Detect Code Similarity

Chen Lin

.

ABSTRACT

This project aims to detect similarity between two python codes. First, we remove all the annotation from the text, thus get the meaningful parts of the code. Then, we process the codes and use Levenshtein package and Universal Sentence Encoder to get the similarity values.

1 Introduction

It's always important to detect code similarity to avoid plagiarisim, thus we develop the experiment to detect the similarity between python codes.

2 Design

We use two methods: Levenshtein Similarity and Universal Sentence Encoder (USE) Similarity to detect plagiarisim between two python codes.

Levenshtein Similarity

We use the function Levenshtein.ratio from the Levenshtein package to calculate Levenshtein Similarity. As manual of Levenshtein package, please refer to Antti Haapala al. [1].

Table 1: Experiment Result - Similarity Scores.

ID	Levenshtein Similarity	USE Similarity
1	0.901	0.871
2	0.888	0.921
3	0.907	0.946
4	0.844	0.911
5	0.795	0.907
6	0.766	0.874
7	0.744	0.784

Universal Sentence Encoder Similarity

The Universal Sentence Encoder makes it easier to get sentence level embeddings. The sentence level embeddings could be used to compute sentence level similarity. We use the average of the maximum sentence similarity between lines of code files as the overall Universal Sentence Encoder similarity.

3 Results and Discussion

From table 1, we can see the similarity between my code and other seven students' codes. The overall result shows that my code is very different from student 7 but seems close to student 3.

4 Conclusion

The Levenshtein Similarity and USE Similarity could be combined to detect similarity between students' codes.

References

[1] A. Haapala, *The Levenshtein Python C extension module* (Pypi.org, 2014)