Syntax Sentinels

Team 2

An NLP-based code plagiarism detector

Presentation Roadmap

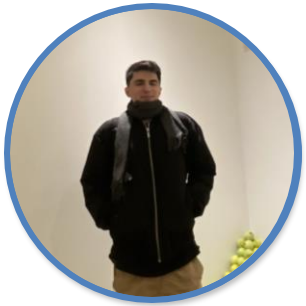
- **Meet the Team** – Introduction to team members.
- **Existing Solutions: MOSS** – Strengths and limitations.
- **Why It Matters** – Importance of effective plagiarism detection.
- **Problem Statement** – Challenges with current solutions and our motivation.
- **Demo** – Live demonstration of the system.
- **Features** – Overview of our NLP-based plagiarism detection system.
- **System Architecture** – Technical breakdown of how our system works.
- **Model Design** – The key components of our model
- **Q&A** – Open floor for questions.

Meet the Team

Dennis Fong

Lucas Chen

Mohammad Mohsin Khan



Luigi Quattrociocchi

Julian Cecchini

PICTURE THIS!

Existing Solution: MOSS (Measure of Software Similarity)

- **Strengths:**
  - Effective for **exact and near-exact code matches**.
  - Widely used in **academia** and **coding competitions**.
  - Works across multiple programming languages.
- **Limitations:**
  - **Syntax-based approach** – Cannot detect **semantic plagiarism** (e.g., logic-preserving transformations).
  - Struggles with **function reordering, and code restructuring**.
  - **High false negatives** – Clever modifications can evade detection.

Why It Matters

- **Academic Integrity** – Ensures fairness in grading and protects originality.
- **Code Competitions** – Detects unfair advantages in submissions.
- **User Trust** – Minimizes false positives/negatives for reliable detection.

Problem statement

Academic institutions, coding competition organizers, and online platforms need a more effective plagiarism detection system because current tools like MOSS rely on syntax-based comparisons. Our NLP-based solution detects semantic plagiarism, providing a more accurate, customizable, and scalable system.

Demo

Features

Comprehensive Detection via an NLP approach

Semantic Understanding

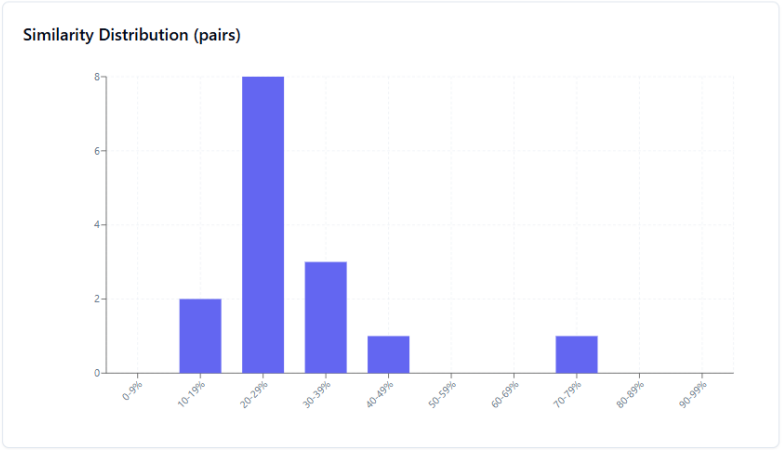
- Unlike traditional tools that rely on **syntactic** matching (such as MOSS), our system uses **Natural Language Processing (NLP)** to understand the **semantic** of the code.

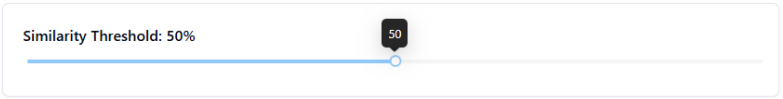
Why is this better

- This makes our product more effective at catching **advanced plagiarism** techniques like **code obfuscation, inlining, block reordering and variable renaming**.
- All in all, this improves detection of **logic-preserving plagiarism** where code structure is changed but the functionality remains the same.

Detailed Reports

- Provides comprehensive reports which includes:
  - Similarity scores as a histogram
  - Line by line analysis of file pairs
- Helps users understand plagiarism insights clearly.






```python
def scoring(d, t, k=10):
    arr = [l for l in last]
    arr[t] = d
    penalty = sum((d - l) * c for l, c in zip(arr, (
    score = S[d][t] - penalty
    score -= (min(D, d + k) - d - 1) * penalty
    return score

D = int(input())
C = list(map(int, input().split()))
S = []
for _ in range(D):
    s = list(map(int, input().split()))
    S.append(s)
last = [-1] * 26
ans = []
for i in range(D):
    res = -10 ** 6
    idx = -1
    for j in range(26):
        tmp = scoring(i, j)
        if tmp > res:
            res = tmp
            idx = j
    last[idx] = i
    ans.append(idx+1)
print(*ans, sep='\n')
```

```python
import time
import copy

start_time = time.time()

#input
D = int(input())
c_list = list(map(int, input().split()))

s_grid = []
for i in range(D):
    array = list(map(int, input().strip().split('
    s_grid.append(array)

def calculate_score(d,t,last):
    score = s_grid[d][t]
    last[t] = -1
    for i in range(26):
        score -= c_list[i]*(last[i]+1)
    return score

t_list = [] #task_list
last_list = [0] * 26

for k in range(0,D):
    X = -1  # k-日目に変える番号を探す
    p = 0
    for i in range(26):  # 26通り試す
        tmp = calculate_score(k,i,last_list)
        if tmp > p:
            X = i
```

Why these features matter?

Streamlines the user experience of educators in maintaining academic integrity.

Improves precision and reduces false positives with NLP.

Visualizations provide a high-level overview of plagiarism within a corpus.

Changes to requirements documentation

Zero Data Retention (ZDR)

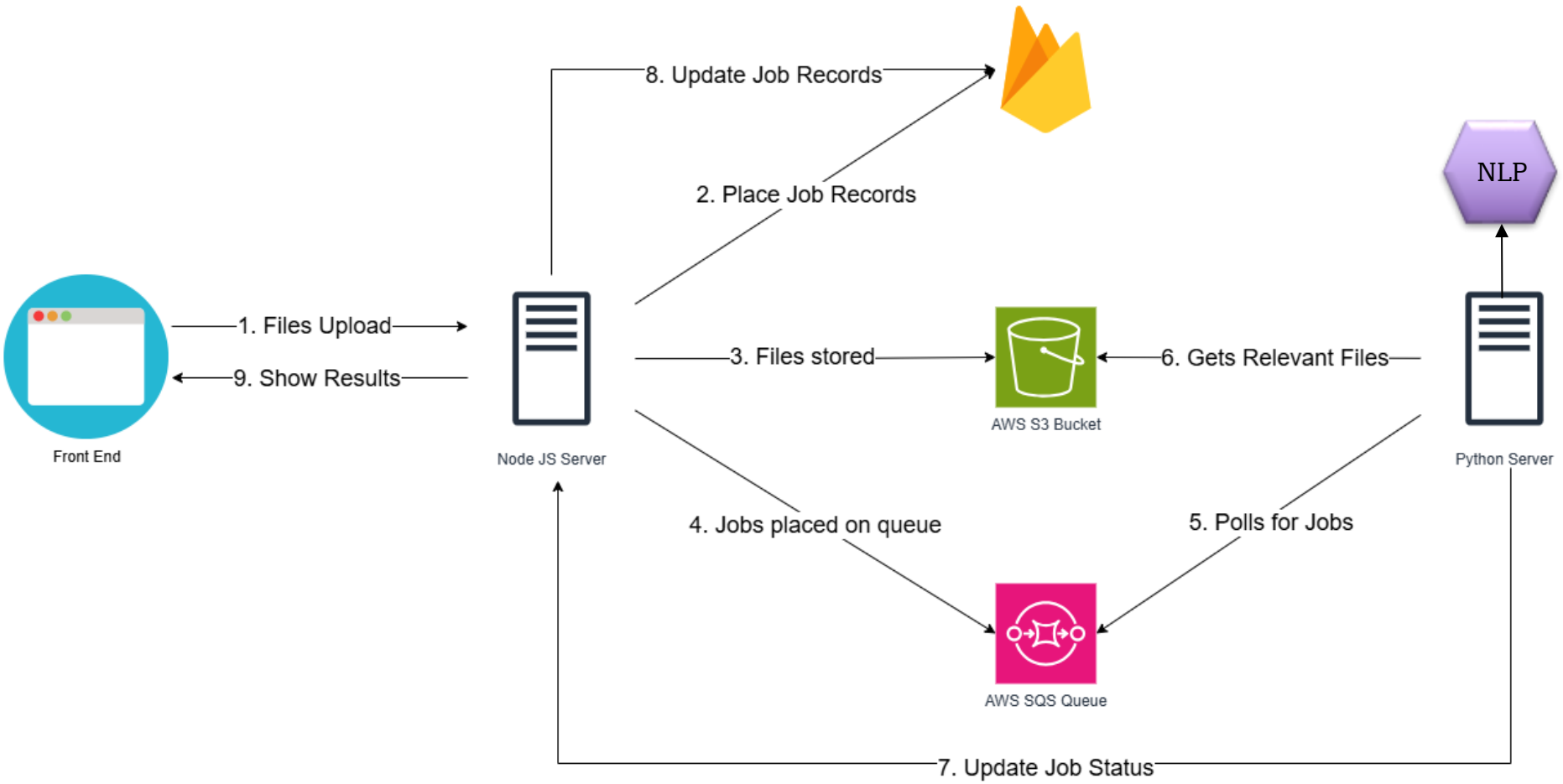
- This was limiting from a user perspective.
- Eliminating this requirement helped to streamline the user experience.

Emailing

- The user flow for our revision 0 (receiving an email with your results) was not intuitive.
- Instead of sending the results in a zip file, we have a button to download the results.

Architecture

System Architecture



Firebase - Firestore

What is Firestore?

- Firestore is a **NoSQL database**
- Firestore lives on the **cloud**, and is built on Google's infrastructure

Why Firestore?

- We wanted to utilize a NoSQL database due to the unstructured nature of the data contained in a job
- **Real Time Sync** – Connected clients receive updated data in real time
    - Other NoSQL databases such as DynamoDB or MongoDB require extra set up and cost to make this possible
- **No Management Required** – Firestore is fully managed by Firebase. Scaling read/write capacities are done automatically
    - No need to manage read/write capacities (in DynamoDB), or manage clusters (in MongoDB)

Simple Storage Service (S3)

What is S3?

- Low cost, high-capacity object storage
- Enables fast GET operation on objects

Why S3?

- Zero cost for use cases this product was designed for
- Can be scaled up to store much more data, while still costing almost nothing
    - To spend 1 cent, you'd need to upload 10,000 files approximately 43.5 KB each
- **Extremely Durable** – objects in S3 are stored across multiple availability zones, meaning data can be recovered even through disaster
- **Highly Secure** – Data is encrypted, and secured by permissions set by admins

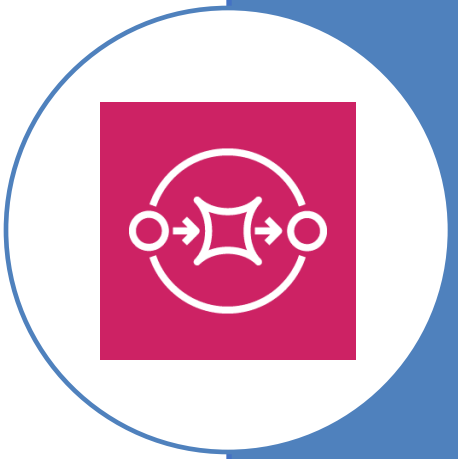
Simple Queue Service (SQS)

What is SQS?

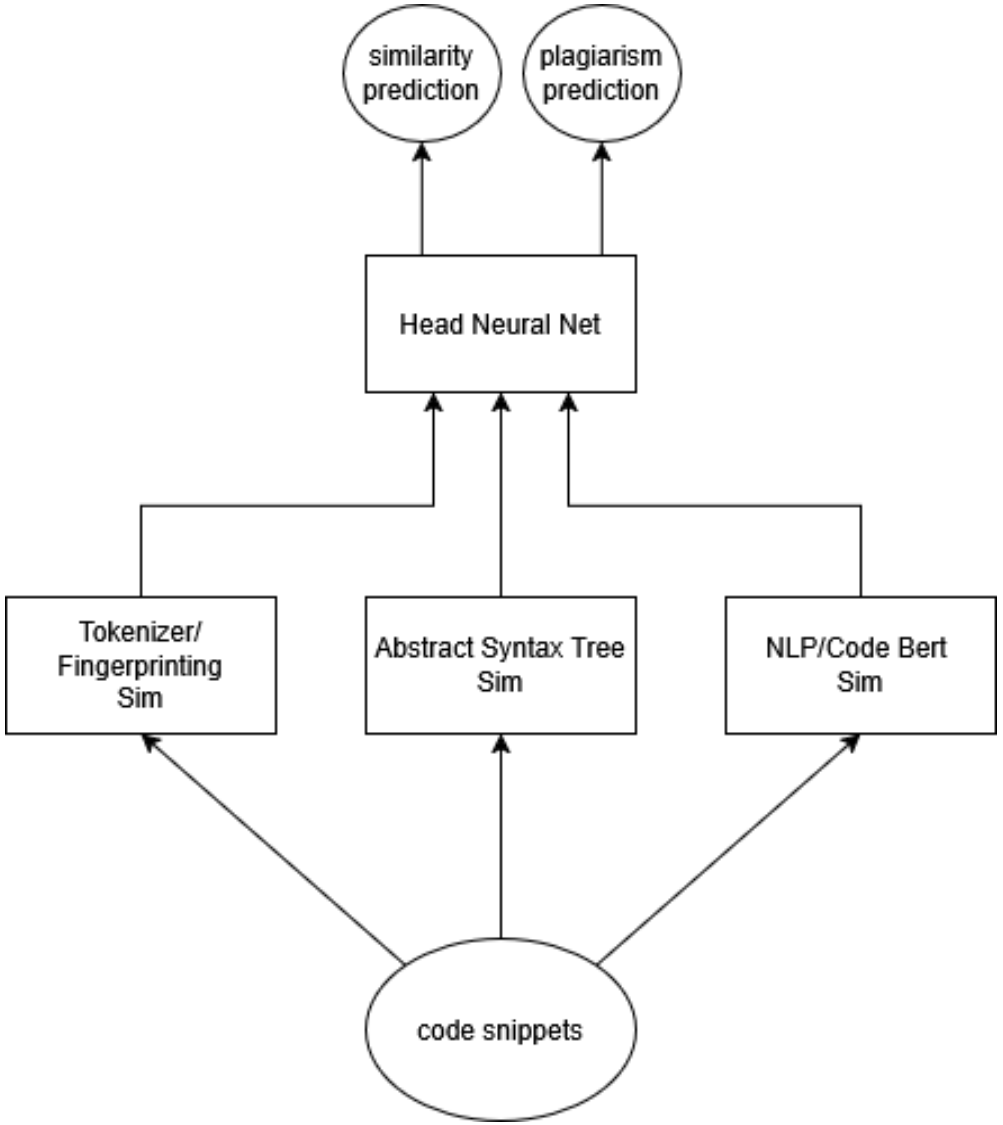
- SQS is a **message queuing service**
- Allows **decoupled applications to communicate** with each other without interfering with each other
    - No more "Oh no, I broke the endpoint"

Why SQS?

- **NO** cost – AWS allows you to send 1,000,000 requests in SQS every month, free of charge
    - Therefore, it is essentially **FREE** (unless you check for plagiarism more than 1,000,000 times a month)
- **Very Reliable** – Provides "at least once" or "exactly once" delivery, ensuring messages are always received
- **No Management Required** – SQS is fully managed by AWS. No need to worry about set up, or scalability. It is all done by AWS



Model Design (Diagram)



Model Design (Diagram)

Key Components

- Tokenization Sim Module (MOSS)
- AST Sim Module
- NLP Module

Why include all?

- Best of all worlds. Majority of the lifting is done by the NLP module, but additional pointers/clues that the NLP module may not catch can be provided by other similarity methods without significant computational overhead.

Results

- Outperforms standard algorithm on cases where obvious signs of plagiarism are present.

What are the signs?

- Token occurrence and frequency
- Structure/ordering
- Logical equivalences

Example

plag2.py VIEW SUBMISSION >

```
1  import math
2  j = 1
3  for i in range (j):
4
5      count = int(input("Enter number of elements (0 to exit): "))
6      if count == 0:
7          exit()
8
9      values = list(map(int, input("Enter the elements separated by space: ").split()))
10     average = sum(values) / len(values)
11     variance_sum = sum((item - average) ** 2 for item in values)
12     std_deviation = math.sqrt(variance_sum / count)
13
14     print(f'{std_deviation:.8f}')
15     j+=1
16
```

Plagiarized file

Original file

s129271906.py VIEW SUBMISSION >

```
1  from math import sqrt
2  while 1:
3      n = int(input())
4      if n == 0:
5          break
6      a = list(map(int, input().split()))
7      av = sum(a)/len(a)
8      Sum = sum((x-av)**2 for x in a)
9      print(f'{sqrt(Sum/n):.8f}')
10
```

What does MOSS say?

30% similarity with a batch mean of 64%

Above threshold of plagiarism provided by 7%

Overall, weak indication that plagiarism occurred

What does our model say?

Similarity of 68% with batch mean of 55%

Above threshold of plagiarism provided by 30%

Stronger indication that plagiarism has occurred

Thanks for listening!

Q&A

Another example from dataset

```python
import math
j = 1
for i in range (j):

    count = int(input("Enter number of elements (0 to exit): "))
    if count == 0:
        exit()

    values = list(map(int, input("Enter the elements separated by space: ").split()))
    average = sum(values) / len(values)
    variance_sum = sum((item - average) ** 2 for item in values)
    std_deviation = math.sqrt(variance_sum / count)

    print(f'{std_deviation:.8f}')
    j+=1
```

```python
import math
while 1:
    n = int(input())
    if n == 0:
        break
    s = list(map(int,input().split()))
    m = sum(s) / len(s)
    x = 0
    for i in range(n):
        x += (s[i] -m )**2 / n
    a = math.sqrt(x)
    print(a)
```