Cross-Language Source Code Plagiarism Detection using Explicit Semantic Analysis and Scored Greedy String Tilling

Tomáš Foltýnek[1], Richard Všianský[1], Norman Meuschke[2,3], Dita Dlabolová[1], Bela Gipp[2,3]
[1]Mendel University in Brno, Czech Republic, {tomas.foltynek | xvsiansk | dita.dlabolova}@mendelu.cz
[2]University of Wuppertal, Germany, {meuschke | gipp}@uni-wuppertal.de
[3]University of Konstanz, Germany

ABSTRACT

We present a method for source code plagiarism detection that is independent of the programming language. Our method EsaGst combines Explicit Semantic Analysis and Greedy String Tiling. Using 25 cases of source code plagiarism in C++, Java, JavaScript, PHP, and Python, we show that EsaGst outperforms a baseline method in identifying plagiarism across programming languages.

CCS CONCEPTS

• **Information systems~Information retrieval~Specialized information retrieval**

KEYWORDS

Source Code Plagiarism Detection, Explicit Semantic Analysis, Greedy String Tilling

ACM Reference format:

Tomáš Foltýnek, Richard Všianský, Norman Meuschke, Dita Dlabolová and Bela Gipp. 2020. Cross-Language Source Code Plagiarism Detection via Explicit Semantic Analysis and Scored Greedy String Tilling. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'20), August 1 - 5, Virtual event, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3383583.3398594

1 Introduction & Related Work

Source code plagiarism detection (SCPD) is an effective deterrent to undue reuse of code in programming assignments, which are common in computer science and related study programs.

Many SCPD methods focus on specific programming languages by employing approximate string matching to identify similar programs [1]. Other methods additionally analyze the structure or semantics of source code [2]. Some methods addressed the cross-language SCPD task using Latent Semantic Analysis [3].

JCDL'20, August 1–5, 2020, Virtual event, China.

ACM ISBN 978-1-4503-7585-6/20/06...$15.00.
DOI: https://doi.org/10.1145/3383583.3398594

We presume that plagiarists trying to obfuscate reused code preserve the semantics of the identifiers, comments, and other tokens. Thus, we see a semantic analysis as promising for devising a language-independent SCPD method. Therefore, we adapt Explicit Semantic Analysis (ESA) [4], a well-established semantic analysis method, and Greedy String Tiling to the SCPD use case. ESA models text as concept vectors. The concepts are the topics in a knowledge base, which is typically Wikipedia or another encyclopedia. The vector components reflect the relevance of the modeled text for each of the concepts. Greedy String Tiling (GST) is an algorithm with near-linear complexity to find all individually longest substring matches in two strings [5].

2 Method

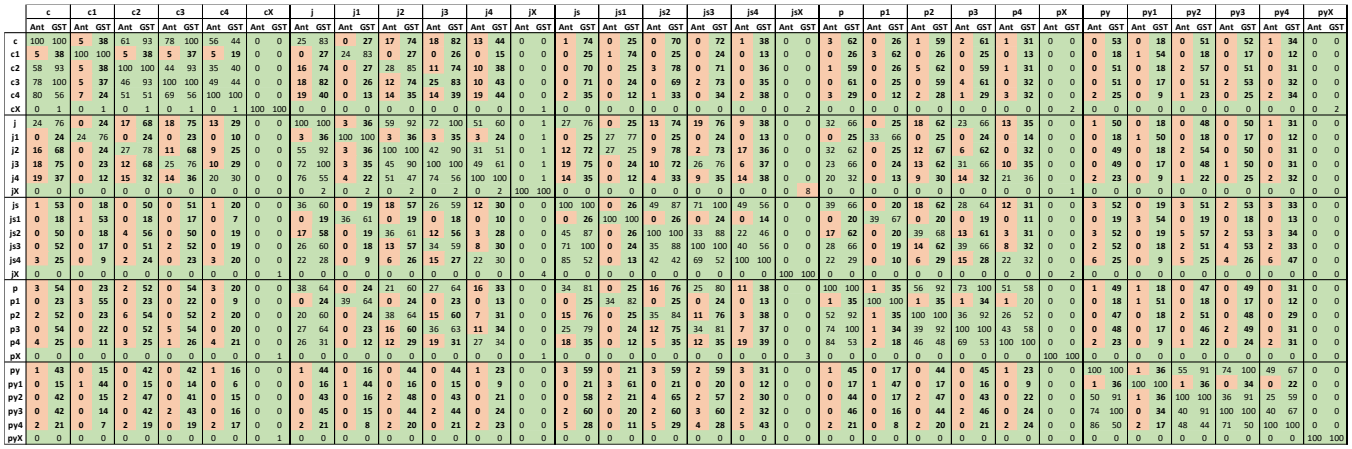
To perform ESA, we used the EsaPlag system [6] and thirty thousand articles from the categories "Computer programming" and "Fields of mathematics" in the English Wikipedia. Using the title of articles as concepts, we represented each document, i.e., a computer program, by deriving a concept vector for each term in the document. To maintain all semantic information of documents, we only removed line breaks before forming the vectors.

To compute the similarity of documents, we devised the Scored GST algorithm that determines the longest sequence of semantically similar terms. Other than GST, Scored GST matches not only identical elements but all elements whose similarity is above a threshold. Here, the concept vectors for document terms are the elements, whose similarity we computed via the cosine measure. We set the cosine similarity above which we consider concept vectors a match to 50% and the final score above which we report results to 5% as this value maximized the F1 score.

3 Experiments

To evaluate our SCPD method, we created a dataset of simulated source code plagiarism. We implemented a basic programming assignment – a calculator supporting basic arithmetic operations – in the five most common languages on GitHub, i.e., C++, Java, JavaScript, PHP, and Python. For each language-specific implementation, we created four plagiarized versions using the following obfuscation methods: (1) renaming identifiers, (2) renaming identifiers by converting camelCase to snake_case, (3) restructuring the code, and (4) reusing half of the code. To test for false positives, we used unrelated code with no semantic matches.

As a baseline, we used the text matching system Anton [5].



	c		c1		c2		c3		c4		cX		j		j1		j2		j3		j4		jX		js		js1		js2		js3		js4		jsX		p		p1		p2		p3		p4		pX		py		py1		py2		py3		py4		pyX	
	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST	Ant	GST
c	100	100	5	38	61	93	78	100	56	44	0	0	25	83	0	27	17	74	18	82	13	44	0	0	1	74	0	25	0	70	0	72	1	38	0	0	3	62	0	26	1	59	2	61	1	31	0	0	0	53	0	18	0	51	0	52	1	34	0	0
c1	5	38	100	100	5	38	5	37	5	19	0	0	0	27	24	83	0	27	0	26	0	15	0	0	0	25	1	74	0	25	0	24	0	13	0	0	0	26	3	62	0	26	0	25	0	13	0	0	0	18	1	54	0	18	0	17	0	12	0	0
c2	58	93	5	38	100	100	44	93	35	40	0	0	16	74	0	27	28	85	11	74	10	38	0	0	0	70	0	25	3	78	0	71	0	36	0	0	1	59	0	26	5	62	0	59	1	31	0	0	0	51	0	18	2	57	0	51	0	31	0	0
c3	78	100	5	37	46	93	100	100	49	44	0	0	18	82	0	26	12	74	25	83	10	43	0	0	0	71	0	24	0	69	2	73	0	35	0	0	0	61	0	25	0	59	4	61	0	32	0	0	0	51	0	17	0	51	2	53	0	32	0	0
c4	80	56	7	24	51	51	69	56	100	100	0	0	19	40	0	13	14	35	14	39	19	44	0	0	2	35	0	12	1	33	0	34	2	38	0	0	3	29	0	12	2	28	1	29	3	32	0	0	2	25	0	9	1	23	0	25	2	34	0	0
cX	0	1	0	1	0	1	0	1	0	1	100	100	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	2
j	24	76	0	24	17	68	18	75	13	29	0	0	100	100	3	36	59	92	72	100	51	60	0	1	27	76	0	25	13	74	19	76	9	38	0	0	32	66	0	25	18	62	73	66	13	35	0	0	1	50	0	18	0	48	0	50	1	31	0	0
j1	0	24	24	76	0	24	0	23	0	10	0	0	3	36	100	100	3	36	3	35	3	24	0	1	0	25	27	77	0	25	0	24	0	13	0	0	0	25	33	66	0	25	0	24	0	14	0	0	0	18	1	50	0	18	0	17	0	12	0	0
j2	16	68	0	24	27	78	11	68	9	25	0	0	55	92	3	36	100	100	42	90	31	51	0	1	12	72	27	25	9	78	2	73	17	36	0	0	32	62	0	25	12	67	6	62	0	32	0	0	0	49	0	18	2	54	0	50	0	31	0	0
j3	18	75	0	23	12	68	25	76	10	29	0	0	72	100	3	35	45	90	100	100	49	61	0	1	19	75	0	24	10	72	26	76	6	37	0	0	23	66	0	24	13	62	31	66	10	35	0	0	0	49	0	17	0	48	1	50	0	31	0	0
j4	19	37	0	12	15	32	14	36	20	30	0	0	76	55	4	22	51	47	74	56	100	100	0	1	14	35	0	12	4	33	9	35	14	38	0	0	20	32	0	13	9	30	14	32	21	36	0	0	2	23	0	9	1	22	0	25	2	32	0	0
jX	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	2	0	2	0	2	100	100	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
js	1	53	0	18	0	50	0	51	1	20	0	0	36	60	0	19	18	57	26	59	12	30	0	0	100	100	0	26	49	87	71	100	49	56	0	0	39	66	0	20	18	62	28	64	12	31	0	0	1	50	0	19	3	51	2	53	3	33	0	0
js1	0	18	1	53	0	18	0	17	0	7	0	0	0	19	36	61	0	19	0	18	0	10	0	0	0	26	100	100	0	26	0	24	0	14	0	0	0	20	39	67	0	20	0	19	0	11	0	0	0	19	3	54	0	19	0	18	0	13	0	0
js2	0	50	0	18	4	56	0	50	0	19	0	0	17	58	0	19	36	61	12	56	3	28	0	0	45	87	0	26	100	100	33	88	22	46	0	0	17	62	0	20	39	68	13	61	3	31	0	0	3	52	0	19	5	57	2	53	3	34	0	0
js3	0	52	0	17	0	51	2	52	0	19	0	0	26	60	0	18	13	57	34	59	8	30	0	0	71	100	0	24	35	88	100	100	40	56	0	0	28	66	0	19	14	62	39	66	8	32	0	0	2	52	0	18	2	51	4	53	2	33	0	0
js4	3	25	0	9	2	24	0	23	3	20	0	0	22	28	0	9	6	26	15	27	22	30	0	0	85	52	0	13	42	42	69	52	100	100	0	0	22	29	0	10	6	29	15	28	22	32	0	0	6	25	0	9	5	25	4	26	6	47	0	0
jX	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
p	3	54	0	23	2	52	0	54	3	20	0	0	38	64	0	24	21	60	27	64	16	33	0	0	34	81	0	25	16	76	25	80	11	38	0	0	100	100	1	35	56	92	73	100	51	58	0	0	1	49	1	18	0	47	0	49	0	31	0	0
p1	0	23	3	55	0	23	0	22	0	9	0	0	0	24	39	64	0	24	0	23	0	13	0	0	0	25	34	82	0	25	0	24	0	13	0	0	1	35	100	100	1	35	1	34	1	20	0	0	0	18	1	51	0	18	0	17	0	12	0	0
p2	2	52	0	23	6	54	0	52	2	20	0	0	20	60	0	24	38	64	15	60	7	31	0	0	15	76	0	25	35	84	11	76	3	38	0	0	52	92	1	35	100	100	36	92	26	52	0	0	0	47	0	18	2	51	0	48	0	29	0	0
p3	0	54	0	22	0	52	5	54	0	20	0	0	27	64	0	23	16	60	36	63	11	34	0	0	25	79	0	24	12	75	34	81	7	37	0	0	74	100	1	34	39	92	100	100	43	58	0	0	0	48	0	17	0	46	2	49	0	31	0	0
p4	4	25	0	11	3	25	1	26	4	21	0	0	26	31	0	12	12	29	19	31	27	34	0	0	18	35	0	12	5	35	12	35	19	39	0	0	84	53	2	18	46	48	69	53	100	100	0	0	2	23	0	9	1	22	0	24	2	31	0	0
pX	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0
py	1	43	0	15	0	42	0	42	1	16	0	0	1	44	0	16	0	44	0	44	1	23	0	0	3	59	0	21	3	59	2	59	3	31	0	0	1	45	0	17	0	44	0	45	1	23	0	0	100	100	1	36	55	91	74	100	49	67	0	0
py1	0	15	1	44	0	15	0	14	0	6	0	0	0	16	1	44	0	16	0	15	0	9	0	0	0	21	3	61	0	21	0	20	0	12	0	0	0	17	1	47	0	17	0	16	0	9	0	0	1	36	100	100	1	36	0	34	0	22	0	0
py2	0	42	0	15	2	47	0	41	0	15	0	0	0	43	0	16	2	48	0	43	0	21	0	0	0	58	2	21	4	65	2	57	2	30	0	0	0	44	0	17	2	47	0	43	0	22	0	0	50	91	1	36	100	100	36	91	25	59	0	0
py3	0	42	0	14	0	42	2	43	0	16	0	0	0	45	0	15	0	44	2	44	0	24	0	0	2	60	0	20	2	60	3	60	2	32	0	0	0	46	0	16	0	44	2	46	0	24	0	0	74	100	0	34	40	91	100	100	40	67	0	0
py4	2	21	0	7	2	19	0	19	2	17	0	0	2	21	0	8	2	20	0	21	2	23	0	0	5	28	0	11	5	29	4	28	5	43	0	0	2	21	0	8	2	20	0	21	2	24	0	0	86	50	2	17	48	44	71	50	100	100	0	0
pyX	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	100

Figure 1: Similarity scores of the baseline method (Ant) and EsaGst (GST) for programming languages (c = C++, j = Java, js = Javascript, p = PHP, py = Python), obfuscation methods (1-4), and unrelated documents (X), cf. Section 3.

Anton computes the similarity of documents as the ratio of hashed 5-word chunks occurring in both documents to the chunks in the first document. Since the score is asymmetric, so are the baseline results in Figure 1. We used Anton's default threshold for reporting results, i.e., a 20% overlap in chunks.

4 Results and Discussion

Figure 1 shows the results. Cells colored in green indicate correct detections, i.e., true positive and true negatives. Red fill denotes false positives and false negatives. Boldface font emphasizes cases, in which EsaGst identified a case that the baseline missed.

The baseline identified many monolingual cases and some cross-lingual cases involving syntactically similar languages like Java and JavaScript and, to a lesser extent, Java and C++.

For syntactically different languages, EsaGSt achieved much better results. Particularly, EsaGst yielded high similarity scores for the documents with a semantic similarity of 100% (x0, x2, x3). For monolingual plagiarism cases, the average EsaGst score is 99.2%, and for cross-lingual cases 89.0%. These results show that EsaGst is more robust to syntactic variations of the programming languages than the baseline method. Nonetheless, the similarity scores for cross-lingual cases involving Python are often low due to syntactic peculiarities of Python. For example, Python's keyword elif is expressed by two words (else if) in all other languages. Moreover, Python is an untyped language, which leads to shorter word sequences omitting words corresponding to the data types. This is why Python yielded the highest similarity with Javascript, which is an untyped language as well.

In our experiments, EsaGst could distinguish plagiarized documents from unrelated documents in all cases using a reporting threshold of 5%. However, a limitation of our small-scale test set is that it does not include non-plagiarized documents on related programming tasks, which would likely yield higher similarity scores and potentially false positive detections.

5 Conclusion

This poster presents preliminary results on using ESA and Scored GST to identify cross-language source code plagiarism in a small set of test programs. Important questions we plan to investigate in our future research include: (1) the effect of varying parameters like the dimensionality of concept vectors, the similarity thresholds, and the minimum substring lengths for GST, (2) the influence of the ESA knowledgebase dataset, and (3) the results of EsaGst and more cross-language SCPD methods on a larger, more diverse test set. The test set should include programs that different developers implemented for the same task to distinguish plagiarized from topically related content.

Despite the limitations of our initial experiments, we see Explicit Semantic Analysis combined with Scored Greedy String Tiling as a promising method for revealing semantically equivalent source code in different programming languages.

REFERENCES

[1] M. Ďuračík, E. Kršák, and P. Hrkút, "Current Trends in Source Code Analysis, Plagiarism Detection and Issues of Analysis Big Datasets," *Procedia Eng.*, vol. 192, pp. 136–141, 2017.

[2] A. Caliskan *et al.*, "When coding style survives compilation: De-anonymizing programmers from executable binaries," *arXiv Prepr. arXiv1512.08546*, 2015.

[3] G. Cosma and M. Joy, "An approach to source-code plagiarism detection and investigation using latent semantic analysis," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 379–394, 2012.

[4] E. Gabrilovich and S. Markovitch, "Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis," in *IJcAI*, 2007, vol. 7, pp. 1606–1611.

[5] R. Všianský, D. Dlabolová, and T. Foltýnek, "Source code plagiarism detection for PHP language," *Eur. J. Bus. Sci. Technol.*, vol. 3, no. 2, 2017.

[6] N. Meuschke, N. Siebeck, M. Schubotz, and B. Gipp, "Analyzing Semantic Concept Patterns to Detect Academic Plagiarism," in *Proceedings of the 6th International Workshop on Mining Scientific Publications*, 2017, pp. 46–53.

Citation for this Paper

Foltynek, T. & Vsiansky, R. & Meuschke, N. & Dlabolova, D. & Gipp, B., "Cross-Language Source Code Plagiarism Detection using Explicit Semantic Analysis and Scored Greedy String Tilling," in Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL), Aug. 2020, DOI: 10.1145/3383583.3398594.

BibTeX:

```
@inproceedings{FoltynekVMD20,
title = {Cross-{Language} {Source} {Code} {Plagiarism} {Detection} using
{Explicit} {Semantic} {Analysis} and {Scored} {Greedy} {String} {Tilling}},
doi = {10.1145/3383583.3398594},
booktitle = {Proceedings of the {ACM}/{IEEE} {Joint} {Conference} on {Digital}
{Libraries} ({JCDL})},
author = {Foltynek, Tomas and Vsiansky, Richard and Meuschke, Norman and
Dlabolova, Dita and Gipp, Bela},
month = aug,
year = {2020}
}
```

RIS:

```
TY  - CONF
TI  - Cross-Language Source Code Plagiarism Detection using Explicit Semantic
Analysis and Scored Greedy String Tilling
AU  - Foltynek, Tomas
AU  - Vsiansky, Richard
AU  - Meuschke, Norman
AU  - Dlabolova, Dita
AU  - Gipp, Bela
C3  - Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)
DA  - 2020/08//
PY  - 2020
DO  - 10.1145/3383583.3398594
ER  -
```