SINONIM DAN *WORD SENSE DISAMBIGUATION* UNTUK MELENGKAPI DETEKTOR PLAGIAT DOKUMEN TUGAS AKHIR

Devi Dwi Purwanto

Sistem Informasi Sekolah Tinggi Teknik Surabaya

E-mail: devi@stts.edu

Abstract

Plagiarism can be categorized into several levels: *carbon copy*, the addition of words, word substitutions, changing active into passive sentences, and paraphrase. In this research, the detection is only performed by local similarity assessment method. This research is categorized into 3 major processes: preprocessing, candidate determination, and calculation of similarity. In preprocessing, extraction and conversion of a PDF file into XML is performed. Stopword removal and stemming are also performed in this process. For Candidates determination, the process used VSM (Vector Space Model) algorithm using Lucene.NET. It will then calculate the similarity values of the candidates. Similarity values that meet the *threshold* will be processed in the third stage. The next process is detecting plagiarism at the level of *carbon copy*. The plagiarism of the substitution level will be determined by finding synonymous with Lesk algorithm and utilizing WordNet as a language dictionary. Lesk notice the words around it, before doing the search process is synonymous with Lesk, performed first sentence extractor. From this experiment, it is concluded that the determination of synonyms using WordNet and Lesk algorithm does not seem to increase its similarity value role. This is due to the difficulty of finding plagiarism by just substituting words. However, plagiarism at the level of *carbon copy* can be handled with the help of sentence matching.

Keywords: *Plagiarism, Synonym, Word Sense Disambiguation, VSM, Lesk*

Abstrak

Plagiarisme dapat dibagi menjadi beberapa tingkatan yaitu *carbon copy*, penambahan kata, substitusi kata, pengubahan kalimat aktif menjadi pasif, dan paraphrase. Pada penelitian ini deteksi hanya dilakukan dengan metode *local similarity assessment*. Deteksi plagiat dibagi menjadi 3 proses besar yaitu preproses, penentuan kandidat, dan perhitungan *similarity*.

Pada preproses dilakukan ekstrasi file PDF dan mengubahnya ke dalam XML, *stopword removal* dan *stemming*. Untuk penentuan kandidat digunakan algoritma VSM (Vector Space Model) dengan bantuan library Lucene.NET kemudian dihitung nilai *similarity*. Nilai *similarity* yang memenuhi *threshold* akan diproses pada tahap ketiga. Proses selanjutnya melakukan deteksi plagiat pada level *carbon copy*. Sedangkan penentuan plagiat pada tingkat substitusi dilakukan dengan mencari sinonim dengan algoritma Lesk dan memanfaatkan Wordnet Bahasa sebagai kamus. Karena Lesk memperhatikan kata yang ada di sekitarnya, sebelum melakukan proses pencarian sinonim dengan Lesk, dilakukan *sentence extractor* terlebih dahulu.

Dari penelitian ini didapatkan bahwa penentuan sinonim dengan menggunakan Algoritma Lesk dan Wordnet tidak tampak peranannya dalam meningkatkan nilai *similarity*. Hal ini dikarenakan oleh sulitnya menemukan tindakan plagiat dengan hanya melakukan substitusi kata. Namun demikian plagiat dapat ditangani pada tingkat *carbon copy* dengan bantuan *sentence matching.*

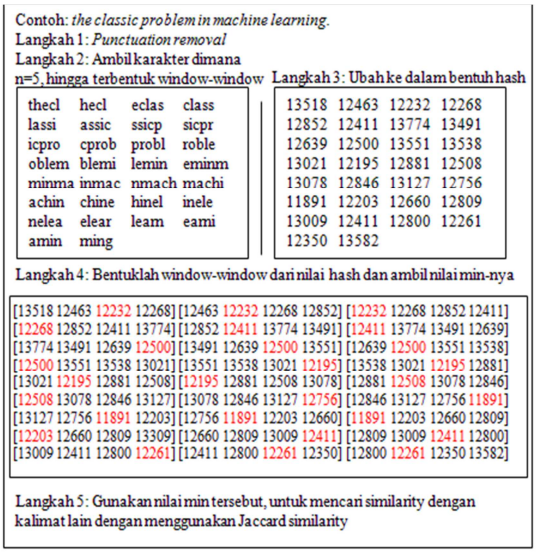
Kata Kunci: *Plagiat, Sinonim, Word Sense Disambiguation, VSM, Lesk*

1. Pendahuluan

Plagiarisme dapat dibagi menjadi beberapa tingkatan/level yaitu *carbon copy*, penambahan kata, substitusi kata, pengubahan kalimat aktif menjadi pasif, dan parafrase. Untuk mengetahui tindakan plagiat terdapat dua metode yaitu *local similarity assessment* dan *global similarity assessment*. Dimana pada penelitian ini metode yang digunakan adalah *local similarity assessment* untuk mendeteksi plagiat pada tingkat *carbon copy* dan substitusi, dimana data yang diuji cobakan adalah PDF tugas akhir.

2. Tinjauan Pustaka

Beberapa penelitian mengenai deteksi plagiat telah dilakukan sebelumnya diantaranya oleh Mario Zechner [**1**] dimana pada penelitian tersebut menggunakan pendekatan IR sistem dengan menggunakan *inverted index*. Penelitian berikutnya dilakukan oleh Eko Nugroho dengan melakukan pendeteksian plagiat menggunakan algoritma Rabin-Karp [**2**], dimana algoritma tersebut digunakan untuk melakukan deteksi plagiarisme dokumen teks dengan melakukan pencocokan *string/terms*. Alasan peneliti menggunakan algoritma ini karena dirasa cocok untuk pola pencarian jamak (*multiple pattern search*). Peneliti menggunakan metode pendeteksi plagiarisme milik Stein **[4]** dimana dibedakan menjadi 3 (tiga) yaitu perbandingan teks lengkap, dokumen *fingerprinting* atau kesamaan kata kunci. Selain menggunakan rabin karp terdapat beberapa penelitian, dimana untuk mendapatkan *fingerprint* menggunakan teknik *hashing* (dimana tiap kata diubah ke dalam angka), contohnya, Winnowing, dan Manber. Contoh kerja dari *winnowing* tersebut dapat dilihat pada Gambar 1.


Contoh: *the classic problem in machine learning.*
Langkah 1: *Punctuation removal*
Langkah 2: Ambil karakter dimana n=5, hingga terbentuk window-window

thecl	hecl	eclas	class
lassi	assic	ssicp	sicpr
icpro	cprob	probl	roble
oblem	blemi	lemin	eminm
minma	inmac	nmach	machi
achin	chine	hinel	inele
nelea	elear	learn	earni
amin	ming		

Langkah 3: Ubah ke dalam bentuh hash

13518	12463	12232	12268
12852	12411	13774	13491
12639	12500	13551	13538
13021	12195	12881	12508
13078	12846	13127	12756
11891	12203	12660	12809
13009	12411	12800	12261
12350	13582		

Langkah 4: Bentuklah window-window dari nilai hash dan ambil nilai min-nya

[13518 12463 12232 12268] [12463 12232 12268 12852] [12232 12268 12852 12411]
[12268 12852 12411 13774] [12852 12411 13774 13491] [12411 13774 13491 12639]
[13774 13491 12639 12500] [13491 12639 12500 13551] [12639 12500 13551 13538]
[12500 13551 13538 13021] [13551 13538 13021 12195] [13538 13021 12195 12881]
[13021 12195 12881 12508] [12195 12881 12508 13078] [12881 12508 13078 12846]
[12508 13078 12846 13127] [13078 12846 13127 12756] [12846 13127 12756 11891]
[13127 12756 11891 12203] [12756 11891 12203 12660] [11891 12203 12660 12809]
[12203 12660 12809 13309] [12660 12809 13009 12411] [12809 13009 12411 12800]
[13009 12411 12800 12261] [12411 12800 12261 12350] [12800 12261 12350 13582]

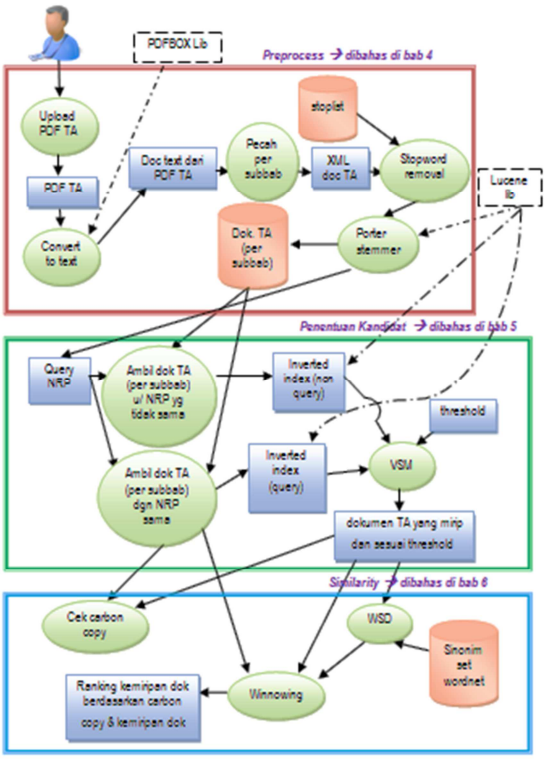
Langkah 5: Gunakan nilai min tersebut, untuk mencari similarity dengan kalimat lain dengan menggunakan Jaccard similarity

Gambar 1. Contoh Cara Winnowing

Pada dua penelitian yang telah dibahas pada tinjauan pustaka, keduanya hanya melakukan deteksi plagiat pada tingkat *carbon copy*, sedangkan untuk deteksi plagiat pada tingkat subtitusi belum dilakukan. Pada penelitian ini penulis mencoba melakukan deteksi pada tingkat subtitusi dengan menggunakan bantuan Wordnet Bahasa Indonesia untuk mendapatkan sinonim dari masing-masing kata sesuai dengan konteks artinya dengan memperhatikan kata yang ada di sekitarnya.

Urutan preproses yang dilakukan dalam deteksi plagiat tugas akhir dapat dilihat pada Gambar 2. Dimana pada bagian tersebut terdapat hubungan langsung dengan *user* yang melakukan *upload* dokumen tugas akhir.

Urutan preproses meliputi *upload* PDF tugas akhir, konversi dokumen PDF ke *text*, melakukan pemecahanan per subbab dimana didalamnya meliputi pembacaan daftar isi, merubah ke dalam format XML, melakukan *stopword removal*, melakukan *stemming*, dan memasukkan dokumen yang telah dipecah tersebut ke dalam *database*.



Gambar 2. Arsitektur Deteksi Plagiat

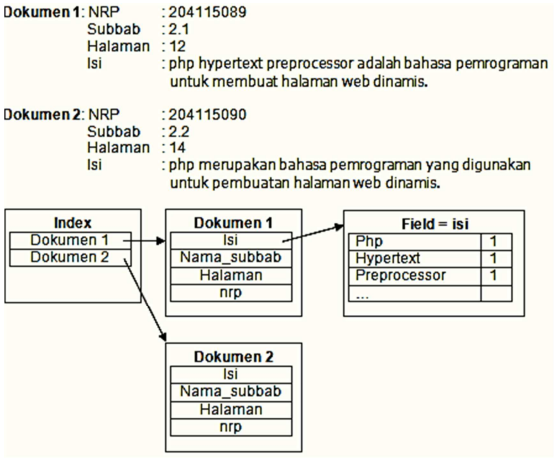
3. Penentuan Kandidat dengan VSM

Pada bagian ini dibahas mengenai cara untuk menentukan kandidat dokumen yang dicurigai sebagai plagiat. Untuk penentuannya dicari dengan menggunakan VSM (*Vector Space Model*). Hal ini dikarenakan VSM dapat digunakan untuk menentukan tingkat kemiripan dengan memperhatikan term yang ada. Proses penentuan kandidat yang meliputi melakukan pengambilan dokumen tugas akhir per subbab sesuai dengan NRP *query*, melakukan pengambilan dokumen tugas akhir per subbab selain NRP *query*, melakukan *indexing*, dan penentuan kandidat dengan Vector Space Model dari indeks yang ada.

Untuk pengindeksan dapat dilihat ilustrasinya pada Gambar 3. Dimana *indexing* tersebut

nantinya akan diambil *term vector*-nya untuk perhitungan nilai *similarity* yang menentukan kandidat tersebut akan diproses selanjutnya atau tidak. Penentuan kandidat ini dimaksudkan agar tidak semua dokumen dicari sinonim kata dan dihitung nilai *similarity*-nya.

Pada beberapa percobaan ditetapkan bahwa *threshold* nilai VSM yang diambil lebih besar sama dengan 0,80. Hal ini disebabkan jika semakin kecil nilai *threshold* akan semakin besar kandidat yang diterima sehingga waktu deteksi juga semakin lama, sebaliknya jika nilai VSM yang diambil terlalu tinggi maka kandidat yang diterima semakin sedikit, sehingga kadang deteksi plagiat pada subtitusi kata tidak terdeteksi.



Gambar 3. Ilustrasi Indexing pada Lucene.NET

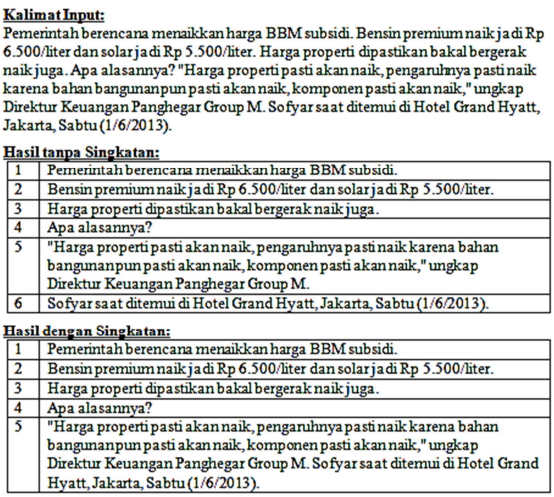
4. Perhitungan *Similarity* Setelah Kandidat Ditentukan

Terdapat 2 tahapan penting yaitu untuk melakukan cek plagiat apakah termasuk *carbon copy* dengan *threshold* n kalimat, dan melakukan pengujian plagiat dengan mengganti sinonim berdasarkan *word sense disambiguation* dari kalimat tersebut.

Untuk menentukan sinonim dari kata tersebut dibutuhkan Wordnet.NET untuk membantu memudahkan pengaksesan data pada Wordnet. Selanjutnya setelah mendapatkan sinonim dari kata tersebut, langkah selanjutnya adalah melakukan pengujian kemiripannya dengan menggunakan Algoritma Winnowing. Sehingga dari situ didapatkan ranking kemiripan dokumen.

Pendeteksian *carbon copy* disini maksudnya adalah melakukan pengujian sebanyak n kalimat apakah sama persis dengan dokumen yang lain dimana ditetapkan n=3.

Untuk mengetahui bahwa kumpulan kata tersebut merupakan sebuah kalimat, dilakukan dengan cara melakukan *sentence extractor* atau ekstraksi kalimat. Ekstraksi kalimat adalah teknik yang digunakan untuk *summarization* teks secara otomatis. *Sentence extractor* yang digunakan menggunakan algoritma genetik [**3**] dengan penambahan *list* singkatan. Contoh hasil *sentence extractor* dengan penambahan deteksi singkatan tersebut dapat dilihat pada Gambar 4.



Kalimat Input:

Pemerintah berencana menaikkan harga BBM subsidi. Bensin premium naik ja di Rp 6.500/liter dan solar ja di Rp 5.500/liter. Harga properti dipastikan bakal bergerak naik juga. Apa alasannya? "Harga properti pasti akan naik, pengaruhnya pasti naik karena bahan bangunan pun pasti akan naik, komponen pasti akan naik," ungkap Direktur Keuangan Panghegar Group M. Sofyar saat ditemui di Hotel Grand Hyatt, Jakarta, Sabtu (1/6/2013).

Hasil tanpa Singkatan:

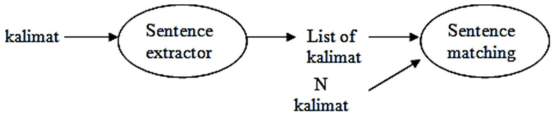
1	Pemerintah berencana menaikkan harga BBM subsidi.
2	Bensin premium naik ja di Rp 6.500/liter dan solar ja di Rp 5.500/liter.
3	Harga properti dipastikan bakal bergerak naik juga.
4	Apa alasannya?
5	"Harga properti pasti akan naik, pengaruhnya pasti naik karena bahan bangunan pun pasti akan naik, komponen pasti akan naik," ungkap Direktur Keuangan Panghegar Group M.
6	Sofyar saat ditemui di Hotel Grand Hyatt, Jakarta, Sabtu (1/6/2013).

Hasil dengan Singkatan:

1	Pemerintah berencana menaikkan harga BBM subsidi.
2	Bensin premium naik ja di Rp 6.500/liter dan solar ja di Rp 5.500/liter.
3	Harga properti dipastikan bakal bergerak naik juga.
4	Apa alasannya?
5	"Harga properti pasti akan naik, pengaruhnya pasti naik karena bahan bangunan pun pasti akan naik, komponen pasti akan naik," ungkap Direktur Keuangan Panghegar Group M. Sofyar saat ditemui di Hotel Grand Hyatt, Jakarta, Sabtu (1/6/2013).

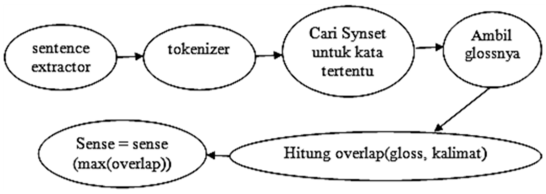
Gambar 4. Contoh Hasil Sentence Extractor

Setelah mendapatkan kumpulan kalimat, langkah selanjutnya yang dilakukan adalah melakukan pengujian *carbon copy*. Untuk lebih jelasnya pada pada Gambar 5akan diberikan tahapan yang dilakukan untuk cek *carbon copy*.



Gambar 5. Proses cek *carbon copy*

Setelah melakukan cek *carbon copy*, berikutnya adalah mendapatkan *synset* untuk kata tertentu dan kategori tertentu, langkah pertama yang harus dilakukan adalah mengambil *gloss* dari tiap *sense*. Dari *gloss* tersebut nantinya akan digunakan sebagai input untuk mencari *sense* yang tepat dengan Algoritma Lesk. Pada gambar 6 diberikan *flowchart* untuk menentukan sinonim yang tepat dari suatu kata berdasarkan kata yang ada di sekelilingnya.



Gambar 6. Proses pencarian *sense* yang sesuai

Langkah selanjutnya setelah melakukan pencarian sinonim, yang bertujuan untuk mengetahui plagiat pada tingkat substitusi. Dimana untuk menentukan sinonimnya digunakan Algoritma Lesk yang dapat dilihat pada Gambar 7.

1. Lakukan *sentence extractor*
2. Carilah synset dari tiap kata hasil pemecahan dari nomor 1
3. Ambil *gloss* pada masing-masing *sense* yang didapat
4. Lakukan perhitungan *overlap*(kalimat,*gloss*)
5. Ambil nilai *overlap* tertinggi dan jadikan sebagai *sense* kata tersebut
6. Ganti kata tersebut dengan kata paling awal dari *sense* tersebut

Gambar 7. Menentukan *Sense* yang Tepat

Untuk pengujian terhadap sinonim, pada Gambar 8 diberikan sebuah contoh kasus dan hasil penentuan *sense* dengan menggunakan Algoritma Lesk.

AWAL:

Kalimat1: Dimana dari koleksi dokumen tersebut akan dilakukan pembacaan terlebih dahulu terhadap daftar isi sehingga dapat diketahui bab dan subbabnya

Kalimat2: Dimana dari koleksi dokumen tersebut akan dilakukan pembacaan terlebih sebelumnya terhadap perincian isi sehingga dapat terungkap bab dan subbabnya

PENENTUAN SENSE DAN NILAI OVERLAP PADA ALGORITMA LESK:

Kalimat1:
Dimana dari[noun-1;0] koleksi[noun-1;1] dokumen[noun-2;1]
tersebut[verb-1;0] akan[noun-1;0] dilakukan pembacaan
terlebih[adv-1;0] dahulu[noun-3;2] terhadap[adv-1;0]
daftar[noun-4;3] isi[noun-2;1] sehingga dapat[verb-1;0]
diketahui[verb-4;1] bab[noun-1;0] dan[noun-1;0] subbabnya

Kalimat2:
Dimana dari[noun-1;0] koleksi[noun-1;1] dokumen[noun-2;1]
tersebut[verb-1;0] akan[noun-1;0] dilakukan pembacaan
terlebih[adv-1;0] sebelum[noun-1;1] terhadap[adv-1;0]
perincian[noun-9;2] isi[noun-2;1] sehingga dapat[verb-1;0]
terungkap[verb-2;1] bab[noun-1;0] dan[noun-1;0] subbabnya

Gambar 8. Contoh dan proses penentuan *sense* pada Algoritma Lesk

Untuk nilai *overlap* yang tidak lebih besar dari nol, maka penentuan *sense* diabaikan, hal ini dikarenakan terdapat banyak *sense* yang tidak memiliki *gloss*. Sedangkan untuk nilai *overlap* yang lebih besar dari nol, telah diberikan kandidat *sense* tersebut dan kata tersebut akan diganti dengan kata yang terdepan dari *sense* kandidat tersebut. Sehingga dari contoh pada Gambar 8 tersebut didapatkan kalimat yang baru setelah mengganti kata sinonim menjadi sebagai berikut:

Kalimat1:
Dimana dari analekta brevet tersebut akan dilakukan pembacaan terlebih dahulu terhadap daftar akar sehingga dapat dapat diungkapkan bab dan subbabnya

Kalimat2:
Dimana dari analekta brevet tersebut akan dilakukan pembacaan terlebih dahulu terhadap daftar akar sehingga dapat dapat diungkapkan bab dan subbabnya

Gambar 9. Contoh penentuan *sense* pada Algoritma Lesk setelah mengganti sinonim

Setelah mengganti kalimat yang baru dengan sinonim langkah selanjutnya adalah melakukan pengujian hasil sinonim tersebut dengan menggunakan Algoritma Winnowing. Winnowing berguna untuk melakukan deteksi plagiat dengan menyalin sebagian *(detecting partial copies)*. Pada Algoritma Winnowing dibutuhkan *hash* untuk mengubah kata tersebut menjadi sebuah angka yang kemudian disebut dengan *hash*. Selain *hash* dibutuhkan pula n-gram, jumlah window, dan *fingerprint*.

Alasan menggunakan nilai *hash* pada *difference algorithm* adalah agar lebih efisien, karena kemungkinan urutan dan nilai ASCII untuk tiap huruf berbeda. Nilai *hash* pada umumnya digambarkan sebagai *fingerprint* yaitu *string* pendek yang terdiri dari huruf yang terlihat acak. Untuk perhitungan nilai *hash* dilakkan dengan rumus berikut.

$$c_1 * b^{k-1} + c_2 * b^{k-2} * \ldots + c_{k-1} * b + c_k \qquad (1)$$

Dimana pada persamaan 1, c_k adalah nilai ASCII pada karakter yang ke-k. Sedangkan b adalah basisnya dalam bilangan prima, dan k adalah banyaknya karakter.

Hasil dari *winnowing* tersebut nantinya dapat diukur dengan menggunakan koefisien Jackard. Nilai koefisien Jackard yang mendekati 1 dapat diduga sebagai bentuk plagiat, dikarenakan mirip dengan dokumen pembanding. Pada *winnowing* hanya membandingkan dua buah dokumen, sehingga untuk membandingkan banyak dokumen harus dilakukan perulangan sebanyak dokumen tersebut.

Koefisien Jackard digunakan untuk menghitung kesamaan dari sekumpulan dokumen. Pada bagian ini koefisien Jackard digunakan untuk menghitung *fingerprint* yang *overlap* antara kumpulan dokumen dengan kandidat dokumen yang diduga sebagai plagiat. Rumus dari koefisien Jackard adalah:

$$\frac{|A \cap B|}{|A \cup B|} = \frac{M_{11}}{M_{01}+M_{10}+M_{11}} \qquad (2)$$

Pada persamaan 2, A dan B pada kasus ini adalah kumpulan *fingerprint* dari masing-masing dokumen.

5. Uji Coba

Uji coba yang dilakukan meliputi performansi deteksi plagiat untuk *carbon copy* dan substitusi kata, uji coba *threshold* VSM, dan uji coba jumlah n-gram. Pada TABEL I diberikan data yang digunakan untuk uji coba. Dimana dokumen yang digunakan sebagai koleksi dokumen adalah PDF tugas akhir sebanyak 100 buah.

Dari data uji coba yaitu 30 PDF buku tugas akhir dengan *threshold* >=0,8 dengan menggunakan *similarity* tanpa sinonim didapatkan bahwa terdapat 85 subbab yang diduga sebagai plagiat. Sedangkan pengujian terhadap 30 PDF buku tugas akhir dengan menggunakan sinonim dan *threshold* yang sama didapatkan bahwa terdapat 84 subbab yang diduga sebagai plagiat. Dari data tersebut terdapat 8 subbab dimana memiliki hasil *similarity* yang lebih mirip dengan menggunakan sinonim. Daftar unit teks yang memiliki nilai *similarity* lebih baik tersebut dapat dilihat pada TABEL II.

TABEL I
DAFTAR UJI COBA DETEKSI PLAGIAT

Keterangan			Hasil (buah)
PDF Tugas Akhir untuk Koleksi Dokumen, dengan perincian jurusan:			100
D3-Informatika	:	5 buah	
S1-Informatika	:	79 buah	
S1-Teknik Elektro	:	3 buah	
S1-Teknik Industri	:	11 buah	
S1-Sistem Informasi Bisnis	:	2 buah	
Subbab untuk Koleksi Dokumen (unit teks)			7337
Dokumen uji coba tahun 2012, dengan perincian periode:			31
Februari 2012	:	3 buah	
Juni 2012	:	1 buah	
Agustus 2012	:	9 buah	
Oktober 2012	:	9 buah	
November 2012	:	9 buah	
Subbab untuk uji coba tahun 2012 (unit teks)			2183

TABEL II
DAFTAR *SIMILARITY* YANG MEMILIKI HASIL LEBIH BAIK DENGAN SINONIM

No	ID Sim	NRP	Subbab	NRP Mirip	Subbab Mirip	Sim Non Sinonim	Sim Sinonim
1	3190	TGPOXGGEL	3.2.2	TGPOXGGXE	3.2.1	99.08	99.65
2	3870	TGPOXGGXE	3.2.1	TGPOXGGEL	3.2.2	99.08	99.65
3	4475	TGPOXGGIL	1.5.2	TGPOXGGEL	1.3.4	93.83	94.50
4	3011	TGPOXGGEL	1.3.4	TGPOXGGIL	1.5.2	93.83	94.50
5	4011	TGPOXGGXE	5.4	TGPOXGGEL	5.4	89.77	92.23
6	3377	TGPOXGGEL	6.9	TGPOXGGXE	6.6	86.02	86.61
7	3370	TGPOXGGEL	6.5	TGPOXGGXE	6.4	79.55	80.37
8	4214	TGPOXGGXE	6.4	TGPOXGGEL	6.5	79.55	80.37

Sedangkan sisanya terdapat 55 subbab yang memiliki hasil *similarity* yang kurang baik bila menggunakan sinonim (atau dengan kata lain *similarity* dengan tanpa menggunakan sinonim lebih baik), dan 21 subbab memiliki hasil *similarity* yang sama baik menggunakan sinonim ataupun tanpa mengunakan sinonim (hal ini dikarenakan kata-kata pada kalimat tersebut sama persis). Dimana penurunan *similarity* yang dihasilkan dengan sinonim rata-rata sebesar 3.27 poin.

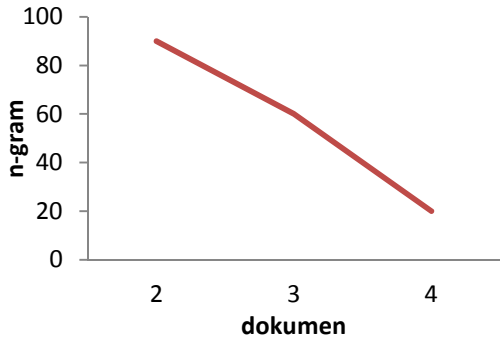
Alasan menggunakan *threshold* 0,8 dikarenakan performansi terhadap waktu dan hasil yang didapatkan. Pada tabel 3, koleksi dokumen yang digunakan aalah 7337, dengan beberapa percobaan *threshold* didapatkan kesimpulan dimana *threshold* yang semakin kecil didapatkan kandidat yang semakin banyak dan waktu yang semakin lama, sedangkan ketika kandidat tersebut diproses untuk cek *carbon copy* dan *similarity*, didapatkan jumlah subbab yang terdeteksi *carbon copy* yaitu 84. Sehingga diambil *threshold* yang lebih tinggi agar waktu yang dibutuhkan relatif lebih singkat.

TABEL III
UJI COBA PERFORMANSI *THRESHOLD* VSM

Threshold	Waktu (menit)	Kandidat (subbab)
0.6	158,32	103
0.7	145,56	96
0.8	120,03	85

Percobaan selanjutnya adalah jumlah n-gram yang digunakan. Pada

Gambar 10 diberikan grafik percobaan hasil deteksi.



Gambar 10. Uji Coba n-gram terhadap Dokumen

6. Kesimpulan

Setelah melakukan percobaan pada dokumen tugas akhir didapatkan beberapa kesimpulan dari penelitian ini, yaitu:

Pencarian sinonim dengan algoritma Lesk *word sense* disambiguation serta pemanfaatan wordnet pada penelitian ini tidak tampak peranannya dalam meningkatkan nilai *similarity*. Hal ini dikarenakan sulitnya menemukan tindakan plagiat dengan hanya melakukan substitusi kata.

Deteksi plagiat pada penelitian ini dapat menangani plagiat pada tingkat *carbon copy*, karena dengan menggunakan *sentence matching*, kalimat yang sama persis sebanyak 3 kalimat akan terdeteksi.

Nilai n-gram dan *window* sangat berpengaruh pada perhitungan *similarity*. Semakin besar nilai n-gram dan *window*, akan semakin kecil nilai *similarity*-nya. Pada kasus ini digunakan bigram.

Penentuan kandidat dengan menggunakan VSM dapat membantu proses pencarian *similarity*, sehingga tidak perlu melakukan proses pencarian *similarity* untuk seluruh unit teks.

Kesalahan penulisaan atau ejaan pada kata dapat menyebabkan nilai *similarity* dan pengujian terhadap *carbon copy* menjadi menurun. Hal ini dikarenakan pengujian tersebut tidak memperhatikan perbaikan pada ejaan.

Referensi

[1] M. Zechner, M. Muhr, R. Kern, and M. Granitzer, "External and Intrinsic Plagiarism Detection Using Vector Space Models, Stein, Rosso, Stamatatos, Koppel," in *Spanish Society for Natural Language Processing 2009 (SEPLN´09)* , San Sebastian, 2009, pp. 47-55.

[2] Nugroho, Eko, "Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp," 2011.

[3] T. Hermawan, G. Gunawan, and J. Santoso, "Natural Language Grammar Induction of Indonesian Language Corpora Using Genetic Algorithm," in *International Conference on Asian Language Processing (IALP)*, Penang, 2011, pp. 15-18.

[4] Stein, Benno, and Sven Meyer Zu Eissen. "Near similarity search and plagiarism analysis." In *From Data and Information Analysis to Knowledge Engineering*, pp. 430-437. Springer Berlin Heidelberg, 2006.