Jurnal Prespektif
P-ISSN: 2549-7111; E-ISSN: 2549-712X
UIN Sunan Gunung Djati Bandung

KUALITAS PERANGKAT LUNAK: MODULARITAS PUSTAKA *TEXT PRE-PROCESSING*

Dian Sa'adillah Maylawati[1]; Hamdan Sugilar[2]; Rully Agung Yudhiantara[3]

[1]Program Studi Informatika, Sekolah Tinggi Teknologi Garut;
dsaadillah@sttgarut.ac.id

[2] Pendidikan Matematika, UIN Sunan Gunung Djati Bandung;
hamdansugilar@uinsgd.ac.id

[3]Program Studi Pendidikan Bahasa Inggris, UIN Sunan Gunung Djati Bandung;
rully.agung@uinsgd.ac.id

Abstract

Perangkat lunak yang baik harus mencapai kualitas yang baik pula. Termasuk pustaka (*library*) dan komponen perangkat lunak yang akan dipanggil pada perangkat lunak lainnya. Terdapat beberapa faktor kualitas yang dapat dicapai pustaka perangkat lunak, antara lain fleksibilitas, modularitas, interoperabilitas, dan portabilitas. Penelitian ini bertujuan untuk mengukur faktor kualitas modularitas dari pustaka *text pre-processing* untuk Bahasa Indonesia. Metodologi pembangunan pustaka menggunakan *Waterfall* dengan menerapkan *Unified Modelling Language* (UML) sebagai model perangkat lunak. Pustaka *text pre-processing* dibangun dengan bahasa pemrograman Java dengan masukan berupa koleksi dokumen teks dan keluaran berupa Himpunan *Frequent Word Itemset* (HFWI) sebagai representasi teks terstruktur. Modularitas pustaka diukur dengan menggunakan metriks *Coupling between Object Classes* (CBO) dan *Lack of Cohesion in Methods* (LCOM). Hasil pengukuran dan evaluasi modularitas pustaka

menunjukkan bahwa modularitas pustaka sudah dicapai dengan kohesi yang cukup tinggi dan kopling yang rendah. Kohesi yang cukup tinggi ditunjukkan dengan nilai LCOM 0.381, sedangkan kopling yang rendah ditunjukkan dengan nilai CBO 1.045.

Kata Kunci: kualitas perangkat lunak, modularitas, pustaka perangkat lunak, *text pre-processing*.

1. Pendahuluan

Melihat begitu pesatnya kebutuhan terhadap pengelolaan data teks dengan teknik *text mining* muncul beragam aplikasi *Text Mining* seperti Weka, Kea, Mallet, LingPipe, GATE, Carrot2, MinorThird, Simmetrics, dan lain sebagainya. Membangun *library* yang mengemas *pre-processing* untuk *text mining* dapat bermanfaat dalam mengefisiensikan dan mempermudah proses *mining.* Terutama untuk Bahasa Indonesia yang dalam beberapa aplikasi *text mining* belum lengkap, bahkan belum tersedia. Adapun hasil dari *pre-processing library* yang dibangun adalah representasi terstruktur dalam bentuk Himpunan *Frequent Word Itemset* (HFWI) yang sudah dibuktikan baik dalam menjaga makna teks Bahasa Indonesia, terutama dari media sosial yang banyak mengandung bahasa alami (Maylawati, 2015) (Maylawati & Saptawati, 2016) (Maylawati, et al., 2016). Untuk memenuhi kebutuhan aplikasi text mining yang beragam, pustaka *text pre-processing* yang dibangun tentunya harus bersifat modular. Modularitas *library* dapat diukur dengan mengevaluasi atribut kualitas modularitas yaitu *cohesion* dan *coupling* (Alvaro, 2010) (Choi, 2008). *Cohesion* diukur menggunakan metriks *Lack Cohesion in Methods* (LCOM), sedangkan coupling diukur menggunakan metriks *Coupling between Object Classes* (CBO).

Perangkat lunak yang modular akan efektif apabila modul-modulnya bersifat mandiri sehingga lebih mudah untuk dibangun, diuji, dan dipelihar (Pressman, 2010). Kuncinya adalah functional independence (kemandirian fungsi) untuk desain yang baik, dan desain adalah kunci dari kualitas perangkat lunak (Pressman, 2010).Functional independence berasal dari modularitas dan konsep abstraksi dan

information hiding. Abstraksi membantu mendefinisikan prosedur atau informasi yang ada pada perangkat lunak, sedangkan information hiding berguna pada saat modifikasi maupun pemeliharaan perangkat lunak karena banyak data dan prosedur yang disembunyikan dapat mengurangi kesalahan yang tidak disengaja saat modifikasi yang cenderung merambat di dalam perangkat lunak (Pressman, 2010). Functional independence dapat diukur dengan dua kriteria kualitatif, yaitu kohesi yang mengukur kekuatan relatifitas setiap fungsi pada modul dan kopling yang mengukur relatifitas kebergantungan antar modul.

2. Metodologi Penelitian

Metode pada penelitian ini menerapkan *Waterfall* sebagai metodologi pembangunan perangkat lunak (Pressman, 2010) (Sommerville, 2011) (Ramdhani, 2013). Mulai dari tahapan elisitasi kebutuhan pustaka *text pre-processing,* analisis kebutuhan pustaka, desain, implementasi, dan pengujian pustaka. Tahap analisis dan desain dimodelkan menggunakan UML dengan memenuhi seluruh kebutuhan pustaka, antara lain kebutuhan masukan dan keluaran pustaka hingga fungsi-fungsi yang harus dimiliki pustaka. Pada tahap implementasi, pustaka dibangun dengan bahasa pemrograman Java. Selanjutnya pada tahap pengujian, pustaka diuji baik secara pengujian fungsional (*black box texting*) dan pengujian struktural (*white box testing*). *White box teting* pada penelitian ini salah satunya adalah dengan menguji, mengukur, dan mengevaluasi modularitas dari pustaka *text pre-processing*.

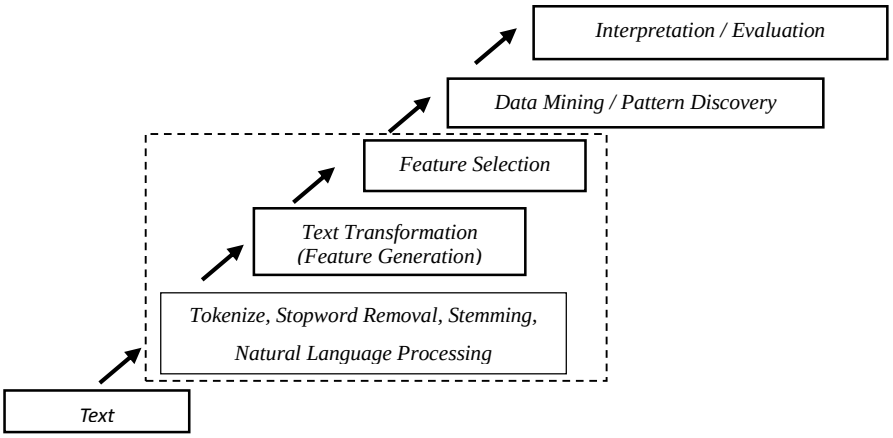
Pembangunan pustaka perangkat lunak dilakukan setelah melakukan studi terhadap literatur terhadap *text mining,* representasi terstruktur dari teks (dalam penelitian ini menggunakan HFWI), hingga mempelajari bagaimana membangun pustaka yang dapat memenuhi fator kualitas modularitas, sehingga pustaka dapat digunakan di aplikasi *text mining* dengan baik. Selanjutnya melakukan identifikasi dan analisis

setiap tahap dari *pre-processing* berupa *text pre-processing, feature generation*, dan *feature selection* dalam kaitannya menghasilkan representasi teks terstruktur.

3. *Pre-processing* pada *Text Mining*

Text mining adalah teknik untuk menghasilkan knowledge baru dari text dengan otomatis maupun semi otomatis, dimana knowledge yang di ekstrak tersebut bermanfaat dan berasal dari data text yang jumlahnya sangat besar (Torre, et al., 2008) (Zohar, 2002). Data yang dikelola dalam teknik text mining adalah data text yang tidak terstruktur. Perbedaan dari data mining dan text mining adalah proses ekstraksi feature atau pola yang berasal dari bentuk data yang berbeda. Pada data mining ekstraksi feature berasal dari data terstruktur, sedangkan pada text mining berasal dari data yang semi terstruktur, tidak dapat dikatakan sama sekali tidak terstruktur maupun dikatakan terstruktur, walaupun kebanyakan berasal dari data yang tidak terstruktur (Han & Kamber, 2006) (Carpineto, et al., 2009) (Maylawati, et al., 2017).

Gambar 1 menunjukkan tahap-tahap melakukan text mining, antara lain text pre-processing, text transformation (feature generation), feature selection, data mining/pattern discovery, dan implementation/evaluation (Zohar, 2002).Tahap text pre-processing adalah proses membersihkan data, pada proses tersebut juga dilakukan penandaan part of speech dan parsing dimana kalimat adalah graf yang berdiri sendiri. Selanjutnya dilakukan proses text transformation atau feature generation yang merupakan proses pembangkitan semua feature yang dapat berbentuk representasi text terstruktur berupa single words atau bag of words maupun multiple word. Selanjutnya tahap feature selection untuk menyeleksi feature yang masih redundant dan hanya feature yang berguna saja. Tahap data mining untuk memperoleh knowledge dari data text dengan menerapkan metode data mining, misalnya clustering, classification, atau association beserta teknik-teknik atau algoritma-algoritma data mining. Terakhir adalah tahap implementation/evaluation yaitu mengimplementasi dan mengevaluasi pola yang dihasilkan dari proses data mining untuk melihat akurasinya senhingga dapat dihasilkan knowledge yang sesuai.



Gambar 1. Tahap-tahap Text Mining (Zohar, 2002)

Tahap pre-processing pada text adalah tahap menyiapkan data text yang tidak terstruktur menjadi representasi data terstruktur sehingga data siap dilakukan proses selanjutnya (Mahgoub, et al., 2008).Hasil yang diharapkan dari pre-processing adalah representasi yang siap digunakan untuk proses mining dengan mentransformasi text menjadi represnetasi terstruktur dari proses ekstraksi hingga seleksi (Helena, 1999) (Doucet, 2005) (Honloor, 2011) (Srividhya & Anitha, 2010). Berdasarkan beberapa definisi di atas dan mengacu pada tahap-tahap text mining pada Gambar 1, maka tahap pre-processing terdiri dari *text-preprocessing, feature generation*, dan *feature selection*.

4. Modularitas sebagai Faktor Kualitas Pustaka Perangkat Lunak

Modularity perangkat lunak adalah atribut tunggal perangkat lunak yang mudah untuk dikelola dengan membagi-bagi perangkat lunak menjadi bagian-bagian kecil yang disebut dengan modul dan diintegrasikan untuk memenuhi requirement (Pressman, 2010). Perangkat lunak modular yang efektif memiliki lima kriteria (Meyer, 1997) (Pressman, 2010), antara lain:

a. Modular decomposability, masalah didekomposisi secara sistematis menjadi sub-masalah untuk mengurangi kompleksitas dari keseluruhan masalah, hal ini dapat menjadi solusi modular yang efektif.
b. Modular composability, jika modul dapat digunakan kembali dan diintergrasikan ke sistem yang baru maka termasuk kriteria modular.
c. Modular understandability, modul harus dapat dimengerti sebagai unit yang mandiri tanpa merujuk ke modul lain agar dapat lebih mudah digunakan dan lebih mudah menerima perubahan.
d. Modular continuity, jika ada perubahan requirement maka hanya perlu mengubah modul dibandingkan mengubah sistem yang lebih besar, hal ini membuat dampak perubahan dapat diminimalisasi.
e. Modular protection, jika terdapat masalah pada sebuah modul maka dampaknya hanya terhadap modul itu sendiri tidak berpengaruh pada modul lainnya atau sistem secara keseluruhan.

Pemrograman berorientasi objek secara alami mengikuti konsep desain perangkat lunak penting seperti abstraksi, information hiding, fuctional independence, dan modularitas. Berdasarkan kriteria dan desain efektif modularitas menurut Meyer yang telah dipaparkan di atas, terdapat lima prinsip desain untuk pemrograman berorientasi objek yang sesuai dengan modularitas (Meyer, 1997) (Pressman, 2010), antara lain *linguistic modular units, few interface, small interface (weak coupling), explicit interface,* dan *information hiding*. Modul dikatakan lingusitic modular units ketika bahasa pemrograman yang digunakan mendukung modularitas secara langsung seperti bahasa perograman berorientasi objek. Jumlah *interface* antar modul dan informasi yang digunakan antar interface sebaiknya seminimal mungkin (*few interface* dan *small interface*) agar kebergantungan antar modul kecil. Kemudian kapanpun modul perlu berkomunikasi, maka komunikasi dapat dilakukan secara langsung (*explicit interface*) dan menerapkan prinsip information hiding dengan

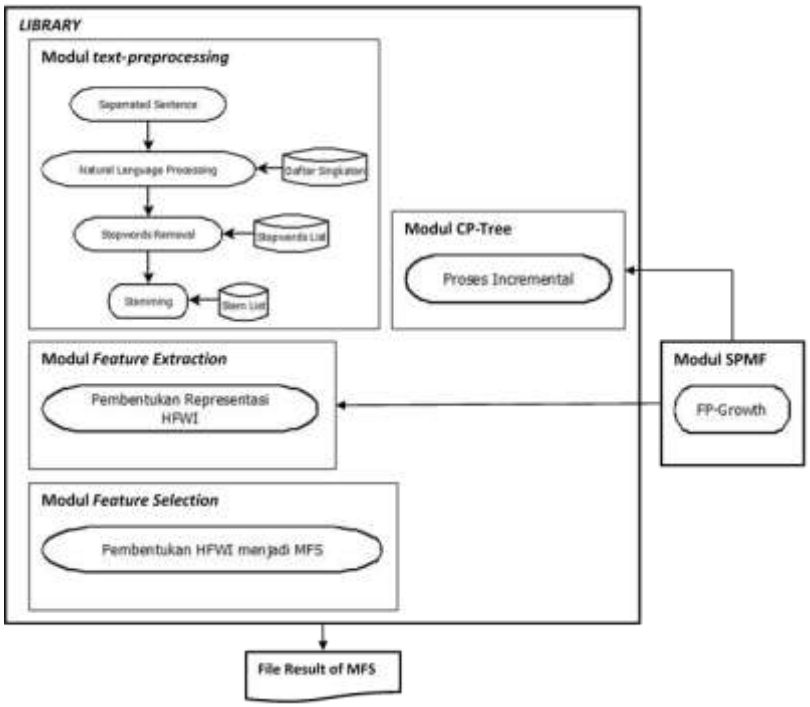
menyembunyikan informasi modul dari akses luar kecuali informasi yang secara khusus didefinisikan sebagai public information.

Pada software component quality model, modularity memiliki atribut kualitas cohesion dan coupling (Alvaro, 2010) (Choi, 2008). Cohesion atau kohesi adalah keterikatan fungsi-fungsi di dalam suatu modul yang memiliki fungsi-fungsi serupa dengan satu tanggung jawab. Sedangkan coupling atau kopling adalah kebergantungan antar modul satu dengan modul lainnya. Sebuah perangkat lunak yang baik memiliki kohesi yang tinggi dan kopling yang rendah. Menurut Alvaro et al, 2010, teknik evaluasi untuk modularity adalah dengan menganalisis cohesion dan coupling. Selanjutnya matriks yang digunakan untuk evaluasi cohesion dan coupling adalah CK Metrics. CK Metrics (Chidamber and Kemerer Metrics) adalah salah satu metriks untuk desain Object Oriented yang popular yang termasuk metriks untuk *complexity, coupling, dan cohesion*. Metrics yang digunakan di dalamnya ada enam, antara lain Weighted Methods Per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Response For a Class (RFC), Coupling between Object Classes (CBO), dan Lack of Cohesion in Methods (LCOM). Sesuai kebutuhan tesis, metrics yang digunakan adalah CBO dan LCOM.

5. Hasil dan Pembahasan

Library yang modular tentunya harus memenuhi kualitas software component modularity yang berarti memiliki kohesi tinggi dan kopling rendah. Kopling yang rendah dapat dimulai dari perancangan kelas-kelas dengan kebergantungan antar kelas yang seminimal mungkin, sedangkan kohesi tinggi juga dapat dimulai dari perancangan method atau fungsi-fungsi dalam kelas dengan keterikatan antar kelas yang sekecil mungkin. Tentunya perancangan kelas dan fungsi-fungsi dalam kelas harus diikuti dengan implementasi yang baik sehingga modularitas library tercapai. Library pre-processing yang bersifat modular juga mendukung fleksibilitas library itu

sendiri, karena dengan kelas-kelas dan fungsi-fungsi yang modular, library dapat dengan mudah digunakan di berbagai aplikasi text mining tanpa banyak modifikasi. Hal ini juga dikarenakan kelas-kelas atau fungsi-fungsi yang ada dapat digunakan secara mandiri sesuai kebutuhan masing-masing aplikasi text mining. Adapun pembangunan library dilakukan sesuai prinsip dan kriteria modularitas, antara lain:



Gambar 2. Arsitektur *Text Pre-Processing Library*

1. Mendekomposisi masalah menjadi sub masalah, mulai dari persoalan pre-processing didekomposisi menjadi *text pre-processing, feature extraction, dan feature* selection sesuai dengan use case yang dapat dilihat pada Gambar 3. Kemudian *text pre-processing* didekomposisi menjadi *tokenizing* (memisahkan

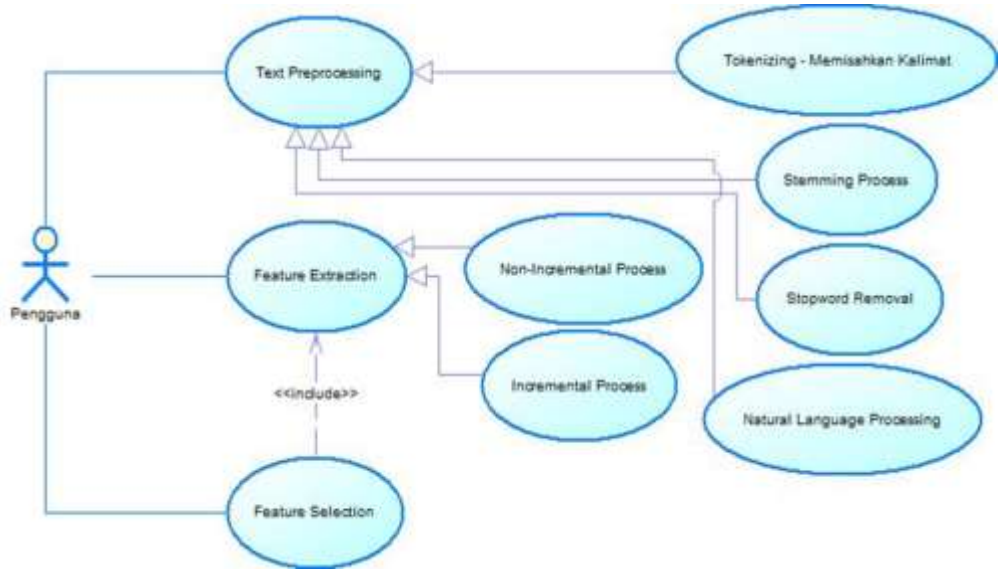
kalimat), *stopword removal*, *stemming*, dan *natural language processing*. Pada penelitian ini proses *Feature extraction* didekomposisi menjadi *non-incremental process* dan *incremental process*. Sedangkan *feature selection* tidak didekomosisi. Sub masalah dijadikan sebuah modul yang tidak saling bergantung satu dengan lainnya, tetapi memiliki interface untuk komunikasi antar modul. Setiap modul memilki fungsi-fungsi kesuai kebutuhan pustaka seperti yang digambarkan pada arsitektur pustaka pada Gambar 2.

2. Menggunakan bahasa pemrograman berorientasi objek yang secara natural mendukung prinsip modularitas. Masalah yang didekomosisi menjadi modul dalam pemrograman berorientasi objek ini dijasikan objek-objek yang saling berkomunikasi.
3. Membuat kelas-kelas abstrak dan kelas-kelas yang seminimal mungkin bergantung satu sama lainnya agar mencapai coupling yang rendah. Karena dengan kebergantungan antar kelas atau objek yang rendah implementasi dari setiap objek akan mudah, tanpa mempengaruhi objek lainnya.
4. Mendefinisikan fungsi-fungsi yang mandiri (fuctional independence) dengan keterkaitan antar fungsinya seminimal mungkin agar mencapai cohesion yang tinggi. Setiap fungsi harus memiliki dokumentasi yang baik agar dapat dimengerti, mudah digunakan, dan mudah dimodifikasi.
5. Menerapkan information hiding, pada bahasa pemrograman Java, saat perangkat lunak di bulid maka secara otomatis informasi dari fungsi-fungsi yang ada pada perangkat lunak akan disembunyikan.

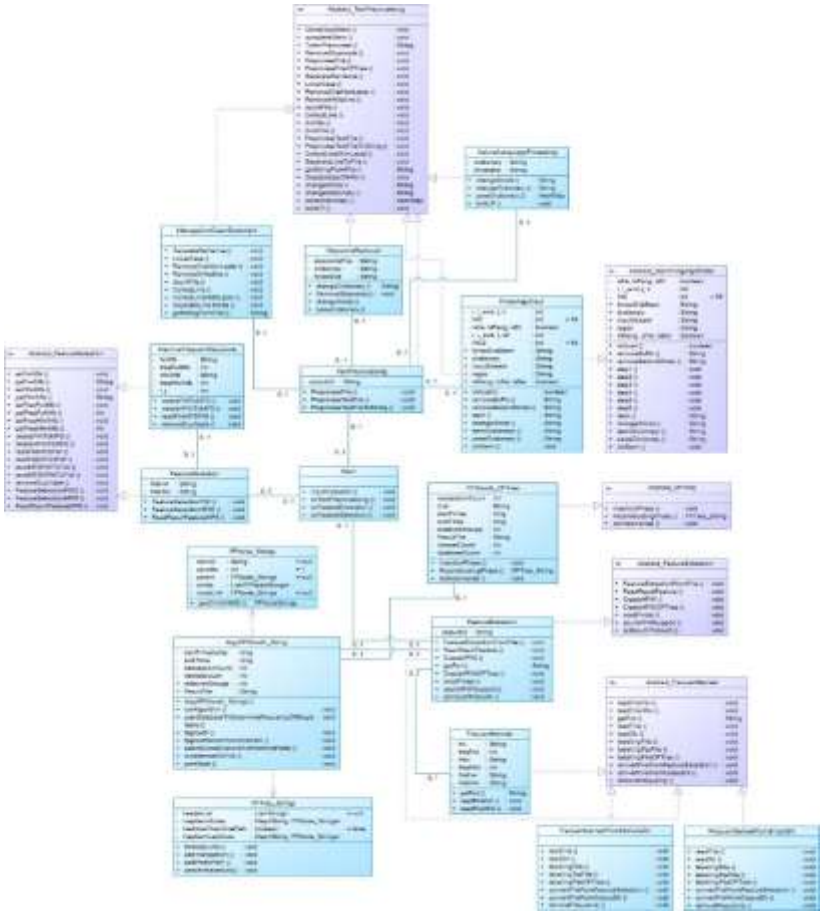
5.1. Analisis dan Desain Pustaka *Text Pre-processing*

Pustaka *Text Pre-processing* memiliki beberapa kebutuhan fungsional yang harus dipenuhi seperti pada Tabel 1. Kebutuhan pustaka *text pre-processing* dimodelkan dengan *use case diagram* yang terdapat pada Gambar 3 serta *class diagram* pada

Gambar 4. Adapun kebutuhan non fungsional pustaka *text pre-processing* yang dibahas pada artikel ini adalah *modularity.* Dimana, pustaka bersifat modular sehingga mudah untuk digunakan kembali, memiliki kopling yang rendah dan kohesi yang tinggi dengan rancangan kelas dan fungsi yang meminimalkan kebergantungan atau keterkaitan antar kelas maupun antar fungsi.



Gambar 3. Diagram *Use Case* untuk *Text Pre-Processing Library*



Gambar 4. Diagram *Class* untuk *Text Pre-Processing Library*

Tabel 1. Daftar Kebutuhan Fungsional Pustaka

Kode Kebutuhan Fungsional	Nama Kebutuhan	Deskripsi
LP-F-01	Terima koleksi dokumen	*Library* mampu menerima masukan koleksi dokumen *text* untuk disiapkan sebagai masukan pembentukan representasi terstruktur. Selanjutnya disebut dengan *input* 1.
LP-F-02	Terima hasil *preprocessing*	*Library* mampu menerima hasil *preprocessing* sebagai masukan pembentukan representasi *text* terstruktur. Selanjutnya disebut dengan *input* 2.
LP-F-03	Terima *support*	*Library* dapat menerima masukan parameter *support* dari pengguna untuk menghasilkan representasi HFWI. Selanjutnya disebut dengan *input* 3.
LP-F-04	Baca koleksi dokumen	*Library* mampu membaca seluruh *file* pada *input* 1.
LP-F-05	Pisah kalimat	*Library* mampu memisahkan kalimat pada koleksi dokumen sebagai kebutuhan representasi HFWI.
LP-F-06	*Lowercase*	*Library* mampu mengubah seluruh isi dokumen menjadi huruf kecil.
LP-F-07	Hilangkan *Stopwords*	*Library* mampu menghilangkan *stopwords* yang ada pada seluruh dokumen. Termasuk menghilangkan tanda baca, karakter selain huruf, symbol, *white space* dan *white line* yang berlebih pada *input* 1.
LP-F-08	*Stemming*	*Library* mampu mengubah isi *input* 1 ke bentuk kata dasarnya.
LP-F-09	Simpan hasil *Preprocessing*	*Library* mampu menyimpan hasil *preprocessing input* 1 ke dalam sebuah *file* sebagai *input* 2. Selanjutnya disebut dengan *file* 1.
LP-F-10	Baca *input 2*	*Library* mampu membaca *input* 2 sebagai masukan pembentukan representasi HFWI.

Kode Kebutuhan Fungsional	Nama Kebutuhan	Deskripsi
LP-F-11	Buat FWI	*Library* mampu membentuk representasi FWI dengan menggunakan algoritma FP-Growth dari *tools* SPMF dan algoritma CP-Tree. Selanjutnya disebut dengan *output* 1.
LP-F-12	Simpan FWI	*Library* mampu menyimpan representasi FWI yang selanjutnya disebut dengan *file* 2.
LP-F-13	Seleksi MFS dari FWI	*Library* mampu menyeleksi FWI yang telah terbentuk pada *file* 2 dan membentuk *set of* MFS.
LP-F-14	Simpan MFS dari FWI	*Library* mampu menyimpan MFS hasil seleksi *feature* ke dalam sebuah *file*. Selanjtnya disebut dengan *file* 3.
LP-F-15	Buat HFWI	*Library* mampu membentuk representasi HFWI dari *file* 3. Selanjutnya disebut dengan *output* 2.
LP-F-16	Simpan HFWI	*Library* mampu menyimpan representasi HFWI ke dala sebuah *file*. Selanjutnya disebut dengan *file* 4.
LP-F-17	Seleksi MFS dari HFWI	*Library* mampu menyeleksi HFWI yang telah terbentuk pada *file* 4 dan membentuk *set of* MFS dari HFWI.
LP-F-18	Simpan MFS dari HFWI	*Library* mampu menyimpan MFS hasil seleksi *feature* ke dalam sebuah *file*. Selanjtnya disebut dengan *file* 5.
LP-F-19	*Natural Languange Processing*	*Library* mampu melakukan perbaikan pada kata-kata yang disingkat dengan menggunakan NLP.

5.2. Evaluasi Pengukuran Kohesi dengan *Coupling between Object Classes* (CBO) dan *Lack Cohesion in Methods* (LCOM)

Definisi: CBO untuk sebuah kelas adalah menghitung jumlah kelas-kelas lain yang bergantung pada kelas tersebut. Dua buah kelas saling berelasi jika *method* dari kelas menggunakan instans *variable* atau *method* dari kelas lain. Dengan demikian metriks CBO dapat dikomputasi dengan menghitung jumlah kelas yang kelasnya saling berelasi dan menghitung semua *reference* dari atribut dan parameter dari *method* kelas tersebut. Namun, hubungan antar kelas tersebut adalah kelas yang bukan kelas turunannya. *Point of view* dari *metrics* ini adalah:

a. Terlalu banyak kopling antar kelas akan merugikan desain modular dan menghambat *reusability*. Semakin kelas tersebut mandiri, semakin mudah untuk di gunakan kembali oleh aplikasi lain.
b. Untuk meningkatkan *modularity* dan enkapsulasi, kebergantukan antar kelas atau kopling harus seminimal mungkin. Nilai kopling yang besar menujukkan sensitifitas yang tinggi jika terjadi perubahan, sehingga akan lebih sulit di *maintain.*
c. Pengukuran kopling berguna untuk menentukan seberapa kompleks pengujian yang dilakukan. Semakin tinggi kopling, maka pengujian yang dilakukan butuh ketelitian yang lebih tinggi.

Nilai batas atas metriks ini sampai saat ini belum ditentukan. Namun, berdasarkan pemaparan di atas semakin kecil nilai CBO maka semakin baik kualitas desain kelas tersebut.

Pengukuran dengan LCOM, misalkan sebuah class C1 dengan n buah methods M1, M2, …, Mn. Lalu {Ij} adalah himpunan *instance variable* yang digunakan oleh method Mi. Dengan demikian terdapat n buah himpunan I1, I2, …, In. Kemudian P = {Ii, Ij | Ii ∩ Ij = Ø} dan Q = {Ii, Ij | Ii ∩ Ij ≠ Ø}. Jika semua n himpunan {I1}, {I2}, …, {In} adalah Ø maka P = Ø. Untuk mendapatkan nilai LCOM, jika P>Q maka LCOM = P-Q selain itu LCOM=0. LCOM diharapkan rendah pada sebuah *class* (*cohesiveness* tinggi) karena menaikkan encapsulation. LCOM yang tinggi (*cohesiveness* rendah) menandakan sebuah *class* yang semestinya dipecah menjadi 2 atau lebih class. Selain itu, LCOM yang tinggi menandakan kompleksitas yang tinggi.

Sampai saat ini, belum dapat ditemukan berapa besar nilai batas atas metric ini. Akan tetapi, dari definisi Chidamber dan Kemerer, 1994, nilai LCOM bergantung pada jumlah methods pada sebuah class. Dengan demikian, terdapat nilai maksimum LCOM pada class tersebut, yaitu ketika |Q| = 0. Berdasarkan sudut pandang dan nilai maksimum LCOM dapat ditarik kesimpulan bahwa semakin kecil nilai aktual LCOM dibanding nilai maksimumnya, semakin baik cohesiveness class tersebut.

Evaluasi modularity dari library preprocessing yang dibangun dilakukan dengan menggunakan metriks CBO dan LCOM. Hasil pengukuran terdapat pada Tabel 2 dan Tabel 3. Keterangan kolom pada hasil metriks CBO dan LCOM antara lain:

a. No. : Nomor Urut
b. Kelas : Nama kelas pada kelas diagram library preprocessing
c. CBO : Nilai metriks CBO
d. P : Jumlah metod yang atributnya tidak beririsan dengan metod lain.
e. Q : Jumlan metod yang atributnya beririsan dengan metod lain.
f. LCOM : Nilai metriks LCOM, jika P>Q maka LCOM = P-Q, selain itu LCOM = 0.

Tabel 2. Hasil Metriks CBO

No.	Kelas	CBO
1.	Abstract_FeatureEkstraction	0
2.	Abstract_FeatureSelection	0
3.	Abstract_FeatureItemset	0
4.	Abstract_CPTree	0
5.	Abstract_TextPreprocessing	0
6.	Abstract_StemmingAlgoPorter	0
7.	Feature Extraction	3
8.	MaximalFrequentSequence	1
9.	FeatureSelection	2
10.	FrequentItemset	1

No.	Kelas	CBO
11.	FrequentItemsetFromMultipleDir	0
12.	FrequentItemsetFromSingleDir	0
13.	FPGrowth_CPTree_Scale	1
14.	ManageAndCleanDocument	1
15.	TextPreprocessing	4
16.	StopwordsRemoval	1
17.	PorterAlgoGaul	1
18.	Main	3
19.	AlgoFPGrowth_String	3
20.	FPNode_Strings	1
21.	FPTree_Strings	1
22.	NaturalLanguageProcessing	0
Jumlah CBO		**23**
Nilai Min. CBO		**0**
Nilai Max. CBO		**4**
Nilai Median		**1**
Nilai Mean		**1.0455**

Tabel 3. Hasil Metriks LCOM

No.	Kelas	P	Q	LCOM
1.	Abstract_FeatureEkstraction	0	7	0
2.	Abstract_FeatureSelection	0	18	0
3.	Abstract_FeatureItemset	0	11	0
4.	Abstract_CPTree	0	3	0
5.	Abstract_TextPreprocessing	0	20	0
6.	Abstract_StemmingAlgoPorter	0	22	0
7.	Feature Extraction	5	1	4
8.	MaximalFrequentSequence	0	4	0
9.	FeatureSelection	3	4	0
10.	FrequentItemset	0	3	0
11.	FrequentItemsetFromMultipleDir	4	5	0
12.	FrequentItemsetFromSingleDir	4	5	0
13.	FPGrowth_CPTree_Scale	24	3	21
14.	ManageAndCleanDocument	0	9	0
15.	TextPreprocessing	0	2	0

No.	Kelas	P	Q	LCOM
16.	StopwordsRemoval	1	3	0
17.	PorterAlgoGaul	1	6	0
18.	Main	0	10	0
19.	AlgoFPGrowth_String	15	4	11
20.	FPNode_Strings	0	2	0
21.	FPTree_Strings	2	1	1
22.	NaturalLanguageProcessing	1	3	0
Jumlah LCOM				**37**
Jumlah Method				**97**
Nilai Min. LCOM				**0**
Nilai Max. LCOM				**21**
Nilai Median				**0**
Nilai Mean				**0.381**

Berdasarkan Tabel 2 rata-rata nilai kopling library adalah 1,0455. Sedangkan Tabel 3 menunjukkan nilai kohesi library adalah 0.381. Walaupun untuk metriks CBO dan LCOM belum memiliki batas nilai yang pasti, berdasarkan kedua tabel tersebut dapat dikatakan kopling library cukup rendah dan nilai kohesi cukup tinggi. Hal ini menunjukkan hasil evaluasi library preprocessing memenuhi kualitas modularity.

6. Simpulan

Berdasarkan hasil evaluasi menggunakan matriks CBO dan LCO, pustaka *text pre-processing* yang dibangun pada penelitian ini dapat dikatakan memenuhi modularitas dengan kopling yang cukup kecil dan kohesi yang cukup tinggi. Perlu dilakukan pengembangan pada pustaka *text pre-processing* sehingga dapat lebih fleksibel dan modular digunakan oleh berbagai aplikasi *text mining* baik menambahkan fungsi-

fungsi maupun penyempurnaan kode sehingga pustaka *text pre-processing* dapat lebih efisien saat digunakan.

Daftar Pustaka

Alvaro, A., 2010. A Software Component Quality Framework. *ACM SIGSOFT Software ENgineering Notes*, November.35(1).

Carpineto, C., Osinski, S., Romani, G. & Weiss, D., 2009. A Survey of Web Clustering Engines. *ACM Computing Survey*, July.41(3).

Choi, Y. e. a., 2008. *Practical S/W Component Quality Evaluation Model*. s.l., s.n.

Doucet, A., 2005. *Advance Document Description, a Sequential Approach*, Finland: Academic Dissertation, Departement of Computer Science, Series of Publication A. Report A-2005-2, University of Helsinki.

Han, J. & Kamber, M., 2006. *Data Mining: Concepts and Techniques*. Second ed. Simon Fraser University: Morgan Kaufmann Publisher.

Helena, A., 1999. Knowledge Discovery in Document by Extracting Frequent Word Sequence. *Library Trends*, 48(1), pp. 160-181.

Honloor, A., 2011. *Sequential Pattern and Temporal Patterns for Text MIning*, New York: Graduate Faculty of Rensselaer Polytechnic Intitute, Major Subject: Computer Science.

Mahgoub, H., Rosner, D., Ismail, N. & Torkey, F., 2008. Text Mining Technique Using Association Rules Extraction. *International Journal of Information and Mathematical Science*, 4(1), pp. 21-28.

Maylawati, D. S., 2015. *Pembangunan Library Pre-processing untuk Text Mining dengan Representasi Himpunan Frequent Word Itemset. Studi Kasus Bahasa Gaul Indonesia*, Bandung: Bandung Institute of Technology.

Maylawati, D. S., Irfan, M. & Zulfikar, W. B., 2016. *Comparison between BIDE, PrefixSpan, and TRuleGrowth for Mining of Indonesian Text*. Medan, IOP Conference Series.

Maylawati, D. S., Ramdhani, M. A., Rahman, A. & Darmalaksana, W., 2017. *Incremental technique with set of frequent word item sets for mining large Indonesian text data.* Denpasar, s.n.

Maylawati, D. S. & Saptawati, G. A. P., 2016. *Set of Frequent Word Item sets as Feature Representation for Text with Indonesian Slang.* Medan, IOP Conf. Series: Journal of Physics.

Meyer, B., 1997. *Object-Oriented Software Construction.* Second ed. s.l.:Prentice Hall.

Pressman, R. S., 2010. *Software Engineering, A Practitioner's Approach.* 7 ed. New York: McGraw-Hill.

Ramdhani, M. A., 2013. *Metodologi Penelitian dalam Riset Teknologi Informasi.* Bandung: UIN Sunan Gunung Djati Bandung.

Sommerville, I., 2011. *Software Engineering.* 9 ed. Massachusetts, United State, America: Addison-Wesley.

Srividhya, V. & Anitha, R., 2010. Evaluating Preprocessing Techniques in Text Categorization. *International Journal of Computer Science ad Application Isues 2010.*

Torre, C. J., Matrin-Bautista, M. J. S. & Blanco, I. D., 2008. Text Knowledge Mining: And Approach to Text Mining. *ESTYLF08,* 17-19 September.

Zohar, Y. E., 2002. *Introduction to Text Mining.* [Online] Available at: http://www.docstoc.com/docs/25443990/Introduction-to-Text-Mining
[Accessed 2014].