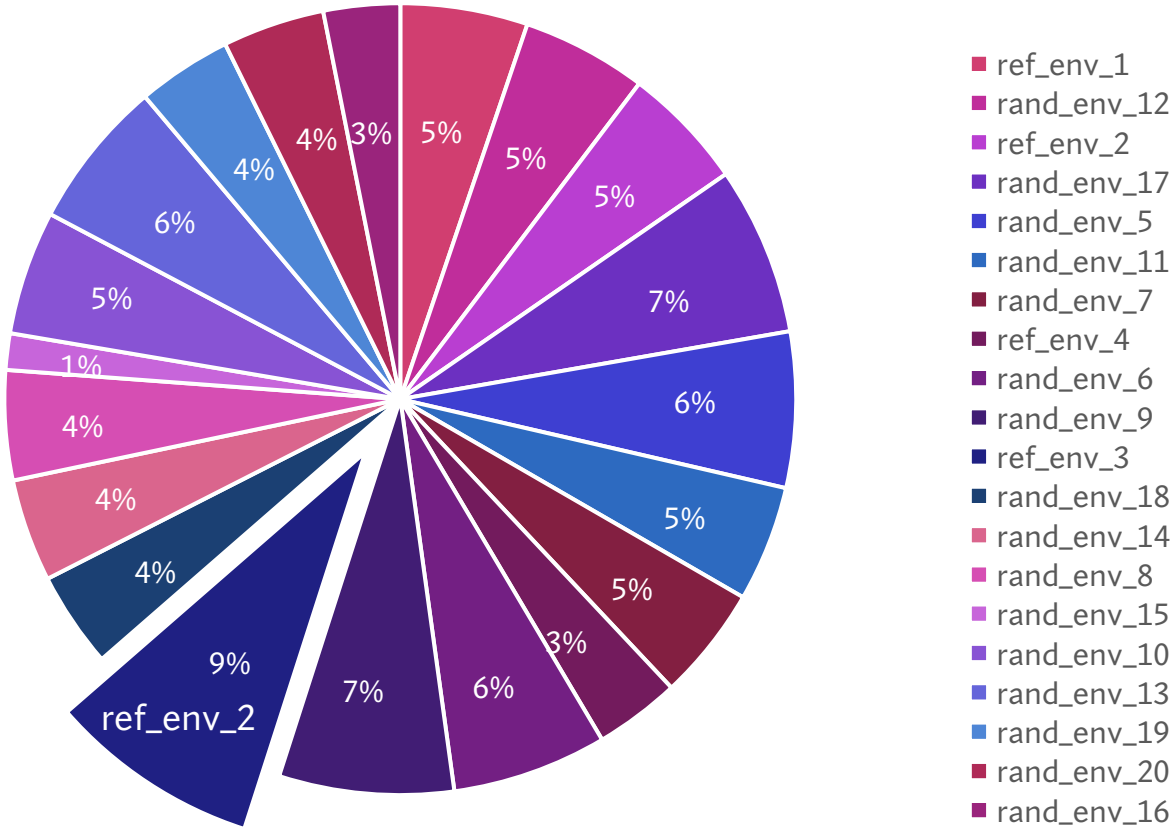


Standard Indexed data results

k-NN(cosinesim) vs Standard (50 runs)

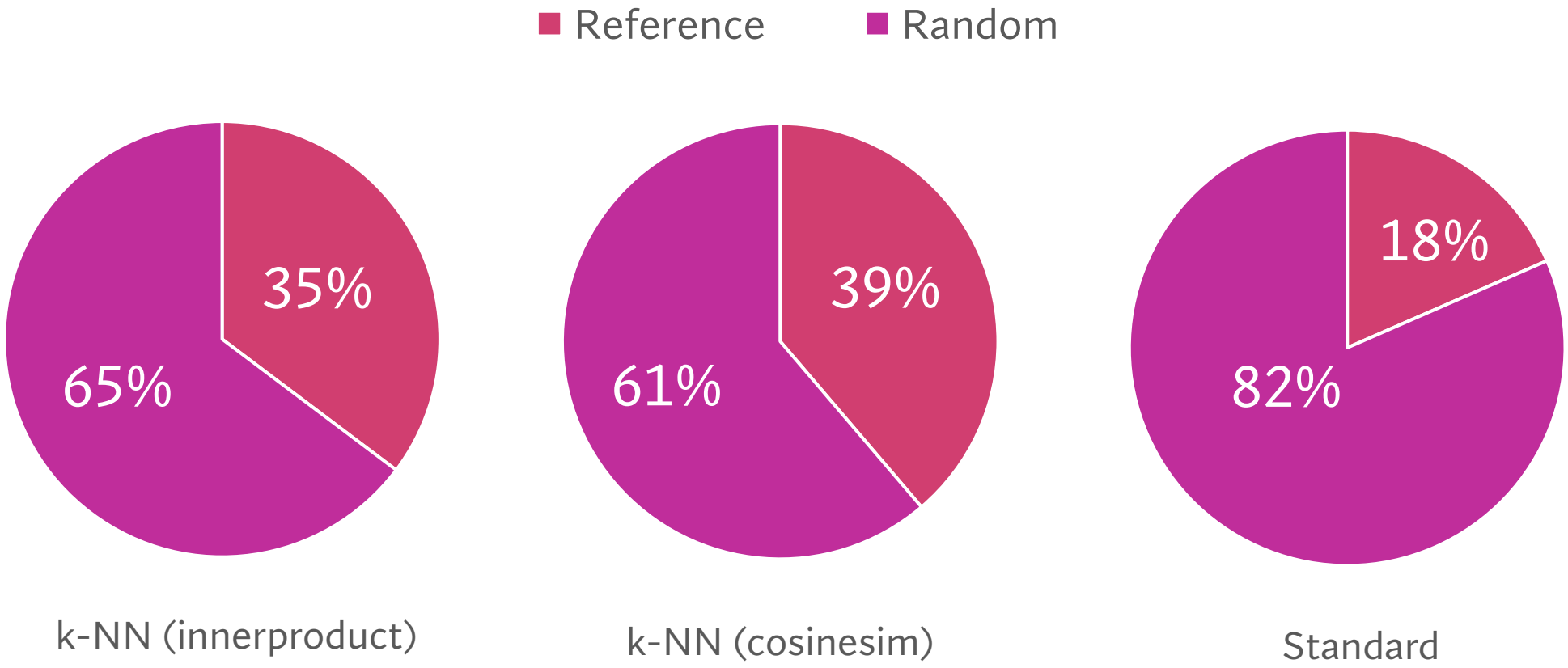


k-NN similarity (innerproduct)



Standard Indexed data results

Reference Vs Random (50 runs)



Conclusion

- This work presents a Search Engine in OpenSearch which can recommend software rendering environments for born-digital data-sets.
- k-NN searches always recommend reference environments top.
- Standard Search results could be enhanced by thorough inspection of environment creation.
- k-NN Search can be enhanced by modifying index vectors to have less dimensions.
- Standard(BM25) Search results can be improved by reducing the “less-used” file formats in environments.

Thank You!