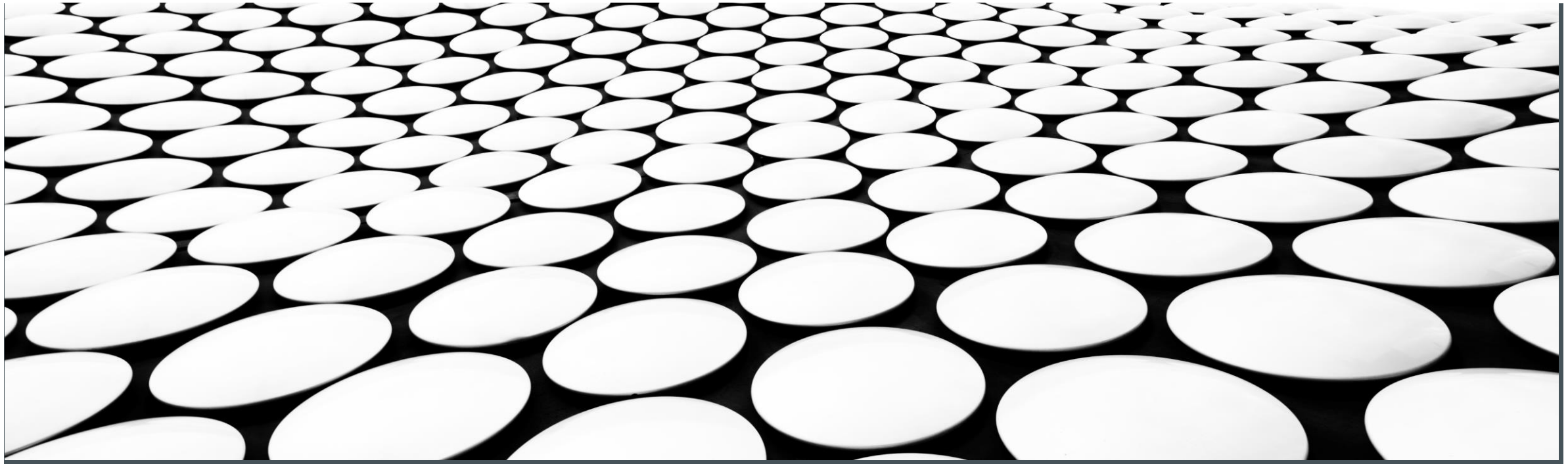
CSCI E-29 FINAL PROJECT

- NEW IMPLEMENTATIONS AND IMPROVEMENTS FOR EDUCATIONALWEB

YAN CUI



AGENDA

- Introduction
  - Goal
  - Architecture
  - Live Demo
- New Implementation
  - PDF Navigator
  - Keyword Search
  - Slides Recommendation
- Improvements With Advanced Python
- Future Development

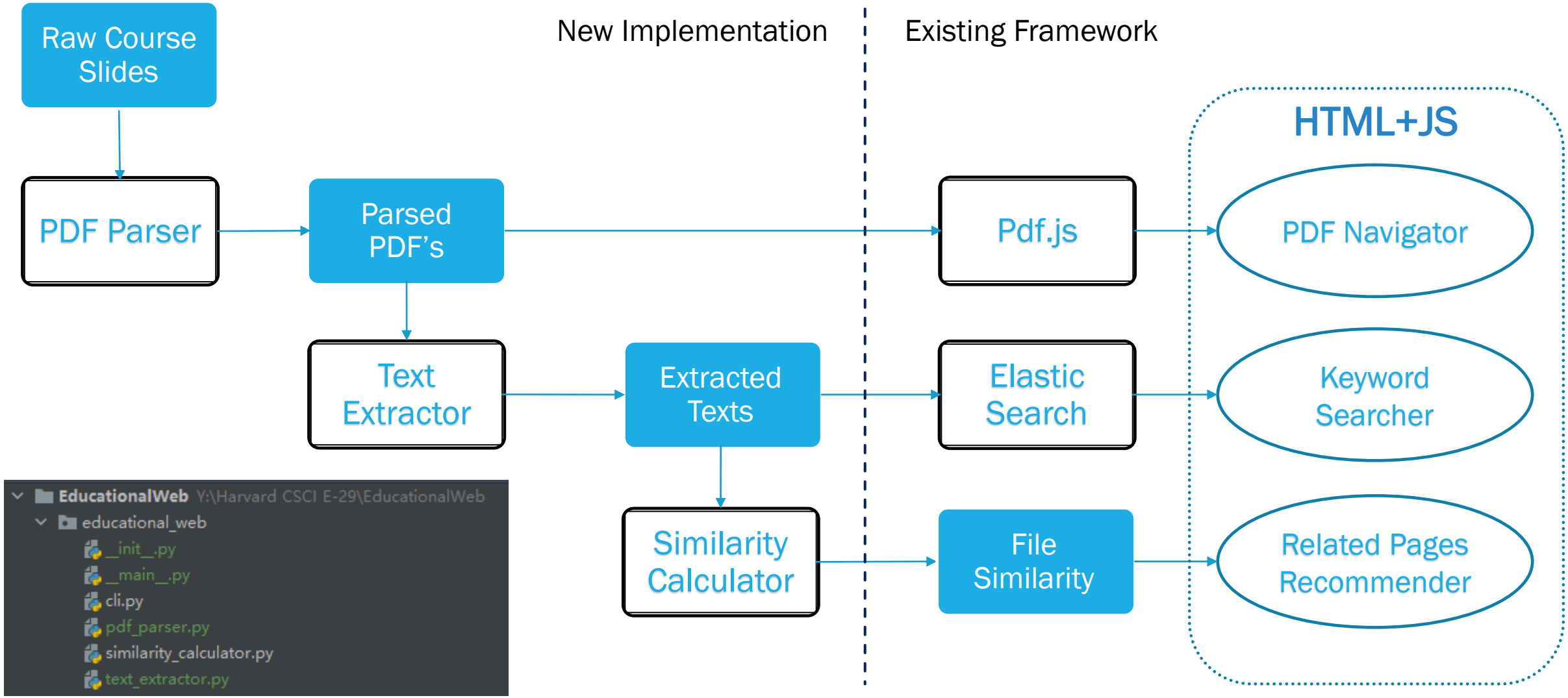
INTRODUCTION

GOAL OF THE PROJECT

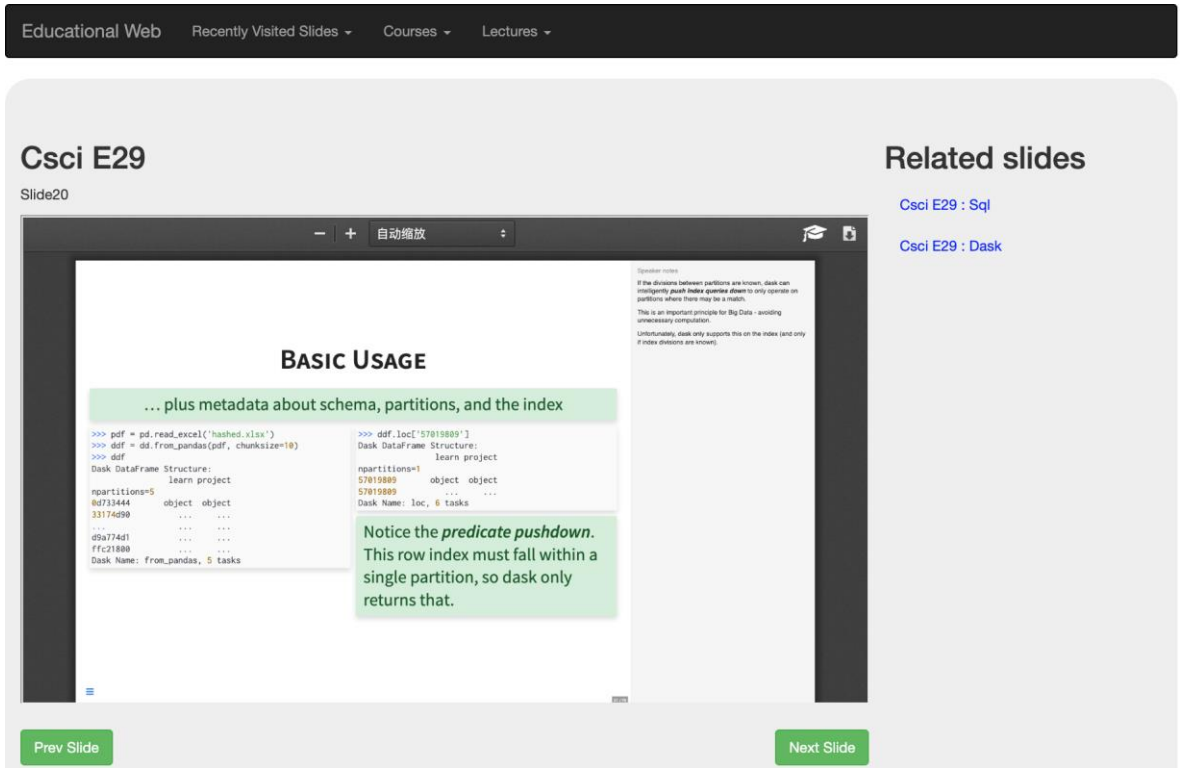
- This project derived from an open-source application provided by my another course: EducationalWeb. It provides functions to navigate slides, do keyword search and recommend related pages from sourced pdf documents, specifically course slides to help students learn and prepare for the exam:

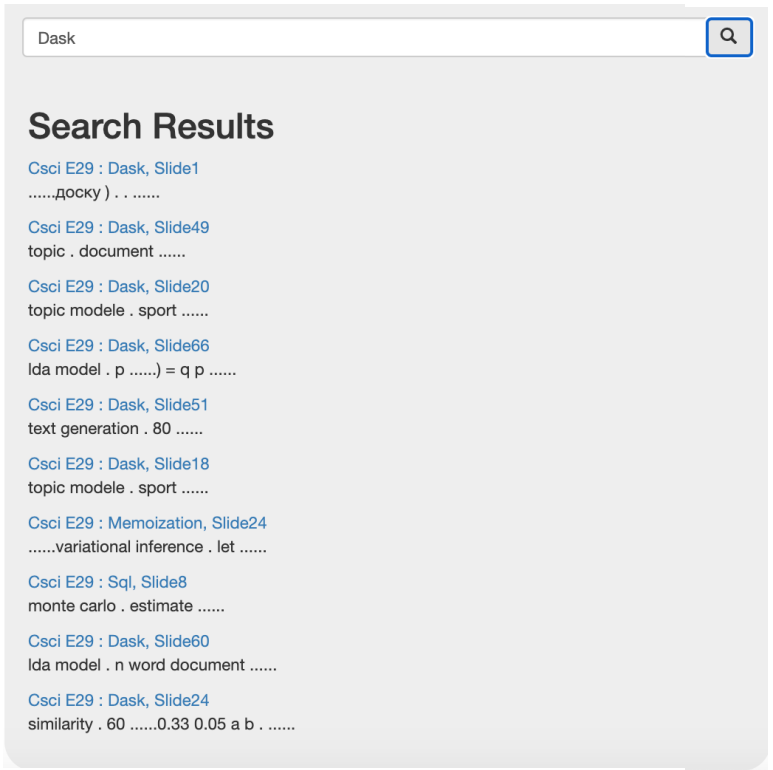
- **Navigate** presentation slides (PDF) by course and lecture name
- **Search** key words in all the PDF slides and return ranked result of related pages
- **Recommend** related slides that contains the similar topic based on current displayed slide
- **Explain*** any word(s) selected in current slide in a machine learning composited paragraph from the text book or a corpus of documents (under development).

ARCHITECTURE



LIVE DEMO


Educational Web
Recently Visited Slides
Courses
Lectures
Csci E29
Slide20
自动缩放
Basic Usage
… plus metadata about schema, partitions, and the index
Notice the predicate pushdown. This row index must fall within a single partition, so dask only returns that.
Related slides
Csci E29 : Sql
Csci E29 : Dask
Prev Slide
Next Slide


Dask
Search Results
Csci E29 : Dask, Slide1
.....доску) . . ..
Csci E29 : Dask, Slide49
topic . document .....
Csci E29 : Dask, Slide20
topic modele . sport .....
Csci E29 : Dask, Slide66
lda model . p .....) = q p .....
Csci E29 : Dask, Slide51
text generation . 80 .....
Csci E29 : Dask, Slide18
topic modele . sport .....
Csci E29 : Memoization, Slide24
.....variational inference . let .....
Csci E29 : Sql, Slide8
monte carlo . estimate .....
Csci E29 : Dask, Slide60
lda model . n word document .....
Csci E29 : Dask, Slide24
similarity . 60 .....0.33 0.05 a b . .....

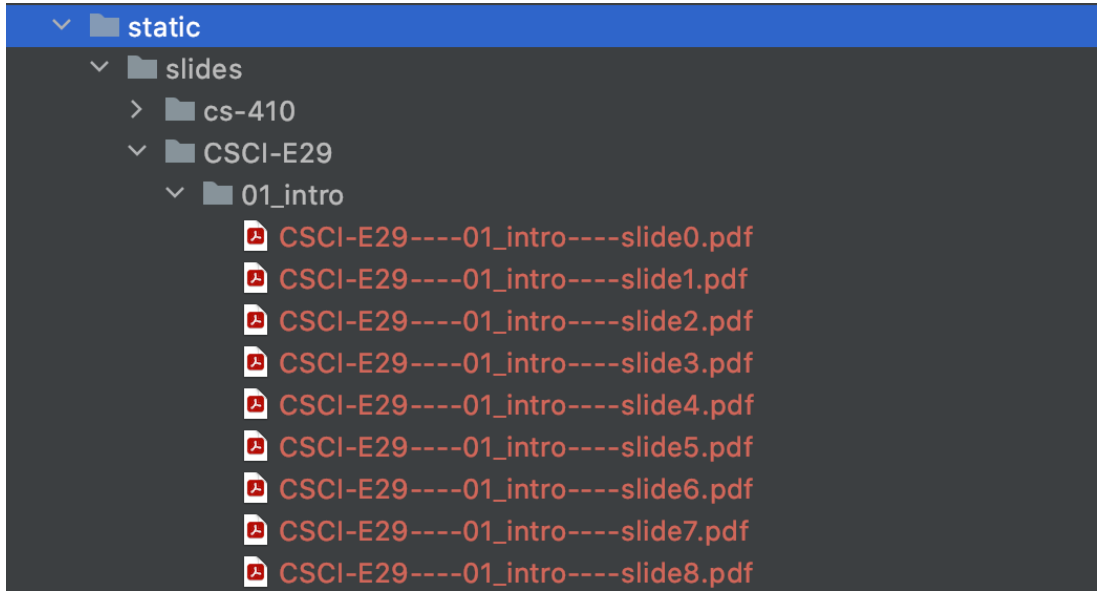
NEW IMPLEMENTATION

PDF NAVIGATOR

- Source data of slides are downloaded from course website and saved on **AWS S3** as ExternalTask.
- The **pdf_parser** pull the raw slides from S3 and parse into single page files by PyPDF2. Those files are saved as Luigi TargetOuput in a course — lecture — page architecture.
- **Pdf.js** is a JavaScript library that renders PDF files using web standards-compliance HTML5 Canvas. It consumes those the parsed slides and run a pdf viewer server at https:/localhost:8888.

pdf_parser.py

```
9   class raw_slides(ExternalTask):...
14
15  class download_slides(Task):...
31
32  class parse_pdf_to_single_page(Task):...
```



KEYWORD SEARCH

- **Text_extractor** relies on previous task and extracts text information from each parsed pdf with two steps:
  I. Use textract package to extract a line of text string from each pdf file;
  II. Use metapy package to create a tokenizer that processes the string to a list of words: length filtering, stop words filtering, stemming, etc.
- The text string from each file is saved in a separate line, corresponding to its label at the same position in another file (both using **atomic_write**), which also links to the PDF viewer.
- The two data files are then indexed and fed into **ElasticSearch**, which is a distributed full-text search engine that provides an HTTP web interface.

text_extractor.py

```
13  def tokenizer(str):...
23
24  class extract_text(Task):
25      LOCAL_ROOT = r'slides'
26      course_name = Parameter('CSCI-E29')
27      requires = Requires()
28      data = Requirement(parse_pdf_to_single_page)
29
30      output = TargetOutput(file_pattern=LOCAL_ROOT+'/{task.course_name}', ext='')
31
32      def run(self):
```

Slides_CSCI-29.dat

```
581  speaker notes assigning variable python affects actual da
582  speaker notes list changing values xextend mutable object
583  speaker notes numpy arrays stored contiguous memory array
584  speaker notes confusing bit expectation modify object hic
585  speaker notes notes slide dierent case hash its sha explo
586  speaker notes notes slide luigi workflow assumes immutabl
587  speaker notes actual functions deterministic outputs dete
```

slides_CSCI-29.dat.labels

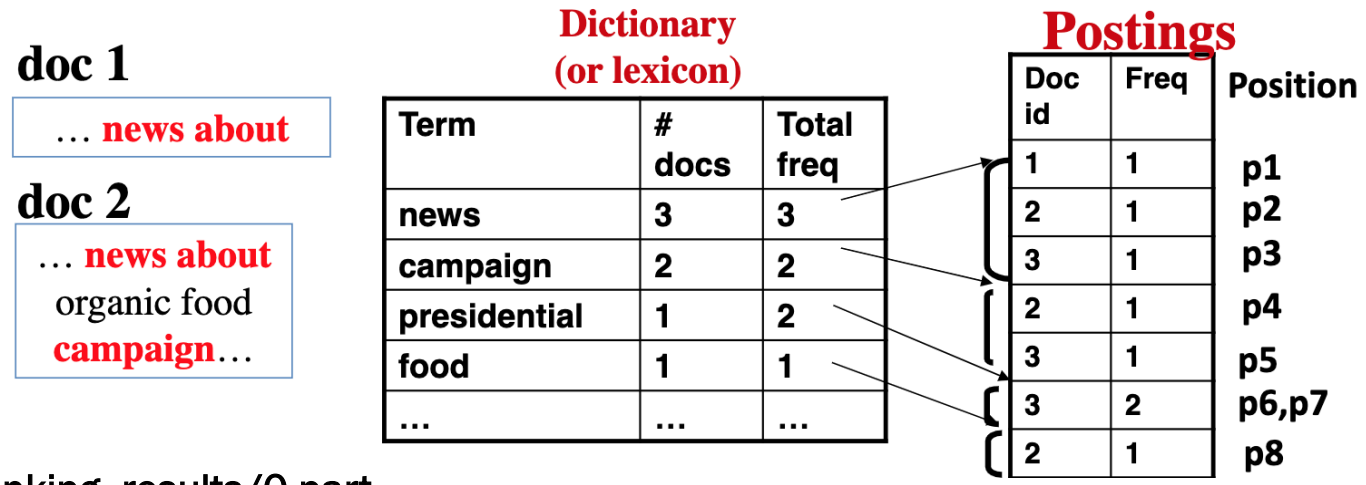
```
581  CSCI-E29##09_sql##slide32
582  CSCI-E29##09_sql##slide33
583  CSCI-E29##09_sql##slide34
584  CSCI-E29##09_sql##slide35
585  CSCI-E29##09_sql##slide36
586  CSCI-E29##09_sql##slide37
587  CSCI-E29##09_sql##slide38
```

SLIDES RECOMMENDATION

- **Similarity_calculator** consumes the extracted texts and calculates the relationship between each pair of documents based on their similarities
- Those documents are converted to a data structures called **Inverted Index** that enable fast search (precomputing as much as we can)
- Use **BM25** to compute the relation with:
  - Term frequency transformation
  - Inverted document frequency
- The top 10 related documents with **normalized** similarity are saved as Dask Dataframes CSVTarget. The related slides up to certain threshold will be recommended on the right side of currently viewing PDF page.

similarity_calculator.py

```
14  class similarity_calc(Task):
15      LOCAL_ROOT = r'static/ranking_results'
16      course_name = Parameter('CSCI-E29')
17      requires = Requires()
18      data = Requirement(extract_text)
19
20      output = TargetOutput(file_pattern=os.path.join(LOCAL_ROOT,'{task.course_name}')+r'/',
21                            ext='', target_class=CSVTarget)
22
23      def run(self):
```

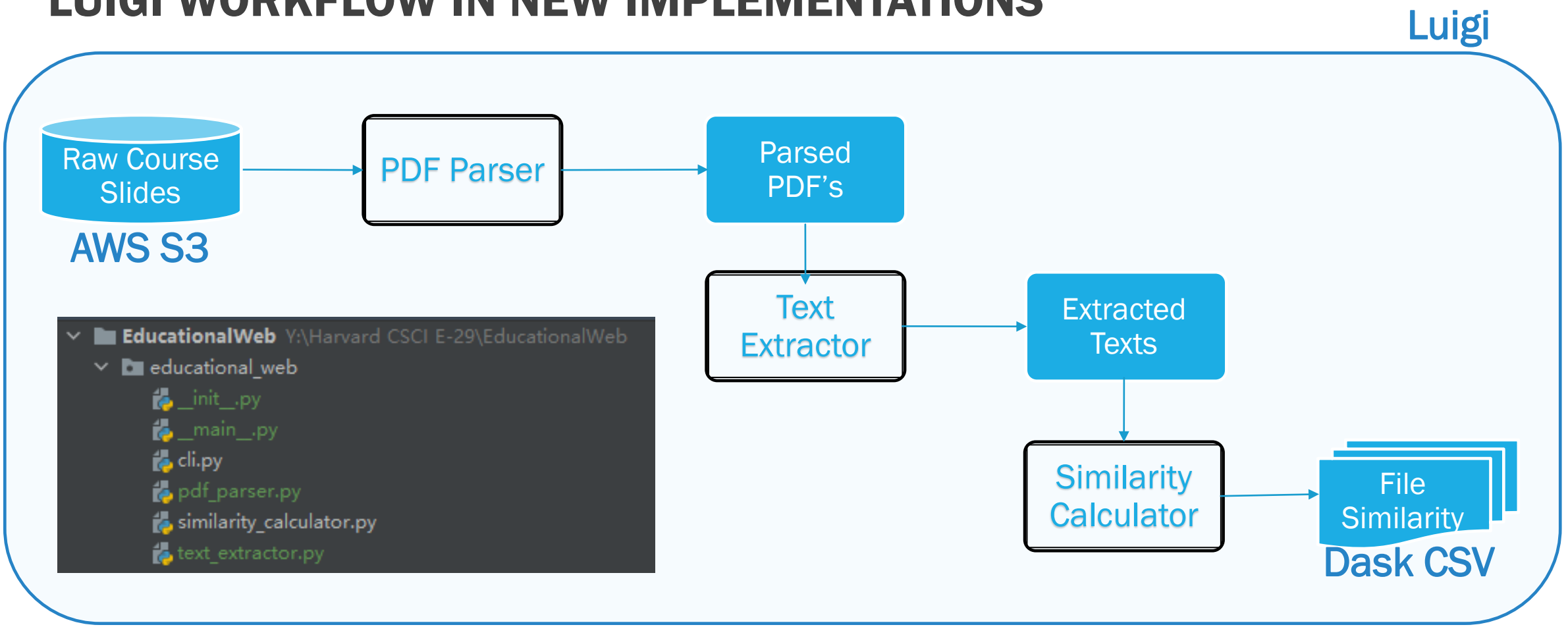


ranking_results/0.part

```
581  CSCI-E29##09_sql##slide33,0.36206248311725897,CSCI-E29##09_sql##slide35,0.3046
582  CSCI-E29##09_sql##slide32,0.29453458105379793,CSCI-E29##12_memoization##slide5
583  CSCI-E29##12_memoization##slide14,0.5622702619670046,CSCI-E29##11_misc##slide7
584  CSCI-E29##09_sql##slide32,0.28556128115619295,CSCI-E29##09_sql##slide33,0.2413
585  CSCI-E29##12_memoization##slide61,0.2708237226489156,CSCI-E29##04_iteration##s
586  CSCI-E29##09_sql##slide38,0.2608773286828653,CSCI-E29##06_graphs##slide21,0.25
587  CSCI-E29##02_continuous_science##slide50,0.31595559746862617,CSCI-E29##02_cont
```

IMPROVEMENTS WITH ADVANCED PYTHON

LUIGI WORKFLOW IN NEW IMPLEMENTATIONS



OTHER ADVANCED PYTHON IMPROVEMENTS

Previous Framework	Improvements
Static requirements.txt file	Create pipenv file to management version of the dependencies
No test cases in existing code	CI/CD will be covered with testing
Just final data	Use Atomic writes to create inputs
Hard coded variables	Use .env to store AWS credentials and environment variables
Data needs manual download	Source data lives in AWS S3 and downloads with Luigi tasks
Store results in plain csv file	Use Dask targets to save dataframes

FUTURE DEVELOPMENT

TECHNICALLY

- Complete the text explanation part that composites a ML generated paragraph from text book
  - Generate TF-IDF data of slides documents
  - Train a predictive language model
- Rewrite the entire framework (including ElasticSearch, PDF.js, etc) as Luigi workflow
- Use Django and DRF to store data and provide data request

IN APPLICATION

- Expand the search and recommendation functions to more areas beyond certain courses
  - Paper/documents needed during research
  - Webpage info/news interested in business

THANKS