Jurnal Ilmu Komputer Agri-Informatika

Tersedia secara *online* di:
http://journal.ipb.ac.id/index.

Volume 7 Nomor 2 halaman 74 – 83
e ISSN: 2654-9735, p ISSN: 2089-6026

Pendeteksian Penjiplakan Kode Program C dengan Menggunakan Algoritme *K-medoids*

C Source Code Plagiarism Detection using K-medoids Algorithm

RADEN FITYAN HAKIM[1]*, IMAS SUKAESIH SITANGGANG[1]

Abstrak

Pada era globalisasi ini, aksi penjiplakan lebih sering dan lebih mudah dilakukan, termasuk penjiplakan terhadap kode program. Pendeteksian manual terhadap aksi penjiplakan memakan banyak waktu maupun tenaga. Oleh karena itu, dibutuhkan sistem yang dapat membantu proses pendeteksian. Pendeteksian dapat dilakukan dengan mengelompokkan kode-kode program yang mirip berdasarkan struktur kode program. Penelitian ini dilakukan untuk menerapkan algoritme *K-medoids* pada 4 buah *dataset* kode program C dan menganalisis hasil pengelompokan yang diperoleh. Hasil percobaan menunjukkan bahwa *clustering* terbaik pada dataset 1 (Kondisi *If-Else* dan Pengulangan *While*) diperoleh pada *k* (jumlah *cluster*) 10 dengan rataan *dissimilarity* 2.655 dengan18.9% tugas mahasiswa memiliki kelompok yang sama. Pada dataset 2 (Pengulangan *While*), *clustering* terbaik diperoleh pada k = 9 dengan rataan *dissimilarity* 2.227 dengan 32.6% tugas mahasiswa memiliki kelompok yang sama. Untuk *dataset* 3 (Pengulangan For Bersarang dan Kondisi If), tugas mahasiswa terbagi menjadi dua buah *cluster* dengan rataan *dissimilarity* 0.719, dengan 87% tugas mahasiswa berada pada *cluster* yang sama. Hasil *clustering* terbaik pada *dataset* 4 (Kondisi *If-Else* dan Pengulangan *For*) diperoleh pada *k* = 6 dengan rataan *dissimilarity* 3.199, dengan 61% tugas mahasiswa berada pada kelompok yang sama.

Kata Kunci**:** *clustering, K-medoids, N-gram,* pendeteksian penjiplakan, program C.

Abstract

In the era of globalization, practice of plagiarism is more common and easier to do, including plagiarism on program codes. Manual detection of plagiarism takes a lot of time and effort. Therefore, we need a system that can help detection process. The detection can be done by grouping program codes that have similar structure. This study aims to apply K-medoids algorithm on 4 pieces C code program dataset and analyze the clustering result. The experimental results show that the best clustering in dataset 1 (If-Else Condition and Looping While) was obtained at k (number of cluster) 10 with average of dissimilarity 2.655, where 18,9% of students have a same group. In dataset 2 (Looping While), the best clustering obtained at k = 9 with average of dissimilarity 2.227 where 32.6% student assignments are in the same group. For dataset 3, the assignments are divided into two clusters with the average of dissimilarty 0.719, where 87% of student assignments are in the same cluster. The best clustering result on the dataset 4 obtained at k = 6 with average of dissimalrity 3.199, where 61% of students assignments are in the same group.

*Keywords**:** clustering, K-Medoids, N-gram, plagiarism detection.*

PENDAHULUAN

Mata kuliah Algoritme dan Pemrograman merupakan salah satu mata kuliah yang memberikan penugasan berupa kode program yang umumnya dikumpulkan dalam bentuk berkas digital. Hal ini menyebabkan tugas kode program rentan terhadap tindak penjiplakan. Saat ini pendeteksian penjiplakan semakin sulit dilakukan. Hal ini disebabkan bertambahnya jumlah mahasiswa dan semakin mudahnya mahasiswa untuk saling berbagi hasil pekerjaan baik melalui internet ataupun media lainnya. Selain itu, pendeteksian semakin sulit dengan adanya modifikasi berupa penambahan atau penghapusan baris komentar, merubah nama variabel atau

[1]Departemen Ilmu Komputer, FMIPA, Institut Pertanian Bogor
*Penulis Korespondensi: Surel: beatwork01@gmail.com

fungsi, maupun merubah susunan struktur kode programnya. Pendeteksian secara manual sangatlah memakan banyak biaya dan juga tenaga sehingga dibutuhkan suatu cara ataupun sistem yang dapat membantu pendeteksian penjiplakan secara otomatis.

Gumilang (2013) menggunakan *structure-oriented code-based system* untuk pendeteksian penjiplakan kode program C dengan *clustering*. Proses pengelompokan pada penelitian Gumilang dilakukan dengan menggunakan algoritme *K-means* dengan iterasi *K-means* secara otomatis. Iterasi *K-means* otomatis yaitu melakukan *clustering* menggunakan *K-means* dengan nilai *K* bertambah secara otomatis pada setiap iterasinya. Iterasi berhenti apabila anggota-anggota *clusters* hasil sudah cukup dekat dengan *centroid*-nya. Notyasa (2013) juga menggunakan *structure-oriented code-based system* untuk pendeteksian penjiplakan kode program C dengan *clustering.* Proses pengelompokan pada penelitian ini dengan menggunakan salah satu metode *clustering* HDC yaitu *bisecting K-means* yang menggunakan pendekatan *top-down,* dimana seluruh dokumen dimasukkan ke dalam satu *cluster* awal kemudian dipisahkan berdasarkan kemiripannya dengan menggunakan *K-means* dengan jumlah iterasi sebanyak 5 kali. Salah satu teknik pengelompokan *flat clustering* adalah *K-medoids*. Algoritme *K-medoids* merupakan penyempurnaan dari algoritme *K-means* yang sangat sensitif terhadap nilai *outlier* atau titik terjauh. *K-medoids* mengambil sebuah objek aktual untuk merepresentasikan sebuah *cluster,* dengan menggunakan sebuah objek untuk setiap *cluster*nya (Han dan Kamber 2011).

Penelitian ini dilakukan untuk menerapkan algoritme *K-medoids* untuk membantu pendeteksian penjiplakan pada kode program dan mengevaluasi dan menganalisis hasil dari pengelompokan kode program. Manfaat dari penelitian ini adalah untuk membantu mempermudah pendeteksian penjiplakan kode program C sehingga efisiensi kerja tenaga pendidik dapat meningkat dan tindak penjiplakan yang dilakukan mahasiswa dapat berkurang.

METODE

Data Penelitian

Data yang digunakan pada penilitan ini merupakan kode program yang diambil dari Bagian Komputasi Terapan Departemen Ilmu Komputer. Data penelitian ini adalah kumpulan berkas digital kode program bahasa C yang merupakan tugas Mata Kuliah Algoritme dan Pemrograman yang diberikan kepada mahasiswa Program Sarjana Ilmu Komputer. Kode program bahasa C tersebut dibagi ke empat kelompok berdasarkan jenis tugas, yaitu DataSet1 untuk jenis tugas pjj0210, DataSet2 untuk pjj0302, DataSet3 untuk pjj0307 dan DataSet4 untuk jenis tugas pjj0309 dengan total data yang digunakan sebanyak 307 buah kode program tugas pemrograman bahasa C. DataSet1 merupakan sekumpulan tugas dengan jenis penugasan tentang Kondisi *If-Else* dan pengulangan *While*, DataSet2 merupakan jenis tugas mengenai Pengulangan *While*, DataSet3 adalah jenis tugas mengenai Pengulangan *For* Bersarang dan Kondisi *If*, dan DataSet4 merupakan jenis tugas mengenai Kondisi *If-Else* dan Pengulangan *For*. Keempat buah jenis tugas yang dipakai untuk *dataset* pada penelitian ini memiliki tingkat kesulitan menengah.

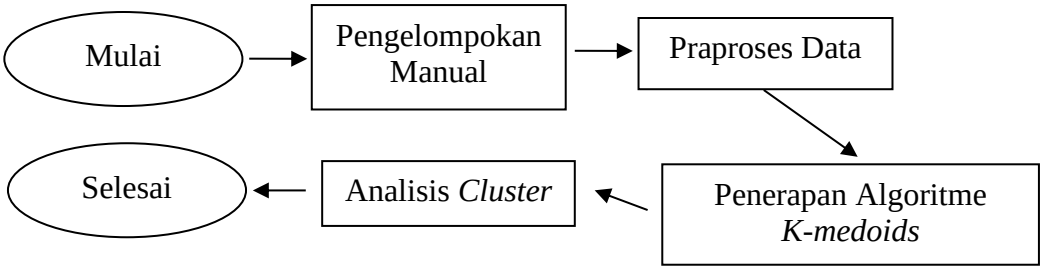
Tahapan Penelitian

Tahapan penelitian yang dilakukan dalam penerapan algoritme *K-medoids* untuk menentukan pengelompokan kode program yang mirip berdasarkan struktur kodenya dapat dilihat pada Gambar 1.

Pengelompokan Manual

Pada tahap ini dilakukan proses pengelompokan data secara manual. Setiap sampel diperiksa satu persatu dan melalui 2 tahap pemeriksaan, yaitu tahap pemeriksaaan berdasarkan nilai *output* dari kode program dan tahap pemeriksaaan berdasarkan struktur kode programnya.

Setiap kode program yang memiliki *output* dan struktur kode program yang secara signifikan sama dikelompokkan kedalam *cluster* yang sama.



Gambar 1 Tahapan penelitian.

Praproses Data

Praproses data dilakukan untuk mengubah data mentah, yaitu kode-kode program C, ke dalam format yang dapat diproses dalam *clustering*, yaitu tabel *term frequency*. Tabel *term frequency* diperoleh setelah melalui beberapa tahap pemrosesan yang dilakukan kepada setiap kode program yaitu dimulai dengan pembuangan *preprocessor directives*, tokenisasi, penyederhanaan, dan *N-gram*. Pada penelitian ini, proses *N-gram* dilakukan pada *token stream* dengan ukuran N=4 karena merupakan nilai *N* terbaik untuk pendeteksian penjiplakan kode program (Chawla 2003 dalam Burrows 2004). *Dataset* yang berupa sekumpulan *term N-gram* kemudian diproses dengan menggunakan perangkat lunak *data mining* WEKA untuk mendapatkan tabel *term frequency* untuk setiap data. Tabel *term frequency* merupakan sebuah tabel yang berisi frekuensi kemunculan sebuah *term* di setiap kode program.

Penerapan Algoritme K-medoids

Pada tahap ini akan diterapkan algoritme *K-medoids* terhadap setiap *dataset*. Tahapan ini dilakukan dengan mengimplementasikan algoritme *K-medoids* dengan menggunakan *tool R*. Pengolahan data dengan algoritme *K-medoids* ini mengelompokan data menjadi beberapa *cluster* yang setiap anggota dari *cluster* yang sama memiliki kemiripan satu sama lainnya.

Analisis Cluster

Pada tahap ini diperoleh hasil akhir berupa sejumlah *cluster* untuk setiap jenis *dataset* beserta daftar anggota dari masing-masing *cluster* dan juga nilai *dissimilarity* untuk setiap *cluster*nya. Analisis akan dilakukan untuk dapat mengetahui mahasiswa yang cenderung saling melakukan penjiplakan dan jenis tugas yang lebih banyak terjadinya penjiplakan.

Algoritme *K-medoids*

Strategi yang mendasar dari algoritme *K-medoids* adalah untuk menemukan sejumlah k *cluster* dari n buah objek dengan terlebih dahulu menentukan objek perwakilan (*medoids*) untuk setiap *cluster*. Setiap objek yang tersisa kemudian dikelompokkan dengan *medoids* yang paling mirip. Algoritme *K-medoids* menggunakan objek representatif sebagai titik acuan, bukan menggunakan nilai rata-rata dari setiap objek dalam sebuah *cluster*. Algoritme ini mengambil *input* parameter k, yaitu jumlah *cluster* yang akan dipartisi antara himpunan *n* objek. Prinsip dari algoritme ini adalah untuk meminimalkan nilai *dissimilarity* antara setiap objek dan titik representatifnya.

Algoritme ini secara acak memilih *k* objek dalam *dataset* sebagai objek representatif awal yang disebut sebagai *medoids*. Sebuah *medoid* dapat didefinisikan sebagai objek didalam *cluster* yang memiliki nilai rata-rata *dissimilarity* antar objek dalam *cluster* paling minimal. Algoritme ini menggunakan pemodelan sebagai berikut.

$$F(x) = minimize \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) z_{ij}$$

F(x) merupakan fungsi utama untuk meminimalkan *d(i,j)* yang merupakan ukuran ketidak miripan antar entitas *i* dan *j*, dan z_{ij} adalah variabel yang memastikan hanya ketidakmiripan antara entitas dari *cluster* yang sama yang dihitung. Algoritme *K-medoids* ditunjukkan oleh Gambar 2.

```
Algorithm : k-medoids, PAM, a k-medoids algorithm for partioningbased on medoid
or central object
Input :
   ▪ k: the number of cluster,
   ▪ D: a data set containingn objects.
Output : A set of k cluster.
Method :
   1) arbitrary choose k object in D as the initial representative object or
      seeds;
   2) repeat
   3)       Assign each remaining object to the cluster with the nearest
      representative object;
   4)       Randomly select a nonrepresentative object, o_random;
   5)       Compute the total cost, S,of swapping representative object, o_j,
      with o_random;
   6)       If S < 0 then swap o_j with o_random to form the new set of k
      representative object;
   7) Until no change
```

Gambar 2 Algoritme *K-medoids* (Han dan Kamber 2006).

HASIL DAN PEMBAHASAN

Pengelompokkan Kode Program secara Manual

Data yang digunakan pada penelitian ini adalah kode program berbahasa C. Jumlah data sebanyak 307 buah kode program yang dipilih dari 4 buah tugas dengan kesulitan menengah. Masing-masing tugas tugas kemudian dijadikan *dataset* dalam penelitian ini, seperti yang dapat dilihat pada Tabel 1.

Tabel 1 *Dataset* awal

Dataset	Jumlah dokumen
DataSet1 (pjj0210)	95
DataSet2 (pjj0302)	92
DataSet3 (pjj0307)	60
DataSet4 (pjj0309)	60
DataSetAll	307

Pengelompokan manual dilakukan untuk keempat *dataset* dengan memeriksa struktur kode program dan juga *output*nya. Pada pengelompokan secara manual ini terdapat *cluster* yang hanya berisi 1 atau 2 kode program. *Cluster* yang berisi terlalu sedikit kode program dihilangkan. Hasil pengelompokan manual dari keempat *dataset* dapat dilihat pada Tabel 2.

Tabel 2 *Dataset* setelah pengelompokan manual

Dataset	Jumlah dokumen	*Cluster* manual
DataSet1 (pjj0210)	95	9
DataSet2 (pjj0302)	92	9
DataSet3 (pjj0307)	60	2
DataSet4 (pjj0309)	60	6
DataSetAll	307	26

Praproses Data

Pada tahap ini proses pertama yang dilakukan adalah mendapatkan *dataset* yang berupa sekelompok *term N-gram* dengan nilai N=4. *Dataset* ini didapatkan melalui beberapa tahap yaitu pembuangan *preprocessor directives*, tokenisasi, penyederhanaan, dan kemudian *N-gram*. Pembuangan *processor directives* adalah pembuangan sejumlah baris kode program yang diawali dengan karakter ”#”, karena pada penelitian ini diasumsikan tidak adanya makro pada kode program. Setelah itu dilanjutkan dengan tahap tokenisasi, dimana pada tahap ini dilakukan pemilihan *keyword* dan *special character* dari masing-masing kode program. *Keyword* dan *special character* didapatkan setelah memilih tiap *term* dengan membuang *whitespace*, baris komentar, dan karakter “;”, karena merupakan karakter umum yang digunakan di setiap akhir baris kode program C. Kemudian dilanjutkan dengan tahap penyederhanaan yaitu mengubah *keyword* dan *special character* yang didapat dari proses tokenisasi menjadi deretan token sederhana yang diurutkan berdasarkan struktur kode programnya. Contoh penyederhanaan kode program menjadi sebuah *token stream* dapat dilihat pada Gambar 3.

Kode program:

```
1  #include <stdio.h>
2  int main (){
3    int var;
4    for (var=0; var<5; var++){
5      printf("%d\n", var);
6    }
7    return 0;
8  }
```

Token stream:
ANhikANMhNoNNmNNaaikNh5jNilRNl

Gambar 3 Contoh penyederhanaan kode program.

Dalam proses pembuatan deretan token yang diurutkan berdasarkan struktur kode programnya, dilakukan pengecekan dan penambahan spasi di awal dan di akhir *keyword* atau *special character* tersebut, untuk memastikan tidak terjadinya kesalahan konversi yang diakibatkan adanya *keyword* dan *special character* yang berhimpitan satu sama lainnya. Perubahan/konversi *keyword* dan *special character* dilakukan dengan mengikuti aturan konversi yang dapat dilihat pada Tabel 3. Hasil dari proses penyederhanaan ini berupa *string* panjang token sederhana yang sering disebut dengan *token stream*.

Tabel 3 Tabel aturan konversi kode program menjadi token sederhana

Keyword/special character	Token sederhana	*Keyword/special character*	Token sederhana	*Keyword/special character*	Token sederhana
+	a	.	p	for	M
-	b	!	q	ALPHANUM	N
*	c	:	r	goto	O
/	d	Int	A	case	P
%	e	Float	B	break	Q
&	f	Char	C	return	R
|	g	Double	D	switch	S
(	h	short	E	const	T
)	i	long	F	continue	U
,	j	signed	G	sizeof	W
{	k	unsigned	H	struct	X
}	l	if	I	enum	Y
<	m	else	J	typedef	Z
>	n	while	K	"STRING"	5
=	o	do	L		

Setelah *token stream* didapatkan kemudian dilanjutkan dengan proses *N-gram*. Proses *N-gram* dilakukan untuk membangkitkan *term* dengan cara memotong atau mengambil sebuah *string* panjang sebanyak N buah huruf. Pada penelitian ini digunakan nilai *N*=4. Gambar 4 merupakan sebuah contoh *N-gram* dari *token stream* pada Gambar 3.

_Nhi	NjNo	oNNh	iKhh	Naai	iNao	jNiR
Nhik	jNoN	NNh5	KhhN	aaii	NaoN	NiRN
hikH	NoNF	Nh5j	hhNh	aiim	aoNN	iRNl
ikHF	oNFA	h5jf	hNhN	iimo	oNNh	RNl_
kHFA	NFAN	5jfN	NhNj	imoN	NNh5	
HFAN	FANo	jfNi	hNjN	moNi	Nh5j	
FANj	AnoN	fNiK	NjNa	oNiN	h5jN	
ANjN	NoNN	NiKh	jNaa	NiNa	5jNi	

Gambar 4 *N-gram* dengan *N*=4.

Selanjutnya setelah dilakukan *N-gram* dengan nilai *N*=4, *frequency* jumlah kemunculan setiap *N-gram* dari masing-masing kode program dihitung, yang kemudian akan diperoleh sebuah tabel *term frequency*. Proses penghitungan *frequency* kemunculan dari *N-gram* tersebut menggunakan aplikasi WEKA yaitu menggunakan fungsi *filter unsupervised attribute* dengan sub fungsi *string to word vector*. Tabel 4 merupakan sebuah contoh *term frequency* dari *N-gram* pada Gambar 4.

Tabel 4 *Term frequency*

Dok	*Term*										
	_Nhi	Nhik	hikH	ikHF	kHFA	HFAN	FANj	AnjN	NjNo	jNoN	....
dok1	1	1	1	1	1	1	1	1	1	1	....
dok2	1	1	1	0	0	2	1	1	2	1	....
dok3	1	1	1	1	1	3	2	1	2	2	....
⋮	....	....	....	....	....	....	....	....	....	....	....

Penerapan Algoritme *K-medoids*

Dari *term frequency* masing masing *dataset* yang didapatkan pada tahap praproses data, kemudian dilakukan proses *clustering*. Proses *clustering* pada penelitian ini menggunakan algoritme *K-medoids*. *K-medoids* dapat dikatakan merupakan penyempuranaan dari algoritme *K-means* yang sangat sensitif terhadap nilai pencilan.

Algoritme *K-medoids* diaplikasikan kepada masing-masing *dataset* dengan mengunakan sebuah aplikasi, yaitu aplikasi R. Tabel *term frequency* dalam bentuk format *comma separated values* (CSV) dipanggil dalam *tool* R yang kemudian diproses menggunakan fungsi PAM (fungsi *K-medoids* dalam R) dan dilakukan serangkaian percobaan *clustering* dengan jumlah *cluster* K dari 2 sampai dengan 10. Nilai rata-rata *dissimilarity* masing-masing *cluster* kemudian dihitung untuk setiap *dataset*. Hasil percobaan *clustering* dengan menggunakan algoritme K-medoids yang diterapkan dengan menggunakan aplikasi R untuk DataSet1, DataSet2, DataSet3 dan DataSet4 dapat dilihat pada Tabel 5, Tabel 6, Tabel 7 dan Tabel 8.

Kolom K merupakan inputan yang dimasukkan untuk menentukan jumlah *cluster* yang diinginkan, sedangkan *Max.dissimilarty* merupakan nilai ketidakmiripan antar *cluster*, *Avg.dissimilarty* menunjukan nilai ketidakmiripan antar *object* didalam *cluster* yang sama. Semakin kecil nilai *avg.dissimilarty* semakin baik *cluster* yang terbentuk. Kemudian kolom *diameter* menunjukan panjang *diameter* dari masing-masing *cluster*.

Tabel 5 Hasil percobaan *clustering* untuk DataSet1

K	Nilai Rata-Rata			
	Max.Diss	*Avg.Diss*	*Diameter*	*Separation*
2	11.560	6.447	14.433	6.000
3	11.234	4.877	13.095	6.516
4	10.928	3.935	12.335	7.067
5	10.711	3.629	11.984	7.109
6	9.642	3.640	10.932	7.472
7	9.645	2.916	10.553	7.332
8	8.988	3.163	9.763	6.816
9	8.479	2.797	9.255	7.218
10	**8.508**	**2.655**	**9.037**	**7.182**

Tabel 6 Hasil percobaan *clustering* untuk DataSet2

K	Nilai Rata-Rata			
	Max.Diss	*Avg.Diss*	*Diameter*	*Separation*
2	8.215	5.516	8.489	4.123
3	7.379	4.939	7.817	5.517
4	5.966	3.926	6.803	5.374
5	5.748	4.037	6.327	5.269
6	5.890	3.381	6.747	5.207
7	5.536	2.718	6.194	5.148
8	5.395	2.446	6.366	5.130
9	**5.338**	**2.227**	**6.170**	**4.960**
10	5.338	2.060	5.971	5.012

Tabel 7 Hasil percobaan *clustering* untuk DataSet3

K	Nilai Rata-Rata			
	Max.Diss	*Avg.Diss*	*Diameter*	*Separation*
2	**3.741**	**0.719**	**3.741**	**16.703**
3	0.000	0.000	0.000	8.062
4	0.000	0.000	0.000	10.222
5	0.000	0.000	0.000	10.222
6	0.000	0.000	0.000	3.741
7	0.000	0.000	0.000	3.741
8	0.000	0.000	0.000	3.741
9	0.000	0.000	0.000	3.741
10	0.000	0.000	0.000	3.741

Tabel 8 Hasil percobaan *clustering* untuk DataSet4

K	Nilai Rata-Rata			
	Max.Diss	*Avg.Diss*	*Diameter*	*Separation*
2	19.766	4.712	19.984	13.341
3	14.307	3.971	15.680	17.359
4	12.603	5.520	12.603	16.819
5	10.165	3.410	15.667	15.987
6	**8.744**	**3.199**	**8.744**	**15.837**
7	8.944	2.981	8.944	12.793
8	0.000	0.000	0.000	12.251
9	0.000	0.000	0.000	12.654
10	0.000	0.000	0.000	12.653

Analisis Hasil *Cluster*

Menurut hasil percobaan yang dilakukan pada tahap penerapan algoritme *K-medoids* dengan menggunakan program R dapat diketahui pada saat nilai *K* berapa peng*cluster*an terbaik terjadi. Pemilihan nilai K terbaik dalam percobaan ini dapat dilihat dari nilai rata-rata keseluruhan nilai *dissimilarty* antar objek maupun antar *cluster*-nya, dan jumlah anggota dalam sebuah *cluster* dimana jumlah anggota untuk sebuah *cluster* tidak kurang dari tiga anggota. Nilai K terbaik untuk masing masing *dataset* yang memiliki kemiripan dalam struktur kode

programnya untuk DataSet1, DataSet2, DataSet3, dan juga DataSet4 dapat dilihat berturut-turut pada Tabel 9, Tabel 10, Tabel 11 dan Tabel 12.

Tabel 9 Hasil *cluster* terbaik untuk DataSet1 ada pada *K*=10

Cluster	Size	Huruf Mutu Mata Kuliah Algoritme dan Pemrograman
1	6	(D), (E), (C), (C), (C), (D)
2	13	(B), (B), (C), (D), (B), (A), (B), (D), (B), (C), (D), (C), (C)
3	18	(C), (D), (A), (D), (B), (D), (C), (C), (C), (C), (C), (C), (C), (B), (C), (B), (B), (E)
4	8	(B), (E), (C), (C), (C), (D), (E), (B)
5	8	(C), (C), (B), (C), (C), (C), (B), (B)
6	18	(B), (-), (C), (C), (C), (A), (C), (A), (D), (B), (A), (C), (B), (B), (C), (B), (D), (C)
7	7	(A), (D), (C), (B), (B), (B), (C)
8	3	(C), (A), (C)
9	10	(C), (B), (C), (A), (C), (B), (B), (B), (B), (C)
10	4	(C), (B), (B), (B)

Pada DataSet1 yaitu dengan kode tugas pjj0210 *cluster* terbaik terjadi pada *K*=10 karena berdasarkan hasil percobaan untuk nila *K*=1 sampai dengan *K*=10 untuk DataSet1 yang dicantumkan pada Tabel 5, pada nilai *K*=10 memiliki rata-rata nilai *dissmilarity* yang paling kecil yaitu 2.65 dan rata-rata *max.dissimilarity* sebesar 8.508. Pada Tabel 9, terlihat jumlah anggota *cluster* dan nilai mutu Mata Kuliah Algoritme dan Pemrograman. Dilihat dari persebaran anggota *clusternya*, penyebaran nilai mutu pada jenis tugas ini di setiap *cluster* merata antara yang nilai mutunya baik maupun yang kurang baik.

Tabel 10 Hasil *cluster* terbaik untuk DataSet2 ada pada *K*=9

Cluster	Size	Huruf Mutu Mata Kuliah Algoritme dan Pemrograman
1	7	(D), (C), (C), (C), (C), (D) , (-)
2	14	(B), (D), (D), (B), (-), (B), (A), (C), (B), (C), (B), (B), (C), (C)
3	12	(C), (B), (C),(B) ,(B), (A), (C), (A), (C), (A), (B), (B)
4	5	(A), (C), (D), (B), (E)
5	30	(D), (A), (C), (D), (C), (C), (C), (B), (C), (C), (D), (D), (C), (B), (C), (C), (C), (B), (C), (C), (C), (B), (C), (C), (C), (D), (D), (E), (C), (B)
6	7	(C), (B), (B), (B), (B), (C), (B)
7	4	(C),(C),(B),(B)
8	4	(B),(E),(B),(C)
9	9	(A), (B), (D), (C), (C), (B), (B), (C), (E)

Pada Tabel 10 diperlihatkan jumlah anggota *cluster* untuk DataSet2 ada pada *K*=9. Pemilihan nilai *K* terbaik untuk DataSet2 dengan kode tugas pjj0302 adalah pada saat *K*=9 dimana nilai rata-rata *dissimilarity* anggota clsuternya paling kecil yaitu 2.227. Namun dari Tabel 6 terlihat bahwa nilai rata-rata *dissimilarity* untuk *K*=10 lebih kecil tetapi nilai tersebut tidak dipilih dikarenakan ada salah satu *cluster*-nya yang hanya beranggotakan 2 buah objek data, sehingga nilai *K*=9 yang dipilih sebagai *cluster* terbaik. Dari tabel hasil mutu terlihat persebaran huruf mutu untuk setiap *cluster* merata dari nilai yang baik dan juga yang kurang baik sehingga dapat di prediksi pola alur penjiplakan yang terjadi.

Tabel 11 Hasil *cluster* terbaik untuk DataSet3 ada pada *K*=2

Cluster	Size	Huruf Mutu Mata Kuliah Algoritme dan Pemrograman
1	52	(B), (D), (B), (B), (D), (B), (A), (C), (C), (C), (A), (D), (C), (C), (C), (B), (A), (C), (C), (D), (D), (C), (C), (C), (C), (C), (B), (C), (D), (B), (C), (C), (C), (C), (A), (C), (C), (C), (B), (B), (C), (D), (D), (E), (C), (C), (C), (B),(D), (-)
2	8	(C), (E), (B), (C), (C), (B), (C), (E)

Untuk DataSet3 dengan kode tugas pjj0307 *cluster* terbaik berada pada saat *K*=2. Hal it dikarenakan nilai rata-rata *dissimilarity* antar objeknya sudah kecil 0.719 dan ketika nilai *K* ditambah menjadi 3, 4, dan seterusnya ada anggota *cluster* yang sudah sangat mirip dengan nilai rata-rata *disimilarity* = 0.00 sehingga *cluster* tersebut dipecah Kembali. Akhirnya, nilai *K* terbaik berada pada saat *K*=2. Nilai K terbaik untuk DataSet4 dengan kode tugas pjj0309 adalah pada saat K=6 dengan nilai rata-rata *dissimilarty* 3.199 dan tidak memiliki *cluster* yang beranggotakan lebih kecil dari 2 objek.

Tabel 12 Hasil *cluster* terbaik untuk DataSet4 ada pada K=6

Cluster	Size	Huruf Mutu Mata Kuliah Algoritme dan Pemrograman
1	3	(C), (E), (C)
2	7	(D),(-), (A), (C), (C), (C), (C)
3	37	(B), (C), (D), (B), (A), (C), (C), (C), (B), (C), (C), (D), (D), (D), (C), (B), (C), (C), (C), (B), (C), (D), (B), (C), (C), (C), (C), (A), (C), (C), (B), (C), (D), (E), (C), (D), (-)
4	5	(A),(A),(D),(A),(B)
5	5	(C),(B),(C),(D),(B)
6	3	(A),(B),(C)

Evaluasi Hasil *Clustering*

Evaluasi hasil *clustering* dilakukan dengan cara membandingkan hasil *cluster* yang dihasilkan dengan menggunakan penerapan *K-medoids* dengan hasil *cluster* yang dihasilkan pada tahap pengelompokan manual. Tabel 13 menunjukkan nilai akurasi untuk masing-masing *dataset* serta nilai akurasi rata-rata dari semua *dataset* yang digunakan pada penelitian ini.

Tabel 13 Nilai akurasi setiap *dataset*

Jenis DataSet	Akurasi (%)
DataSet1 (pjj0210)	89.4
DataSet2 (pjj0302)	83.7
DataSet3 (pjj0307)	100.0
DataSet4 (pjj0309)	100.0
Rata-rata akurasi	93.28

Berdasarkan Tabel 13 dapat dinyatakan bahwa algoritme *K-medoids* dinilai efektif untuk mengelompokan *dataset* yang berupa kode program C untuk mendeteksi kemiripan, dengan nilai akurasi rata-rata untuk seluruh *dataset* mencapai 93.28%.

SIMPULAN

Penelitian ini menerapkan algoritme *K-medoids* untuk mengelompokkan *dataset* kode program C sebagai tugas mata kuliah Algoritme dan Pemrograman. Sehingga dapat diketahui mahasiswa yang tugasnya memiliki kemiripan atau melakukan tindak penjiplakan. Hasil *clustering* pada Dataset1dengan jenis soal mengenai Kondisi If-Else dan Pengulangan For, di dapatkan bahwa *clustering* terbaik pada saat *K*=10 dengan nilai rata-rata *dissimilarity* untuk setiap objek sebesar 2.655 dengan sebanyak 18% dari total mahasiswa berada pada *cluster* yang sama, dan untuk DataSet2 dengan jenis soal mengenai Kondisi If-Else dan Pengulangan While didapatkan hasil *clustering* terbaik pada K=9 dengan nilai rata-rata *dissimilarity* untuk setiap objek dalam satu cluster sebesar 2.227 dengan 36.8% mahasiswa berada pada *cluster* yang sama.

Selanjutnya untuk DataSet3 dengan jenis soal mengenai Pengulangan While didapatkan hasil dimana *clustering* terbaik terjadi pada saat nilai K=2 dengan nilai rata-rata *dissimilarity* antar objek dalam satu *cluster* sebesar 0.719 dengan 87% mahasiswa berada pada kelompok yang sama. Sedangkan untuk DataSet4 *cluster* terbaik adalah pada saat *K*=6 dengan nilai rata-rata *dissimilarity* sebesar 3.199 dan sebanyak 61% mahasiswa berada pada *cluster* yang sama dan memiliki kemiripan dalam struktur kode program tugasnya. Apabila dibandingkan dengan

nilai huruf mutu Mata Kuliah Algoritme dan Pemrograman, pola dari alur penjiplakan dapat dilihat. Di setiap *cluster*, terdapat mahasiswa dengan huruf mutu tinggi dan rendah dan dapat diprediksi bahwa mahasiswa dengan nilai huruf mutu rendah melakukan penjiplakan tugas dari mahasiswa yang memiliki huruf mutu tinggi.

Penelitian ini telah berhasil menerapkan algoritme *K-medoids* dalam mendeteksi potensi penjiplakan Kode Program C sederhana. Pada penelitian selanjutnya, pendekatan *clustering* dapat dikembangkan untuk mendeteksi potensi penjiplakan pada kode program C yang lebih kompleks.

DAFTAR PUSTAKA

Burrow S. 2004. Efficient and effective plagiarism detection for large code repositories [tesis]. Melbourne (AU): RMIT University.

Chawla M. 2003. An indexing technique for efficiently detecting plagiarism in large volume of source code [tesis]. Melbourne (AU): RMIT University.

Gumilang AP. 2013. Pendeteksian penjiplakan kode program C dengan *K-means* [skripsi]. Bogor (ID): Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor.

Han J, and Kamber M, 2011, *Data Mining : Concept and techniques Ed ke-3.* Amsterdam (NL): Elsevier.

Notyasa A. 2013. Pendeteksian penjiplakan kode program C dengan *bisecting K-means* [skripsi]. Bogor (ID): Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor.