JURNAL MEDIA INFORMASI TEKNOLOGI
Volume 1, No. 2, Oktober 2024
DOI: https://doi.org/10.69616/mit.v1i2.190

Comparative Analysis Scoring System Auto Essay (simple-o) Based Algoritma Generalized Latent Semantic Analysis (GLSA) Laplacian Eigenmaps Embedding (LEM) and *Hybrid indexing*

Dandun Kusuma Yudha [1] *, Anak Agung Putri Ratna [1]
[1] Universitas Indonesia, Indonesia.
***** Corresponding Author. E-mail: dandun.kusuma@ui.ac.id

Keywords
GLSA;
Hybrid indexing;
LEM;
Pseudocode;
Simple-O.

ABSTRACT
This research discusses a comparison of two algorithms for an automatic essay assessment system (Simple-O), namely generalized latent semantic analysis (GLSA), Laplacian Eigenmaps embedding (LEM) and hybrid indexing. The two algorithms are compared to find out how the two algorithms work, processing speed, and assessment results. Comparison of how it works is done by comparing the pseudocode of each algorithm. The processing speed is calculated to find out a faster algorithm for assessing essays. The GLSA hybrid indexing algorithm is a development of the LEM algorithm. The fundamental difference between the two algorithms is in the treatment of nouns and words other than nouns. This research used a sample of eight questions completed by 48 students (384 data). From the research results, GLSA LEM has a total processing time of 46.51454 seconds, which is faster than GLSA hybrid indexing. Meanwhile, the average processing time for GLSA LEM and GLSA hybrid indexing to assess one answer is 6-6.6 seconds. The assessment results from GLSA LEM and GLSA hybrid indexing have the highest similarity level of 95.83% and the lowest 16.67%. Of the eight questions tested, five of them had a similarity level of more than 83.33%.

PENDAHULUAN

Dewasa ini, seorang pengajar dapat menilai kemampuan siswa di bidang akademis dengan berbagai metode. Metode tersebut dapat berupa tes benar salah, tes pilihan ganda, dan tes esai. Dibandingkan dengan metode lain, tes esai merupakan metode yang baik untuk menilai hasil dari kegiatan belajar mengajar yang dilakukan oleh pengajar dan siswa, karena penulisan esai dapat merepresentasikan kemampuan siswa dalam mengingat, mengorganisasikan, mengekspresikan, dan mengintegrasikan gagasan yang dimiliki oleh siswa tersebut. Akan tetapi sistem penilaian esai menimbulkan masalah baru, yaitu pemeriksaaan esai yang membutuhkan waktu lebih lama jika dibandingkan dengan memeriksa tes pilihan ganda ataupun tes benar salah. Untuk mempermudah proses penilaian (grading) suatu esai, telah banyak dikembangkan aplikasi essay grading secara otomatis yang dapat mempercepat dan mempermudah penilaian esai.

Telah banyak penelitian yang dikembangkan untuk aplikasi essay grading sejak era tahun 1960-an dan ada beberapa algoritma yang diajukan oleh beberapa ilmuwan. Dimulai dari algoritma

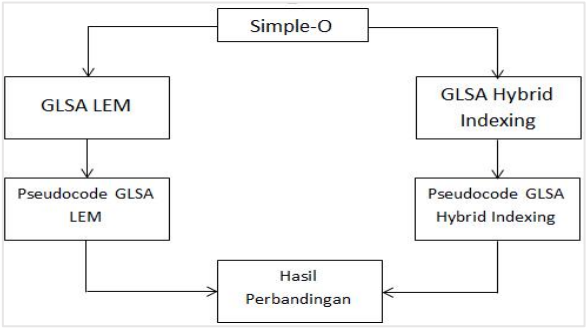
kombinasi teknik perhitungan statistik dan natural language processing (NLP) untuk analisa konten tulisan yang digunakan pada aplikasi essay grading E-RATER[1] dan Project Essay Grade (PEG)[2]. Algoritma bayesian model yang digunakan pada Bayesian Esay Testing System (BETSY)[3]. Algoritma penilaian esai dengan menggunakan latent semantic analysis (LSA)[4] dan pengembangannya generalized latent semantic analysis (GLSA)[5] telah diaplikasikan dalam Sistem Penilaian Esai Otomatis (Simple-O)[6] yang dikembangkan oleh Departemen Teknik Elektro Universitas Indonesia. Pada versi terbaru Simple-O algoritma yang digunakan adalah GLSA lapacian eigenmaps embedding (LEM)[7] dan algoritma tersebut akan dikembangkan lagi menjadi GLSA *hybrid indexing*.

Penelitian ini akan membandingkan cara kerja, waktu proses, dan nilai mahasiswa dari dua metode yang digunakan untuk penilaian essai pada Simple-O, yaitu GLSA lapacian eigenmaps embedding (LEM) dan GLSA *hybrid indexing*. Cara kerja dibandignkan dengan melihat pseudocodenya, sedangkan waktu proses dan nilai dilihat dari hasil keluaran GLSA LEM dan GLSA *Hybrid indexing*. Kedua algoritma tersebut dipilih karena algoritma GLSA LEM merupakan algoritma yang saat ini dipakai di Simple-O dan GLSA *hybrid indexing* merupakan algoritma yang dikembangkan dari GLSA LEM, sehingga diperlukan perbandingan diantara keduanya untuk mengetahui cara kerja, waktu proses, dan nilai mahasiswa dari masing-masing algoritma tersebut.

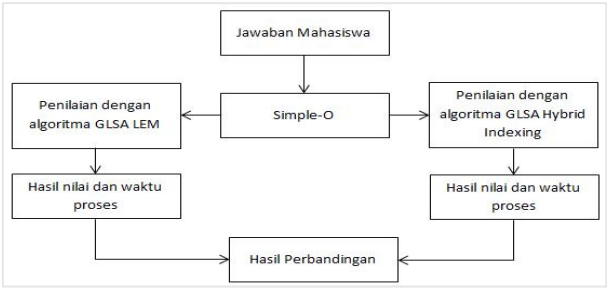
PERANCANGAN SISTEM PENILAIAN ESAI OTOMATIS (SIMPLE-O) DENGAN ALGORITMA GLSA LEM DAN *HYBRID INDEXING*

A. *Skenario Perbandingan*

Sistem penilaian esai otomatis (Simple-O) dengan algoritma GLSA LEM dan GLSA *hybrid indexing* akan dibandingkan dengan menganalisis cara kerja sistemnya, waktu prosesnya, dan hasil keluaran nilai ujiannya. Untuk membandingkan cara kerja sistemnya, akan dilihat dari pseudocode dari masing-masing algoritma penilai esai tersebut seperti yang dapat dilihat pada Gambar 1 Untuk perbandingan waktu proses dan nilainya, akan dilakukan dengan menguji sistem untuk mengoreksi 48 jawaban mahasiswa yang berbeda pada ujian mata kuliah Jaringan Komputer yang berjumlah delapan soal. Masing-masing soal akan dinilai satu kali menggunakan algoritma GLSA LEM dan GLSA *hybrid indexing* untuk membandingkan hasilnya seperti yang dapat dilihat pada Gambar 2.



Gambar 1. Skenario Perbandingan Untuk Membandingkan Pseudocode Algoritma GLSA LEM dan GLSA *Hybrid indexing*



Gambar 2. Skenario Perbandingan Untuk Membandingkan Nilai dan Waktu Pada GLSA LEM dan GLSA *Hybrid indexing*

B. *Kelengkapan Penelitian*

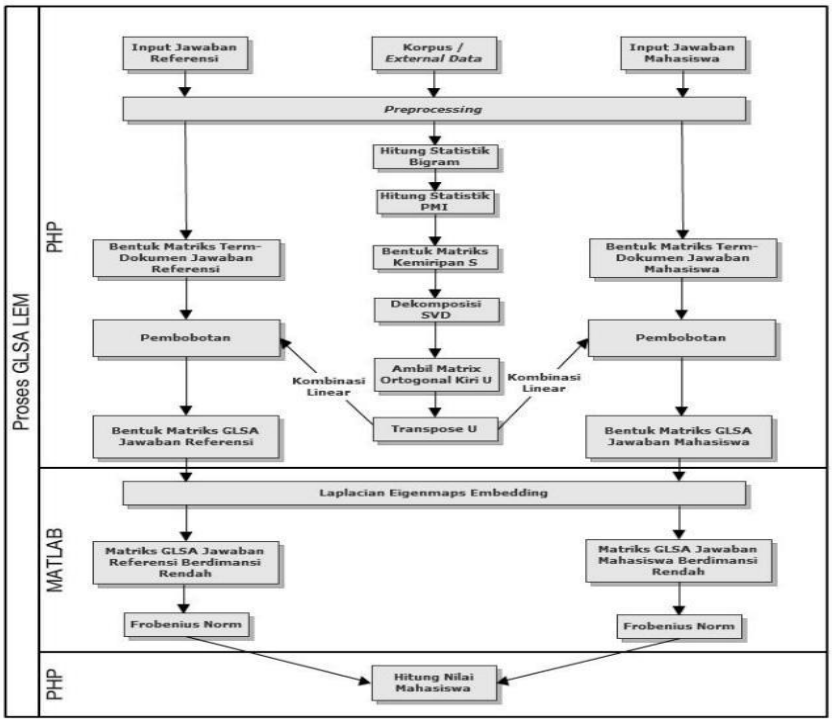
Berikut ini adalah komponen perangkat keras dan perangkat lunak yang digunakan untuk menguji performa aplikasi SIMPLE-O. Perangkat keras yang digunakan dalam perancangan sistem ini sebuah laptop dengan spesifikasi sebagai berikut:

1. Prosesor: AMD A6-4400M APU with Radeon™ HD Graphics (2 CPUs) 2.7GHz
2. RAM: 4096MB RAM
3. Harddisk: 500 GB

Perangkat lunak yang digunakan pada perancangan sistem ini adalah sebagai berikut:

1. Operating system Windows 7
   Sebagai sistem operasi laptop untuk menjalankan semua fungsi dari perangkat keras pada sistem ini.
2. Apache 2.2.14
   Apache merupakan sebuah perangkat lunak yang digunakan untuk menjadi web server.
3. MySql 3.2.4
   MySql digunakan untuk menyimpan dan memanajemen data yang ada pada database.
4. PHP versi 5.3.1
   Bahasa pemrograman yang digunakan untuk membuat sistem yang merupakan aplikasi yang berbasis web.
5. Matlab versi R2012a
   Digunakan untuk menggabungkan matriks, memproses matriks dengan laplacian dan mencari nilai normalisasi frobenius.

C. *Algoritma GLSA LEM*



Gambar 3. Rancangan Simple-O Berbasis GLSA LEM

Gambar 3 adalah rancangan sistem Simple-O pada GLSA LEM. Setiap jawaban baik jawaban referensi ataupun jawaban mahasiswa sebelum proses kalkulasi melalui tahap preprocessing untuk menghilangkan tanda baca dari jawaban referensi dan jawaban mahasiswa. Matriks jawaban referensi dan mahasiswa dibentuk menggunakan nilai frekuensi kemunculan setiap kata yang sudah

diberi pembobotan pada dokumen dengan window sebesar sepuluh. Matriks ortogonal kiri hasil dekomposisi matriks kemiripan PMI kemudian dikombinasi linear dengan matriks jawaban untuk menghasilkan matriks dokumen GLSA. Matriks ini kemudian diproses dengan metode spektral laplacian eigenmaps embedding (LEM) untuk menghasilkan hasil akhir matriks berdimensi rendah dokumen GLSA.

Pada proses kombinasi linear matriks ortogonal kiri hasil dekomposisi matriks kemiripan PMI dengan matriks jawaban digunakan fungsi dari library PHP Java Matrix. Sedangkan proses LEM dilakukan di Matlab. Perhitungan nilai jawaban mahasiswa dilakukan dengan membandingkan nilai normalisasi Frobenius vektor jawaban referensi dengan vektor jawaban mahasiswa. Nilai perbandingan ini kemudian digunakan sebagai nilai mahasiswa. Gambar 4 berikut adalah pseudocode untuk proses Simple-O berbasis GLSA LEM:

```
<?php
$jawaban = "SELECT * from tabel jawaban mahasiswa/referensi";
$katabenda = "SELECT * from tabel kata benda";
    $waktumulai = microtime(true);
    $jawaban = bersihkan dari tanda baca($jawaban);
    $jawaban = masukkan ke dalam array(" ",$jawaban);
    $matrix = array jawaban diubah ke dalam matrix($pilih,$pilihGlobal,$jawaban)
    $nilaifrobenius = hitungnilaifrobenius($matrix,$jawaban);

function hitungnilaifrobenius($matrix,$jawaban){
    $kalimatrixpmi = $matrix->dikalikan($matrixpmi);
    $nilaifrobenius = matlab_exec("a=[$kalimatrixpmi];
        LEM = petakan(a,'Laplacian');
        normalisasifrobenius = normalisasi(LEM,'fro')");
        return $nilaifrobenius;
}

$nilaifinal = $nilaifrobeniusjawabanreferensi/$nilaifrobeniusjawabanmahasiswa;
$waktuselesai=microtime(true);
$lamaproses=$waktuselesai-$waktumulai;
?>
```

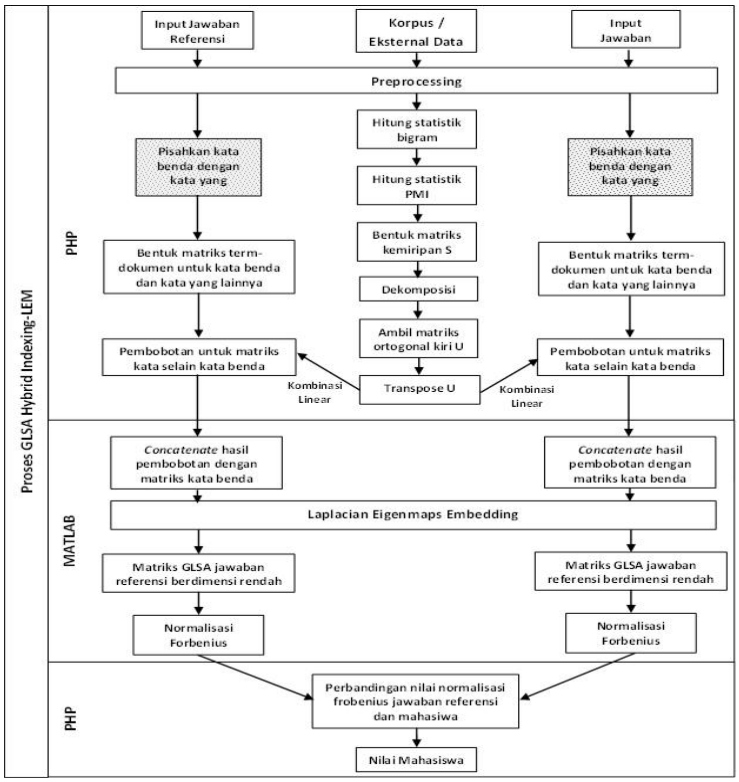
Gambar 4. Pseudocode GLSA LEM

Berikut ini adalah tahapan untuk proses Simple-O berbasis GLSA LEM:

1. Bersihkan jawaban dari karakter non alfabetik
2. Jawaban dibagi menjadi sejumlah dokumen dengan ukuran window 10.
3. Bentuk matriks jawaban
4. Kombinasikan matriks ortogonal kiri hasil dekomposisi matriks PMI dengan matriks jawaban.
5. Terapkan metode spektral laplacian eigenmaps embedding kepada matriks
6. Hitung nilai normalisasi frobenius matriks jawaban.
7. Bandingkan nilai normalisasi frobenius matriks jawaban mahasiswa dan matriks jawaban referensi.
8. Ambil hasil perbandingan sebagai nilai mahasiswa.

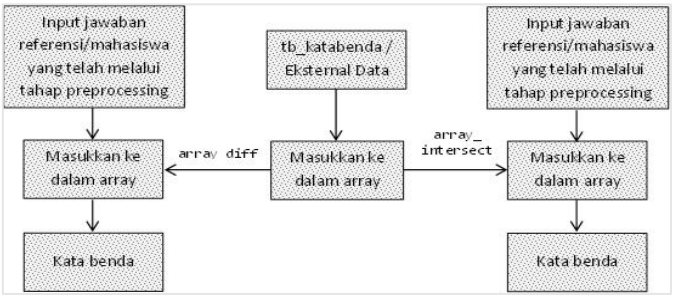
D. *Algoritma GLSA Hybrid indexing*

Gambar 5 adalah rancangan sistem Simple-O pada GLSA *hybrid indexing*. Pada GLSA *hybrid indexing* semua jawaban referensi dan jawaban mahasiswa akan melalui tahap preprocessing. Pada proses preprocessing, jawaban referensi dan jawaban mahasiswa akan diubah menjadi lowercase dengan menggunakan perintah strtolower yang ada pada php. Tujuan dari lowercase disini adalah untuk memudahkan pemisahan kata benda dan kata-kata selain kata benda. Setelah itu tanda baca dan line break akan dhilangkan dari jawaban referensi dan jawaban mahasiswa.



Gambar 5. Rancangan Simple-O Berbasis GLSA *Hybrid indexing*

Setelah tahap preprocessing selesai, jawaban referensi dan jawaban mahasiswa akan dipisahkan menjadi kata benda dan kata selain kata benda. Untuk proses pemisahan kata benda dan kata selain kata benda dapat dilihat pada Gambar 6 berikut.



Gambar 6. Proses Pemisahan Kata Benda dan Kata Selain Kata Benda

Untuk memisahkan kata benda dan kata selain kata benda, diperlukan sebuah tabel yang bernama *tb_katabenda* yang berisi kata benda pada database MySql. Kata benda dan kata selain kata benda pada jawaban mahasiswa atau jawaban referensi yang akan dipisahkan pertama-tama dimasukkan ke dalam *array* . Isi dari *tb_katabenda* juga dimasukkan ke dalam *array* . Lalu isi *array* dari jawaban referensi atau mahasiswa akan dibandingkan dengan isi *array* dari *tb_katabenda* dengan meggunakan perintah *array _diff* dan *array _intersect*, kedua perintah terbebut merupakan perintah yang *case sensitive*, oleh karena itu seluruh kata pada tahap preprocessing diubah ke dalam lowercase. Perintah *array* _diff digunakan untuk memisahkan kata selain kata benda, karena prinsip dari perbandingan *array* _diff jika ada isi *array* yang berbeda diantara *array* jawaban dan *array* dari tabel kata benda maka hasil keluarannya berupa isi *array* yang berbeda tersebut. Sedangkan perintah *array* _intersect digunakan untuk memisahkan kata benda, *array* _intersect merupakan

kebalikan dari perintah *array* _diff, jadi jika ada isi *array* yang sama diantara *array* jawaban dan *array* dari tabel kata benda maka hasil keluarannya berupa isi *array* yang sama tersebut.

tb_katabenda merupakan tabel yang berisi lebih dari 20.000 kata benda yang ada pada Bahasa Indonesia dan ditambah dengan kata-kata yang tidak ada di dalam Bahasa Indonesia namun sering digunakan pada Mata Kuliah Jaringan Komputer. Isi dari *tb_katabenda* didapat dari[20] yang hanya diambil kata bendanya saja

Setelah itu kata benda dan kata selain kata benda akan diubah menjadi matriks dengan menggunakan nilai frekuensi kemunculan setiap kata yang sudah diberi pembobotan pada dokumen dengan window sebesar sepuluh. Untuk menjaga agar besar matriks kata benda dan kata selain benda sama besarnya, jumlah kata jawab unik yang digunakan untuk membuat matriksnya diambil dari jawaban yang kata benda dan kata selain kata bendanya belum dipisahkan. Setelah itu tahap selanjutnya adalah melakukan kombinasi linear dari matriks ortogonal kiri hasil dekomposisi matriks kemiripan PMI dengan matriks kata selain kata benda benda. Setelah didapatkan hasil matriks kombinasi linear dari kata selain kata benda benda, tahap selanjutnya adalah menggabungkan (*concatenate*) secara horizontal matriks kombinasi linear dengan matriks kata benda. Setelah itu matriks tersebut akan diproses dengan metode spektral *laplacian eigenmaps embedding* (LEM) untuk menghasilkan hasil akhir matriks berdimensi rendah dokumen GLSA.

Nilai jawaban mahasiswa didapatkan dengan membandingkan nilai normalisasi frobenius vektor jawaban referensi dengan vektor jawaban mahasiswa. Pada proses perhitungan nilai, input yang digunakan adalah matriks hasil gabungan dari matriks kata benda dan matriks kata selain kata benda yang dipetakan kedalam PMI secara horizontal. Gambar 7 berikut ini adalah *pseudocode* untuk proses Simple-O berbasis GLSA *hybrid indexing*:

```
<?php
$jawaban = "SELECT * from tabel jawaban mahasiswa/referensi";
$katabenda = "SELECT * from tb_katabenda";
    $waktumulai = microtime(true);
    $jawaban = rubah huruf kapital ke lowercase ($jawaban);
    $jawaban = bersihkan dari tanda baca($jawaban);
    $jawaban = masukkan ke dalam array(" ",$jawaban);
    $katabenda = masukkan kata benda ke dalam  array($jawaban,$katabenda);
    $nonkatabenda = masukkan kata selain kata benda ke dalam array ($jawaban,$katabenda);
    $matrixkatabenda = array kata benda diubah ke dalam matrix($katabenda);
    $matrixkatalain = array kata selain kata benda diubah ke dalam matrix ($nonkatabenda);
    $nilaifrobenius = hitungnilaifrobenius($matrixkatabenda,$matrixkatalain,$jawaban);

function hitungnilaifrobenius($matrixkatabenda,$matrixkatalain,$jawaban){
    $kalimatrixpmi = $matrixkatalain->dikalikan($matrixpmi);
    $nilaifrobenius = matlab_exec("b=[$kalimatrixpmi];
        c=[$matrixkatabenda];
        doncatenate=[c,b];
        LEM = petakanconcatenate,'Laplacian');
        normalisasifrobenius = normalisasi(LEM,'fro')");
        return $nilaifrobenius;
}

$nilaifinal = $nilaifrobeniusjawabanreferensi/$nilaifrobeniusjawabanmahasiswa;
$waktuselesai=microtime(true);
$lamaproses=$waktuselesai-$waktumulai;
?>
```

Gambar 7. Pseudocode GLSA *Hybrid indexing*

Berikut ini adalah tahapan untuk proses Simple-O berbasis GLSA *hybrid indexing*:

1. Seluruh input jawaban diubah menjadi lowercase.
2. Bersihkan jawaban dari karakter non alfabetik.
3. Pisahkan kata benda dan kata-kata selain kata benda.
4. Jawaban dibagi menjadi sejumlah dokumen dengan ukuran window 10.
5. Bentuk matriks kata benda dan matriks selain kata benda dari jawaban.
6. Kombinasikan matriks ortogonal kiri hasil dekomposisi matriks PMI dengan matriks kata selain benda.
7. Gabungan (*concatenate)* matriks kata bendan dengan matriks hasil dekomposisi secara mendapatkan matriks jawaban.
8. Terapkan metode spectral Laplacian eigenmaps embedding kepada matriks jawaban.

9. Hitunga nilai normalisasi frobenius matriks jawaban.
10. Bandingkan nila normalisasi frobenius matriks jawaban mahsiswa dan matriks jawaban referensi.
11. Ambil hasil perbandingan sebagai nilai mahasiswa.

Dan tahapan untuk memisahkan kata benda dan kata selain kata benda dipaparkan sebagai berikut:

1. Masukkan isi jawaban mahasiswa/referensi ke dalam *array* .
2. Masukkan isi dari *tb_katabenda* ke dalam *array* .
3. Bandingkan isi *array* dari jawaban mahasiswa/referensi dengan *tb_katabenda* .
4. Jika isi *array* dari jawaban mahasiswa/referensi sama dengan isi *array* dari *tb_katabenda* , kelompokkan ke dalam kata benda.
5. Jika isi *array* dari jawaban mahasiswa/referensi berbeda dengan isi *array* dari *tb_katabenda* , kelompokkan ke dalam kata selain kata benda.

ANALISIS PERBANDINGAN GLSA LEM DAN GLSA *HYBRID INDEXING*

A. *Perbandingan Algoritma*

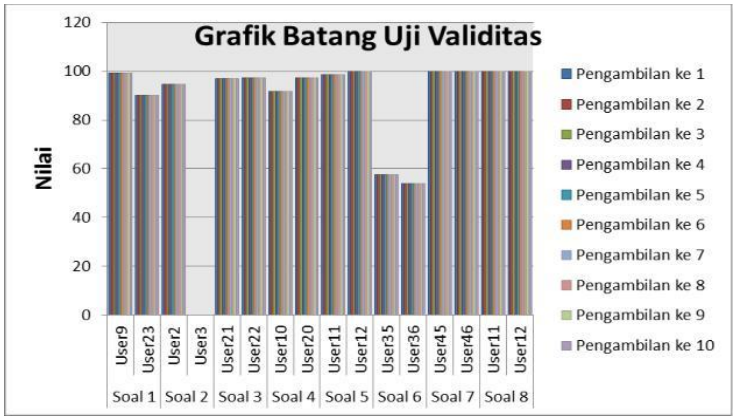
Algoritma GLSA *hybrid indexing* merupakan pengembangan dari GLSA lapacian eigenmaps embedding (LEM)[7], sehingga ada beberapa perbedaan pada cara kerja dari kedua algoritma tersebut. Perbedaan cara kerja kedua algoritma tersebut dapat dilihat dari pseucode kedua algoritma tersebut.

Dilihat dari pseudocode pada Gambar 4 dan Gambar 7, perbedaan dari algoritma GLSA LEM dan GLSA *hybrid indexing* ada pada pengubahan input jawaban menjadi lowercase, pemisahan kata benda dan kata selain kata benda, proses pengalian dengan matriks PMI, dan proses concetenate matriks. Poin-poin berikut ini adalah perbedaan dari algoritma GLSA LEM dan GLSA *hybrid indexing*:

1. Pada algoritma GLSA *hybrid indexing* jawaban diubah menjadi *lowercase* untuk memudahkan pemisahan kata benda dan kata-kata selain kata benda. Perintah yang dipakai untuk memisahkan kedua jenis kata tersebut adalah *array* _diff dan *array* _intersect.
2. Pada GLSA LEM seluruh kata dirubah menjadi matriks dan dikalikan dengan matriks PMI. Sedangkan pada GLSA *hybrid indexing* kata benda dan kata selain kata benda dirubah menjadi matriks dan matriks kata selain kata benda dikalikan dengan matriks PMI.
3. Pada GLSA LEM hasil dari pengalian matriks jawaban dengan PMI akan ditransformasikan dengan metode laplacian. Sedangkan pada GLSA *hybrid indexing* yang ditransformasikan dengan metode laplacian adalah hasil dari gabungan (concetenate) matriks kata benda dengan hasil dari matriks kata lain yang dikalikan dengan PMI.
4. Untuk proses penilaian mahasiswa kedua metode menggunakan cara yang sama, yaitu dengan membandingkan nilai normalisasi frobenius jawaban mahasiswa dan jawaban referensi dari matriks jawaban yang sudah ditransformasikan dengan metode laplacian.

B. *Uji Validitas Algoritma GLSA Hybrid indexing*

Uji validitas diperlukan untuk menguji GLSA *hybrid indexing*, karena GLSA *hybrid indexing* belum pernah diuji coba sebelumnya. Uji validitas dilakukan dengan menilai jawaban dari dua orang user yang dipilih secara acak untuk setiap soal yang dilakukan sebanyak sepuluh kali. Uji validitas ini dilakukan untuk melihat apakah ada perubahan nilai mahasiswa yang dinilai dengan menggunakan Simple-O berbasis GLSA *hybrid indexing*. Hasil data uji validitas dapat dilihat pada Gambar 8 berikut.



Gambar 8. Grafik Batang Uji Validitas GLSA *Hybrid indexing*

Dilihat dari grafik batang pada Gambar 8, nilai uji validitas yang dilakukan untuk jawaban dari dua orang user yang dipilih secara acak untuk setiap soal yang dilakukan sebanyak sepuluh kali tidak terlihat perubahan nilai pada pengujian soal. Hal tersebut membuktikan bahwa algoritma GLSA *hybrid indexing* melakukan penilaian secara konsisten dan tidak berubah-ubah untuk masing-masing user.

C. *Perbandingan Waktu Proses*

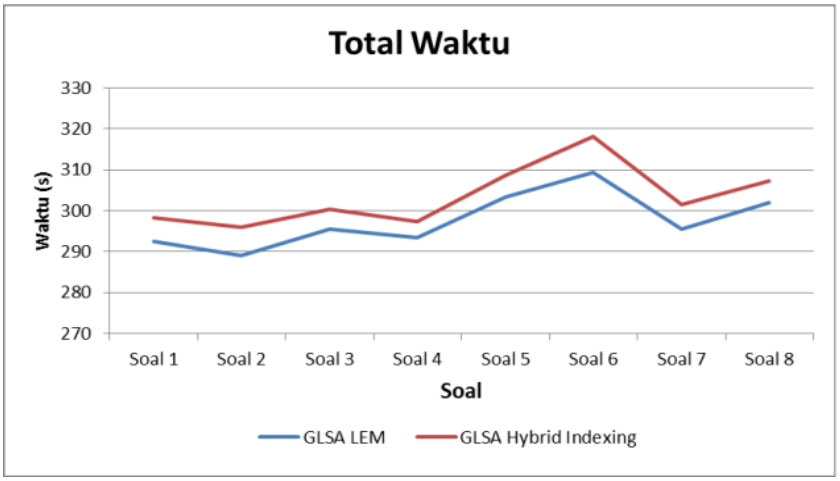
Perbandingan waktu proses dilakukan untuk mengetahui total waktu proses dan rata-rata waktu proses algoritma GLSA LEM dan GLSA *hybrid indexing*. Dari total waktu dan rata-rata waktu proses, dapat dilihat algoritma yang lebih cepat dalam memproses jawaban mahasiswa.

1. *Perbandingan Total Waktu Proses*

Pada Tabel 1 dan Gambar 9 dapat dilihat bahwa Simple-O berbasis GLSA LEM memiliki total waktu pengolahan yang lebih cepat 46.51454 detik daripada GLSA *hybrid indexing*. Total waktu proses dari GLSA LEM adalah 2380.847 detik, sedangkan GLSA *hybrid indexing* memiliki total waktu proses selama 2427.362 detik. Total waktu proses tercepat ada pada soal 2 yang dinilai dengan menggunakan GLSA LEM dengan total waktu proses 289.045 detik. Untuk total waktu proses terlama ada pada soal 6 yang dinilai dengan menggunakan GLSA *hybrid indexing* dengan waktu proses 318.0578 detik. Proses GLSA *hybrid indexing* memiliki waktu total yang sedikit lebih lama daripada GLSA LEM karena GLSA *hybrid indexing* memiliki proses yang lebih banyak daripada GLSA LEM. Proses tambahan pada GLSA *hybrid indexing* diantaranya adalah pemisahan kata benda dan kata selain kata benda dan concenate (penggabungan) matriks kata benda dan kata selain kata benda.

Tabel 1. Tabel Perbandingan Total Waktu Proses GLSA LEM dan GLSA *Hybrid indexing*

Total	**GLSA LEM (*s*)**	**GLSA *Hybrid indexing* (*s*)**
Soal 1	292.5878	298.2588
Soal 2	289.045	295.9339
Soal 3	295.5866	300.2676
Soal 4	293.4086	297.3923
Soal 5	303.4502	308.6752
Soal 6	309.3181	318.0578
Soal 7	295.4584	301.558
Soal 8	301.9927	307.2183
Total	2380.847	2427.362



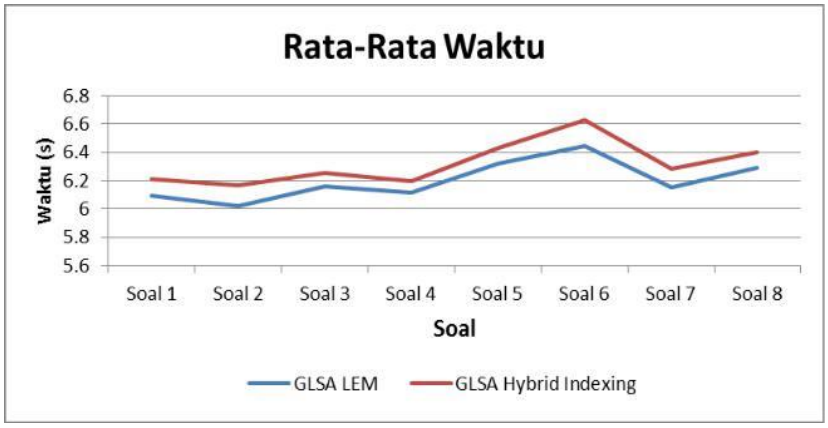
Gambar 9. Grafik Perbandingan Waktu Total GLSA LEM dan GLSA *Hybrid indexing*

2. *Perbandingan Rata-Rata Waktu Proses*

Pada Tabel 2 dan Gambar 10 dapat dilihat bahwa rata- rata waktu proses Simple-O berbasis GLSA LEM adalah 6-6.4 detik sementara GLSA *hybrid indexing* 6.1-6.6 detik. Perbedaan rata-rata waktu proses dari GLSA LEM dan GLSA *hybrid indexing* kurang dari 0.2 detik. Perbedaan rata-rata waktu proses terkecil ada pada soal 4 dengan Δ sebesar 0.08299 detik. Sementara perbedaan rata-rata waktu proses terbesar ada pada soal 6 dengan Δ sebesar 0.18208.

Tabel 2. Tabel Perbandingan Rata-Rata Waktu Proses GLSA LEM dan GLSA *Hybrid indexing*

Rata-rata Waktu	GLSA LEM (*s*)	GLSA Hybrid Indexing (*s*)	Δ Rata-rata Waktu
Soal 1	6.09558	6.213725	0.11815
Soal 2	6.021771	6.16529	0.14352
Soal 3	6.158053	6.255575	0.09752
Soal 4	6.112679	6.195672	0.08299
Soal 5	6.32188	6.430734	0.10885
Soal 6	6.444127	6.626205	0.18208
Soal 7	6.155384	6.282459	0.12707
Soal 8	6.291515	6.400382	0.10887



Gambar 10. Grafik Waktu Proses GLSA LEM dan GLSA *Hybrid indexing*

D. *Perbandingan Nilai*

Pada Tabel 3 dapat dilihat frekuensi kemunculan nilai yang selisihnya kurang dari 10 dan lebih dari 10 dari GLSA LEM dan GLSA *hybrid indexing*. Nilai dari algoritma GLSA LEM dan GLSA *hybrid indexing* dapat dikatakan mirip jika memiliki selisih nilai kurang dari 10. Seperti pada Tabel 4.4 soal 4 memiliki tingkat kemiripan nilai GLSA LEM dan GLSA *hybrid indexing* tertinggi (95,83%) dari semua soal yang ada dan soal 6 memiliki tingkat kemiripan terendah (16,67%). Pada soal 1,3, 4, 5, dan 8 algoritma GLSA LEM dan GLSA *hybrid indexing* memiliki perbedaan nilai yang kecil, karena memiliki tingkat kemiripan lebih dari 83,33%. Sedangkan pada soal 2, 6, dan 7 algoritma GLSA LEM dan GLSA *hybrid indexing* memiliki tingkat kemiripan kurang dari 83,33%.

Tabel 3. Tabel Frekuensi ΔNilai GLSA LEM dan *Hybrid indexing*

Soal	**Frekuensi ΔNilai**	
	0-9,9	10 - 100
Soal 1	40	8
Soal 2	37	11
Soal 3	40	8
Soal 4	46	2
Soal 5	43	5
Soal 6	8	40
Soal 7	39	9
Soal 8	42	6

Perbedaan nilai tersebut disebabkan oleh perbedaan proses yang dilakukan oleh algoritma GLSA LEM dan GLSA *hybrid indexing*. GLSA *hybrid indexing* melakukan proses pemisahan kata benda dan selain kata benda dan memberi perlakuan yang berbeda antara kata benda dan kata selain kata benda, sedangkan GLSA LEM semua kata diperlakukan sama.

SIMPULAN

Kesimpulan dari penelitian ini menunjukkan adanya perbedaan utama antara metode GLSA LEM dan GLSA hybrid indexing dalam perlakuan terhadap kata benda dan kata selain kata benda. Pada GLSA LEM, seluruh kata diubah menjadi matriks yang kemudian dikalikan dengan matriks PMI, sedangkan GLSA hybrid indexing hanya mengalikan matriks kata selain kata benda dengan PMI. Selain itu, perhitungan nilai akhir pada kedua metode ini dilakukan dengan membandingkan hasil normalisasi Frobenius antara jawaban dan jawaban referensi. Perbedaannya, pada GLSA LEM, nilai Frobenius diperoleh dari transformasi laplacian matriks jawaban yang sudah dikalikan dengan PMI, sedangkan pada GLSA hybrid indexing, transformasi laplacian diterapkan pada hasil gabungan matriks kata benda dan matriks kata lainnya yang sudah dikalikan PMI. Hasil uji validitas menggunakan Simple-O berbasis algoritma GLSA hybrid indexing menunjukkan penilaian konsisten tanpa perubahan nilai pada jawaban dari dua orang pengguna yang dipilih secara acak, dengan pengujian dilakukan sebanyak sepuluh kali. Dari segi waktu proses, GLSA LEM menunjukkan efisiensi yang lebih tinggi dengan waktu total 46.51454 detik lebih cepat dibandingkan GLSA hybrid indexing. Untuk menilai delapan soal dari 48 mahasiswa, GLSA LEM membutuhkan waktu 2380.847 detik, sementara GLSA hybrid indexing membutuhkan 2427.362 detik, dengan rata-rata waktu per soal masing-masing 6-6.4 detik dan 6.1-6.6 detik. Kedua metode ini menunjukkan tingkat kemiripan hasil yang cukup tinggi, dengan nilai kemiripan tertinggi mencapai 95,83% dan terendah 16,67%. Dari delapan soal yang diuji, lima di antaranya memiliki tingkat kemiripan di atas 83,33%, yang menunjukkan akurasi metode yang baik dalam mengevaluasi jawaban.

DAFTAR PUSTAKA

[1] J. Burstein. 2003. The e-rater scoring engine: Automated essay scoring with natural language processing. In M. D. Shermis and J. Burstein, editors, Automated essay scoring: A cross-disciplinary perspective. Lawrence Erlbaum Associates, Hillsdale, NJ.

[2] E. B. Page. 1966. The imminence of grading essays by computer. Phi Delta Kappan, 47:238–243.

[3] M. Rudner, Lawrence & Liang, Tahung. 2002. Automated Essay Scoring Using Bayes' Theorem. The Journal of Technology, Learning, and Assessment, volume 1, number 2.

[4] Artajaya, Henry. (2012). ―Analis Algoritma pada Sistem Essay Grading SIMPLE-O‖. Skripsi. Depok: Universitas Indonesia.

[5] Abdy, T.M. Rikza. (2013). ―Analisa Kinerja Algoritma Generalized Latent Semantic Analysis (GLSA) Dengan Proses Stemming Menggunakan Persamaan Kata Pada Sistem Penilaian Otomatis Simple-O. Skripsi. Depok: Universitas Indonesia.

[6] Ratna, A. A. P., Budiardjo, B., Hartanto. D. (2007); SIMPLE: Sistem Penilai Esei Otomatis untuk Menilai Ujian dalam Bahasa Indonesia; Makara, Teknologi, Vol. 11, No. 1,

[7] Artajaya, Henry. (2013). Analisa Unjuk Sistem Penilai Esai Otomatis Berbasis Algoritma Generalized Semantic Analysis (GLSA) Menggunakan Metode Spektral Laplacian Eigenmaps Embedding. Tesis. Depok: Universitas Indonesia.

[8] P. W. Foltz, D. Laham, and T. K. Landauer. 1999. Automated Essay Scoring: Applications to Educational Technology. In Proc. of World Conf. Educational Multimedia, Hypermedia & Telecommunications, Seattle, USA.

[9] Christie, J. R., 1999, Automated essay marking-for both style and content. Proceedings of the Third Annual Computer Assisted Assessment Conference, Loughborough University, Loughborough, UK.

[10] Ming, P.Y., Mikhailov, A.A., and Kuan, T.L., 2000, Intelligent essay marking system. Learners Together, NgccANN Polytechnic, Singapore.

[11] Mitchell, T., Russel, T., Broomhead, P., and Aldridge N. 2002. Towards robust computerized marking of free-text responses. Proceedings of the Sixth International Computer Assisted Assessment Conference, Loughborough University, Loughborough, UK,

[12] I. Matveeva, ―Generalized latent semantic analysis for document representation,‖ ProQuest Dissertations and,Theses. Chicago: Proquest LLC, 2008.

[13] Olney, Andrew M. ―Generalizing Latent Semantic Analysis‖. 2009 IEEE International Conference on Semantic Computing.University of Memphis. Memphis, USA.

[14] D. Widdows, ―Unsupervised methods for developing taxonomies by combining syntactic and statistical information,‖ North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2003.

[15] E. L. Terra and C. L. Clarke, ―Frequency estimates for statistical word similarity measures,‖ North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2003.

[16] T. F. Cox and M. A. Coxx, Multidimensional Scaling, CRC/Chapman and Hall, 2001.

[17] M. Belkin and P. Niyogi, ―Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation, vol. 15, no. 6, pp. 1373-1396, 2003.

[18] C. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, Cambridge, MA: MIT Press, 1999.

[19] F. Choi, P. Wiemer-Hastings, and J. Moore, ―Latent semantic analysis for text segmentation,‖ in Proc. Of Empirical Methods on Natural Language Processing, pp. 109-117, 2001.
[20] Database Kata Dasar Bahasa Indonesia. 2010. http://www.bahtera.org 8 Oktober 2013