WEB SERVICE DISCOVERY MENGGUNAKAN *COSINE SIMILARITY* UNTUK MENINGKATKAN AKURASI *QUERY* PADA *WEB SERVICE REPOSITORY*

SKRIPSI

Oleh :
RATIH MAYLLIA DEWI
NIM. 13650073

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK BRAHIM
MALANG
2018

WEB SERVICE DISCOVERY MENGGUNAKAN *COSINE SIMILARITY* UNTUK MENINGKATKAN AKURASI *QUERY* PADA *WEB SERVICE REPOSITORY*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
RATIH MAYLLIA DEWI
NIM. 13650073

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2018

LEMBAR PERSETUJUAN

WEB SERVICE DISCOVERY* MENGGUNAKAN *COSINE SIMILARITY
UNTUK MENINGKATKAN AKURASI *QUERY* PADA
WEB SERVICE REPOSITORY

SKRIPSI

Oleh :
Ratih Mayllia Dewi
NIM. 13650073

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal: 08 Mei 2018

Pembimbing I,	Pembimbing II,
M. Ainul Yaqin, M.Kom	M. Imamudin, Lc., MA
NIP. 19761013 200604 1 004	NIP. 19740602 200901 1 010

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

LEMBAR PENGESAHAN

WEB SERVICE DISCOVERY* MENGGUNAKAN *COSINE SIMILARITY
UNTUK MENINGKATKAN AKURASI *QUERY* PADA
WEB SERVICE REPOSITORY

SKRIPSI

Oleh :
Ratih Mayllia Dewi
NIM. 13650073

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: Juni 2018

Susunan Dewan Penguji		Tanda Tangan
Penguji Utama	: Syahiduz Zaman, M.Kom NIP. 19700502 200501 1 005	( )
Ketua Penguji	: Supriyono, M.Kom NIP. 19841010 20160801 1 078	( )
Sekertaris Penguji	: M. Ainul Yaqin, M.Kom NIP. 19761013 200604 1 004	( )
Anggota Penguji	: M. Imamudin, Lc., MA NIP. 19740602 200901 1 010	( )

Mengesahkan,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama : Ratih Mayllia Dewi

NIM : 13650073

Fakultas/Jurusan : Sains dan Teknologi/ Teknik Informatika

Judul Skripsi : ***WEB SERVICE DISCOVERY*** **MENGGUNAKAN** ***COSINE SIMILARITY*** **UNTUK MENINGKATKAN AKURASI** ***QUERY*** **PADA** ***WEB SERVICE REPOSITORY***

Menyatakan dengan sebenarnya bahwa hasil penelitian saya ini tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.
Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur penjiplakan, maka saya bersedia untuk mempertanggungjawabkan, serta diproses sesuai peraturan yang berlaku.

Malang, 04 Juni 2018
Yang membuat pernyataan

Ratih Mayllia Dewi
NIM. 13650073

HALAMAN PERSEMBAHAN

Syukur *Alhamdulillah* puji syukur kehadirat Allah SWT atas karunia kehidupan dan diberikannya keluasan ilmu pengetahuan serta ridlo-Nya sehingga diberikan kekuatan menempuh jenjang pendidikan S1 Teknik Informatika di kampus UIN Maliki Malang tercinta. Sholawat serta salam kepada Nabi Muhammad SAW, yang syafaatnya diharapkan di hari akhir. Saya persembahkan karya penelitian ini untuk keluarga yang saya sayangi:

Ayah saya, H. Hari Purnomo (Alm.), sosok dermawan penuh inspirasi yang senantiasa bekerja keras, bersyukur dan telah mengajarkan saya agar tidak mudah berputus asa serta berlapang dada dan bagaimana menghargai orang lain. Terima kasih telah memberikan perlindungan kepada saya hingga 18 tahun lamanya.

Ibu saya, Tutut Kadarningsih, terima kasih banyak atas dukungan, kasih sayang, nasehat luar biasa, doa dan pengalaman hidup yang telah dibagi dengan saya. Sehingga saya dapat memutuskan pilihan yang tepat di skenario kehidupan yang sedang menanti saya.

Kakak saya, Nova Candra NT dan Happy Indra Astuti (Alm.) serta ipar dan keponakan, terima kasih atas doa dan nasehat tiada hentinya.

Sahabat saya, Ica, Izza, Baiti, dan Ratna, terima kasih atas perjuangan yang telah kita lalui bersama.

Sahabat tetangga kos tercinta, Fian, Ilma, Peon, Imamah, Neni, Anggra, dan Qolbi, terima kasih atas bantuan dan doanya.

Abang saya, Dayu Mr., sumber inspirasi tentang arti hidup sesungguhnya dan terima kasih atas motivasi hebat demi kehidupan yang lebih baik. Semoga dari sini bisa

Sahabat jauh disana, Ryan, Yani dan Cindy, terima kasih atas waktu, tenaga dan pikiran yang telah kita bagi bersama. Banyak mengajarkan arti kedewasaan.

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Segala puji bagi Allah SWT tuhan semesta alam, karena atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan skripsi dengan baik dan lancar. Shalawat serta salam selalu tercurah kepada tauladan terbaik Nabi Muhammad SAW yang telah membimbing umatnya dari zaman kebodohan menuju islam yang rahmatan lil alamiin.

Proses penyelesaian skripsi ini banyak pihak yang telah memberikan bantuan baik secara moril, nasihat dan semangat maupun materiil. Atas segala bantuan yang telah diberikan, penulis ingin menyampaikan doa dan ucapan terimakasih yang sedalam-dalamnya kepada:

1. Prof. Dr. Abdul Haris, M.Ag, selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. M. Ainul Yaqin, M.Kom, selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, memotivasi, dan mengarahkan dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.
5. M. Imamudin, Lc., MA, selaku dosen pembimbing II yang senantiasa memberi masukan dan nasihat serta petunjuk dalam penyusunan skripsi ini.
6. Syahiduz Zaman, M.Kom, selaku dosen penguji I yang telah meluangkan waktu untuk menguji dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.
7. Supriyono, M.Kom, selaku dosen penguji II yang telah meluangkan waktu untuk menguji dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.

8. Seluruh dosen dan staff jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
9. Ayah dan ibu, serta seluruh keluarga besar yang senantiasa memberikan do'a dan motivasi kapada penulis dalam menuntut ilmu serta do'a yang senantiasa mengiringi setiap langkah penulis.
10. Teman yang telah terlibat dan dan memberi bantuan dalam menyelesaikan penulisan Fian, Tya, Ayom, Candra, Wanna dan Tim Skripsi Sukses
11. Teman-teman seperjuangan Teknik Informatika angkatan 2013.
12. Semua pihak yang ikut dalam berkontribusi dalam membantu menyelesaikan skripsi.

Adapun kekurangan dan kesalahan mungkin pembaca temukan dalam penulisan skripsi ini, untuk itu penulis menerima segala kritik dan saran yang membangun dari pembaca sekalian. Semoga apa yang menjadi kekurangan bisa dikembangkan dan disempurnakan oleh peneliti berikutnya dan semoga karya ini senantiasa dapat memberi manfaat. *Amin Ya Rabbal A'lamin*.

Wassalamualaikum Wr.Wb

Malang, 31 Mei 2018

Penulis

MOTTO

"Jika Hidup Menggempurmu Jatuh, Maka Ada 2 Pilihan

Bangkit! atau Tetap Tersungkur"

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Dewi, Ratih Mayllia. 2018. *Web Service Discovery* Menggunakan *Cosine Similarity* untuk Meningkatkan Akurasi *Query* pada *Web Service Repository*. Skripsi. Jurusan Teknik Infomatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing: (I) M. Ainul Yaqin, M.Kom, (II) M. Imamudin, Lc., MA

Kata kunci: *Web Service, Web Service Discovery, Vector Space Matching, Cosine Similarity*.

Web service telah menjadi acuan untuk pengembangan *software systems* terdistribusi. Meningkatnya jumlah *web service* menjadi permasalahan dalam menemukan *web service* yang diinginkan dari besarnya jumlah *web service* yang ada. Oleh karena itu, menemukan *web service* yang tepat sesuai dengan kebutuhan menjadi tantangan, atau bisa disebut dengan *web service discovery*.

Teori *Vector space matching* (VSM) memberikan solusi untuk menghitung kemiripan dengan mendefinisikan sebuah *vector* yang mempresentasikan tiap dokumen *web service*, dan sebuah *vector* yang mempresentasikan *query* menggunakan metode *cosine similarity*. Jadi, untuk membangun *web service discovery* diusulkan menggunakan metode *cosine similarity* untuk memanajemen *web service* di dalam *web service repository*.

Berdasarkan pengujian dan analisa yang telah dilakukan terhadap 20 *query* pengujian dengan membandingkan antara SQL *query* dan *cosine similarity*, diperoleh peningkatan akurasi *query* sebesar sebesar 7.01% dari 90.18% menjadi 97.20%. Jadi, tingkat akurasi *query* yang dihasilkan lebih relevan menggunakan *cosine similarity* daripada menggunakan SQL *query*. Karena secara waktu komputasi fungsi *cosine* memiliki kinerja lebih cepat daripada SQL *query*.

ABSTRACT

Dewi, Ratih Mayllia. 2018. ***Web Service Discovery Using Cosine Similarity to Improve Query Accuracy in Web Service Repository***. Thesis. Department of Informatic Engineering. Faculty of Science and Technology. State Islamic University of Maulana Malik Ibrahim Malang.

Adviser: (I) M. Ainul Yaqin, M.Kom, (II) M. Imamudin, Lc., MA

Keywords: *Web Service, Web Service Discovery, Vector Space Matching, Cosine Similarity*.

Web services have become a reference for the development of distributed software systems. Increasing the number of web services becomes a problem in finding the desired web service from the large number of current web services. Therefore, finding the right web service in accordance with the needs of a challenge, or can be called the *web service discovery*.

Vector space matching (VSM) theory provides a solution to calculate the similarity by defining a vector that presents each web service document, and a vector that presents a query using cosine similarity method. So, to build web service discovery is proposed using cosine similarity method to manage web service in web service repository.

Based on testing and analysis of 20 query test by comparing SQL query and cosine similarity, obtained query accuracy increase equal to 7.01% from 90.18% to 97.20%. Thus, the resulting query accuracy level is more relevant using cosine similarity than using SQL queries. Because the computation time of cosine function has performance faster than SQL query.

الـ بحث ملخص

ديوي ، راتيه مايليا. 2018. اكتشاف خدمة الويب باستخدام التشابه لجيب التمام (*Cosine Similarity*) لتحسين دقة الاستعلام في مستودع خدمة الويب (*Web Service Repository*). البحث الجامعي. قسم المعلوماتية، كلية العلوم والتكنولوجيا الجامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج.

المشرف: محمد عين اليقين، الماجستير، وإمام الدين، الماجستير

الكلمات الرئيسية: خدمة الويب، اكتشاف خدمة الويب، مطابقة الفضاء المتجه (*Vector Space Matching*) ، تشابه لجيب التمام.

أصبحت خدمة الويب مرجعًا لتطوير أنظمة البرامج. تصبح زيادة عدد خدمة الويب إلى مشكلة في العثور على خدمة الويب المطلوبة من كبير الخدمات الويب المتاحة. لذلك ، لاكتشاف على خدمة الويب المناسبة وفقا لاحتياجات التحدي يمكن أن يسمى اكتشاف خدمة الويب

توفر نظرية مطابقة الفضاء المتجه (VSM) حلاً لحساب التشابه بتعريف المتجه الذي يقدم كل مستند خدمة ويب ، والمتجه الذي يقدم استعلامًا باستخدام طريقة تشابه لجيب التمام. لذلك ، لإنشاء اكتشاف خدمة الويب يقترح باستخدام طريقة التشابه لجيب التمام لإدارة خدمة الويب في مستودع خدمة الويب.

بناءً على اختبار وتحليل الذى أجرى على 20 استعلامات لاختبار بمقارنة استعلام لغة الاستعلام الهيكلية (SQL) وتشابه لجيب التمام ، حصلت على زيادة دقة الاستعلام بنسبة 7.01٪ من 90.18٪ إلى 97.20٪. وبالتالي ، مستوى دقة الاستعلام هو أكثر صلة باستخدام تشابه الجيب من استخدام استعلام لغة الاستعلام الهيكلية. لأن وقت حساب وظيفة لجيب التمام له الأداء الأسرع من استعلام لغة الاستعلام الهيكلية

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Web service telah menjadi acuan untuk pengembangan *software systems* terdistribusi. *Web services* mengelola *software* yang dapat *described, discovered* dan dipanggil dalam platform independen melalui jaringan (misalnya, intranet, extranet dan internet). Meskipun memiliki kelebihan arsitektur seperti itu, representasi *web service* pada praktek industri saat ini masih kurang, *fundamental semantik* diperlukan untuk memenuhi tujuan dari s*emantic web* (Bell, 2005).

Evaluasi web semantik telah memotivasi para peneliti untuk memperkaya *web service* dengan informasi, yang diinterpretasikan oleh mesin. Sehingga dapat melakukan *web service discovery*, *composition*, dan lain-lain. Tujuan utama dari teknologi semantik *web service* adalah untuk mengotomatisasi *web service discovery* dan meminimalkan proses *discovery* manual (R.K, 2016).

WSDL merupakan bahasa berbasis XML yang digunakan untuk mendefinisikan *web service* dan menggambarkan bagaimana cara untuk mengakses *web service* tersebut. WSDL menjelaskan *method-method* apa saja yang tersedia dalam suatu *web services*, *parameter* apa saja yang diperlukan untuk memanggil suatu *method*, dan apa hasil atau tipe data yang dikembalikan oleh *method* yang akan dipanggil tersebut (http://kemenpora.go.id).

Proses *decompose* dapat didefinisikan sebagai sekelompok elemen proses yang terhubung dan memiliki potensi *reusability*. Fitur penting proses *decompose* adalah melakukan *break down* terhadap model proses bisnis kompleks menjadi

lebih kecil (Ma & Leymann, 2008). Pada penelitian ini, *output* dari proses *decompose* akan digunakan sebagai *input* untuk menemukan *web service*.

Jumlah *web service* yang semakin meningkat, sehingga dibutuhkan solusi yang sistematis untuk penanganan *web service*. *Web service* harus disimpan ke dalam *repository* proses untuk digunakan kembali nantinya (Ma & Leymann, 2008). Selanjutnya, *web service* yakni WSDL *atomic* dapat digunakan sebagai *results* dari *query* untuk mencari *web service* tertentu dalam *repository* besar. Misalnya, sebuah *web service* yang digunakan dalam versi *web service* x dapat digunakan kembali dalam versi baru y dari *web service* lain.

Pencarian dengan jaringan *semantic* berarti pencarian dokumen berdasarkan kata kunci penelusuran dan makna yang terkait dengan kata kunci tersebut. Pencarian semantik berusaha untuk meningkatkan akurasi pencarian dengan memahami maksud pencari dan makna kontekstual istilah seperti yang ditampilkan dalam data pencarian. Dikutip penulis dari penelitian Rahutomo (2009), tujuan pencarian *semantic* adalah mencari konten yang sesuai dengan konteks yang diinginkan pengguna. Jadi, pencarian *semantic* memberikan saran bagi pengguna berdasarkan penarikan kesimpulan yang dilakukan oleh sistem berdasarkan batasan-batasan tertentu (Sarno, Yeni, & Fitri, 2012). Metode yang digunakan untuk pencarian semantik dalam menemukan *web service* beragam, salah satunya adalah *vector space matching*.

Vector space matching menghitung kemiripan dengan mendefinisikan sebuah *vector* yang mempresentasikan tiap dokumen, dan sebuah *vector* yang mempresentasikan *query*. Jika kata-kata dalam dokumen dapat direpresentasikan dalam *vector*, sangat dimungkinkan untuk membandingkan dokumen dengan

query untuk menunjukkan kemirapan konten keduanya. Jika sebuah *query* dianggap mirip dengan sebuah dokumen, koefesien kemiripan yang mengukur kemiripan antara dokumen dan *query* dapat dihitung. Dokumen yang kontennya paling mirip dengan konten pada *query* dianggap paling relevan.

Alternatif dari fungsi jarak (Hamzah, 2008) adalah fungsi similaritas, antara lain *jaccard*, dice, *cosine*, dan *pearson*. Hasil penelitian menunjukkan bahwa hasil *clustering* terbaik adalah jika menggunakan fungsi *cosine*, sementara terburuk menggunakan fungsi jarak *eucledian*. Jadi, *cosine similarity* dapat diusulkan pada penelitian *web service discovery* karena secara waktu komputasi fungsi *cosine* juga memiliki kinerja tercepat sedangkan terjelek adalah *pearson*.

Di luar *information retrieval*, penerapan *Recall*, *Precision* dan F-*measure* dianggap cacat karena mengabaikan sel negatif tabel kontinjensi yang sebenarnya, dan mudah dimanipulasi dengan *biasing the preditcions*. Masalah ini (Powers, 2011) dapat dipecahkan dengan menggunakan akurasi. Untuk mengevaluasi akurasi *query* yang diusulkan, maka akan digunakan pengujian sistem menggunakan ukuran akurasi.

Berdasarkan latar belakang yang telah diuraikan di atas, meningkatnya jumlah *web service* menjadi permasalahan dalam menemukan *web service* yang diinginkan dari besarnya jumlah *web service* yang ada. Oleh karena itu, menemukan *web service* yang tepat sesuai dengan kebutuhan menjadi tantangan, atau bisa disebut dengan *web service discovery*. Penelitian *web services discovery* diusulkan menggunakan teori *vector space matching* untuk memanajemen *web services* yang telah di-*decompose* untuk dapat digunakan kembali (*reuse*).

Pembahasan latar belakang ini menjelaskan bahwa menemukan *web services* dari banyaknya *web service* membutuhkan suatu media *discovery* yang berfungsi untuk menghemat waktu dan tenaga serta memanfaatkan *web service* yang telah ada untuk *composing* tanpa membuat dari nol. Sebagaimana yang disebutkan dalam ayat Al Quran yang menerangkan tentang efisiensi waktu yaitu pada Al Quran Surat Al Furqan Ayat 62:

وَهُوَ الَّذِي جَعَلَ اللَّيْلَ وَالنَّهَارَ خِلْفَةً لِمَنْ أَرَادَ أَنْ يَذَّكَّرَ أَوْ أَرَادَ شُكُورًا (62)

Artinya:

"Dan Dia (pula) yang menjadikan malam dan siang silih berganti bagi orang yang ingin mengambil pelajaran atau orang yang ingin bersyukur" (QS. Al-Furqan : 62).

Islam mendorong seseorang untuk menggunakan waktu dengan baik, agar orang tersebut bisa mengambil pelajaran dan bersyukur atas nikmat waktu yang Allah anugerahkan kepadanya. Yaitu dengan perputaran waktu, maka manusia dapat mengambil pelajaran yang sangat penting mengenai tujuan penciptaannya, yaitu beribadah kepada Allah Subhana Wa Ta'ala serta menjalankan Syariat-Nya, mengingat ajal yang pasti akan menjemputnya, dan mempersiapkan bekal bagi kehidupan di akhiratnya yang kekal dan abadi.

Menunjuk pada surat Al-Furqan ayat 62 berkaitan dengan efisiensi waktu. Pentingnya media penampungan *web service* yaitu pada *web service repository*, yang akan digunakan untuk melakukan *web service discovery* sehingga dapat menghemat waktu pengembangan *web service* sesuai dengan kandungan surat Al-Furqan ayat 62 diatas.

Pada Surat Al-Ashr juga telah disebutkan tentang pentingnya memanfaatkan waktu yaitu Ayat 1-3:

وَالْعَصْرِ (1) إِنَّ الْإِنْسَانَ لَفِي خُسْرٍ (2) إِلَّا الَّذِينَ آمَنُوا وَعَمِلُوا الصَّالِحَاتِ وَتَوَاصَوْا بِالْحَقِّ وَتَوَاصَوْا بِالصَّبْرِ(3)

Artinya:

"*Demi masa. Sesungguhnya manusia itu benar-benar dalam kerugian, kecuali orang-orang yang beriman dan mengerjakan amal saleh dan nasehat menasehati supaya mentaati kebenaran dan nasehat menasehati supaya menetapi kesabaran*".

Tafsir dari Prof. Quraish Shihab mengenai Surat Al Ashr Ayat 1 yakni Allah Subhana WaTa'ala bersumpah demi masa karena masa mengandung banyak keajaiban dan pelajaran yang menunjukkan Kemahakuasaan dan Kemahabijaksanaan-Nya bahwa manusia tidak akan lepas dari kekurangan dalam perlakuan dan keadaannya, kecuali orang-orang yang benar-benar beriman yang mengerjakan amal saleh, saling menasihati sesama mereka untuk berpegang teguh dalam kebenaran yang mengandung semua kebaikan, dan saling menasihati untuk bersabar dalam melaksanakan apa yang diperintahkan kepada mereka dan dalam menjauhi segala larangan. Aku bersumpah demi masa karena mengandung banyak peristiwa dan pelajaran.

Ayat ke-2, memaparkan bahwa semua manusia berada dalam kerugian karena banyak dikuasai oleh hawa nafsunya. Dan ayat terakhir ke-3 menjelaskan bahwa, Kecuali orang-orang yang beriman kepada Allah, mengerjakan amal saleh dengan penuh kepatuhan, dan saling menasihati sesamanya untuk berpegang teguh pada

kebenaran--baik berupa keyakinan, ucapan maupun tindakan dan saling menasihati untuk bersabar atas segala kesulitan yang dialami orang yang berpegang teguh dalam beragama. Mereka adalah orang-orang yang selamat dari kerugian tersebut dan beruntung di dunia dan akhirat. Oleh karena itu efisiensi waktu sangatlah penting dalam pengerjaan *composing web service* dengan solusi menggunakan *web services discovery*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, maka identifikasi masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana membangun *web service discovery* menggunakan *cosine similarity* pada *web service repository*?
2. Berapa akurasi *query* untuk mendapatkan rekomendasi *web service* pada *web service discovery*?

1.3 Tujuan Penelitian

Berdasarkan identifikasi masalah diperoleh tujuan dalam penyusunan penelitian, yaitu:

1. Membangun aplikasi *web service discovery* menggunakan *cosine similarity* pada *web service repository.*
2. Menghitung akurasi *query* untuk mendapatkan rekomendasi *web service* pada aplikasi *web service discovery.*

1.4 Batasan Masalah

Adapun batasan masalah pada penelitian ini mencakup:

1. Domain *web service* yang digunakan adalah proses bisnis pada ERP pondok pesantren bagian akademik.

2. Data XML menggunakan standar WSDL yang telah di-*decompose*.
3. Penelitian dilakukan sampai tahap *web service discovery*.

1.5 Manfaat Penelitian

Penelitian ini dilakukan dengan mempertimbangkan manfaat sebagai berikut:

1. Mempermudah dan mempercepat *user* mengembangkan *web service* sehingga tidak membuat sistem dari nol dengan cara memanfaatkan *web service* yang tersedia dari *company* lain.
2. *Low cost* dalam memulai pengembangan *web service* (*for developer web services*).

BAB 2

KAJIAN PUSTAKA

2.1 *Web Service Discovery*

Metode *semantic web* telah membantu terjadinya revolusi dalam hal penyampaian dan pemanfaatan informasi pada *world wide web*. Sebagai contoh, jika seseorang ingin menyusun informasi dari beberapa situs web sekaligus, maka dengan teknologi web yang sekarang ada, dia harus mengunjungi situs-situs tersebut satu-persatu dan kemudian melakukan *cut and paste* pada *content* dari masing-masing situs untuk menciptakan suatu informasi yang menyeluruh. Hal ini sangatlah membuang waktu dan tenaga, karena dilakukan secara manual oleh manusia, dan dengan berbasis teknologi yang sudah ada tidak akan mungkin dibuat otomatisasinya, mengingat halaman web berbasis HTML hanyalah dirancang untuk dipahami oleh manusia bukan mesin.

Dengan metode *semantic web*, data berbasis HTML dapat dirubah menjadi format yang dapat dipahami oleh mesin, sehingga mesin dapat melakukan proses pengumpulan informasi dan memahami hubungan antara informasi. *Semantic web* mampu melakukan perubahan ini dengan bantuan XML (*Extensible Markup Language*) dan *data language standards* seperti RDF (*Resource Description Framework*) dan OWL (*Ontology Web Language*), dua standarisasi dari W3C (*World Wide Web Consortium*) (Mutiara, Muslim, Putri, Oswari, & Silfianti, 2013). Jadi, semantic web yang diimplementasikan pada penelitian ini aplikasi *web service discovery*.

Web service (WS) merupakan tren yang banyak digunakan di dunia. Disisi lain perusahaan di dunia mulai bergerak menuju *Service Oriented Architecture* (SOA), maka *web service* saat ini banyak dibutuhkan. Sebagian besar dari *service oriented architecture*, *business-to-business* dan *business-to customer systems*, *web service* memainkan peranan penting untuk melakukan transaksi sehari-hari dan pertukaran informasi. *Web service* adalah *public interface* dari jarak jauh yang dapat memanggil aplikasi untuk melakukan serangkain fungsi bisnis (Bose, 2008).

Bersama dengan meningkatnya jumlah *web service*, permasalahan dalam mencari *web service* yang diinginkan dari banyaknya *web service* yang ada menjadi penting. Oleh karena itu, menemukan *web service* yang tepat sesuai dengan kebutuhan menjadi tantangan, atau bisa disebut dengan *web service discovery*.

Penelitian Bose memaparkan bahwa munculnya *Service Oriented Architecture*, *web service* menjadi pusat perhatian. Hal ini menandakan pentingnya membangun proses yang efektif dan handal untuk *web service discovery*. Peneliti menyajikan pendekatan untuk meningkatkan akurasi *web service discovery* dengan mencari *web service* yang sama secara semantik atau *query* dari user menggunakan *support-based latent semantic kernel.* Evaluasi dari penelitian ini menegaskan bahwa keakuratan *web service discovery* dengan metode yang diusulkan menunjukkan perbaikan yang signifikan atas metode penemuan tradisional (Bose, 2008).

Penelitan berikutnya adalah *web service discovery* untuk membantu pengguna menemukan dan mengolah *web services*, dan dapat mencocokkan *interfaces* dengan presisi tinggi saat parameter *interfaces* tersebut mengandung sinonim,

abbreviations, dan kombinasi fragmen yang tidak teratur. Pendekatan ini menggunakan mining secara semantic yaitu *web service operations discovery algorithm* (OpD). Pertama, penelitian ini mengusulkan *conceptual web service descrtiption model* di mana akan dimasukkan tipe *path* untuk parameter disertai *traditional text decription*. Kemudian berdasarkan model deskripsi ini, peneliti menganalogikan semantik dari *interface* untuk membuat *index libraries* dengan mengelompokkan nama *interface* dan fragmen. *index libraries* ini dapat membantu menyediakan *interface* dengan efisiensi tinggi yang dapat dilakukan *matching* tidak hanya sinonim tetapi juga kombinasi singkatan dan fragmen (Cheng, 2016).

OpD *discovery* menghasilkan dua tipe dari *web service* yaitu service dengan *single operations* dan *composite operartions*. Metode yang diusulkan tersebut memiliki kelemahan yaitu tidak dapat menemukan sekumpulan *web services* yang *composite*, jadi masih ada kelemahan dalam penerapan semantiknya.

Penelitan hamzah terkait clustering menggunakan fungsi *similarity* terhadap dokumen teks berbahasa Indonesia. *Clustering* dokumen teks banyak diteliti karena peranan pentingnya dalam bidang *text mining* dan *information retrieval*. Dalam algoritma *clustering* pemilihan fungsi jarak atau fungsi similaritas antar objek menjadi kunci keberhasilan algoritma. Pada fungsi jarak *Euclidean* memiliki kelemahan jika digunakan untuk *vector* berdimensi sangat tinggi yang menyebabkan kinerja *clustering* menurun. Alternatif dari fungsi jarak adalah fungsi similaritas, antara lain *jaccard*, dice, *cosine*, dan *pearson*. Hasil penelitian menunjukkan bahwa hasil *clustering* terbaik adalah jika menggunakan fungsi *cosine* dengan rata-rata *F-measure* untuk seluruh koleksi 0,9313, sementara

terburuk menggunakan fungsi jarak *eucledian* dengan rata-rata F-measure 0,4668. Secara waktu komputasi fungsi *cosine* juga memiliki kinerja tercepat dengan rata-rata 12,9 detik sedangkan terjelek adalah *pearson* dengan rata-rata 58,2 detik (Hamzah, 2008).

Untuk menemukan *web services* dan meningkatkan akurasi *query* serta efiensi waktu komputasi peneliti akan mengimplementasikan *web service discovery* dengan menerapkan *semantic* menggunakan *cosine similarty*, sehingga berbagai jenis *web service* dalam ruang lingkup penelitian dapat dilakukan *discovery* dengan penerapan semantik.

Berikut ini beberapa penelitian terkait *cosine similarity* dan akurasi terhadap *discovery web service*.

Tabel 2.1 *Literatur review ws discovery* dengan akurasi

No.	Penelitian Terkait	Hasil Penelitian	Kontribusi Peneliti
1.	Hamzah, A., Soesianto, F., Susanto, A., & Istiyanto, J. E. (2008). "Studi Kinerja Fungsi-Fungsi Jarak dan Similaritas dalam Clustering Dokumen Teks Berbahasa Indonesia. Seminar Nasional Informatika, 1979-2328.	Alternatif dari fungsi jarak adalah fungsi similaritas, antara lain *jaccard*, dice, *cosine*, dan *pearson*. Hasil penelitian menunjukkan bahwa hasil *clustering* terbaik adalah jika menggunakan fungsi *cosine*, sementara terburuk menggunakan fungsi jarak *eucledian*. Secara waktu komputasi fungsi *cosine* juga memiliki kinerja tercepat sedangkan terjelek adalah *pearson*.	Untuk menemukan *web services* dan meningkatkan akurasi *query* serta efiensi waktu komputasi peneliti akan mengimplementasikan *web service discovery* dengan menggunakan *cosine similarty*.

No.	Penelitian Terkait	Hasil Penelitian	Kontribusi Peneliti
2.	Aishwarya and Nayak, Richi and Bruza, Peter Bose, "*Improving web service discovery by using semantic models*," in *International Conference on Web Information Systems Engineering*, 2008, pp. 366-380.	Peneliti menyajikan pendekatan untuk meningkatkan akurasi *web service discovery* dengan mencari *web service* yang sama secara semantik atau *query* dari user menggunakan *support-based latent semantic kernel*.	Pencarian yang akan dilakukan secara semantic atau *query* dari user menggunakan *cosine similarity*.
3.	Wen and Zeng, Nancy and Wang, Ning and others Zhu, "*Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS mplementations*," *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, pp. 1-9, 2010.	Terdapat beberapa aturan yang umum digunakan dalam menghitung akurasi, yaitu *true positive* (TP), *true negative* (TN), *false negative* (FN), dan *false positive* (FP).	Perhitungan akurasi menggunakan formula *true positive* (TP), *true negative* (TN), *false negative* (FN), dan *false positive* (FP).
4.	David Martin Powers, "*Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*," *Bioinfo Publications*, 2011.	Penerapan *Recall*, *Precision* dan F-*measure* dianggap cacat karena mengabaikan sel negatif tabel kontinjensi yang sebenarnya, dan mudah dimanipulasi dengan *biasing the preditcions*. Masalah ini dapat dipecahkan dengan menggunakan Akurasi.	Untuk mengevaluasi akurasi *query* yang diusulkan, maka akan digunakan pengujian sistem menggunakan ukuran akurasi.

No.	Penelitian Terkait	Hasil Penelitian	Kontribusi Peneliti
5.	Cheng, B., Zhao, S., Li, C., & Chen, J. (2016, June). MISDA: *web services discovery approach based on mining interface semantics. In Web Services (ICWS), 2016 IEEE International Conference on* (pp. 332-339). IEEE.	Pendekatan ini menggunakan mining secara *semantic* yaitu *web service Operations Discovery algorithm* (OpD).	Peneliti akan meng-implementasikan *web service discovery* de-ngan menerapkan *semantic m*engguna-kan *cosine similarty*.

2.2 *Semantic Search*

Semantic atau dalam bahasa indonesia semantik, adalah ilmu tentang makna kata dan kalimat; pengetahuan mengenai seluk-beluk dan pergeran arti kata. Bagain struktur bahasa yang berhubungan dengan makna ungkapan atau struktur makna suatu wicara. Jaringan *semantic* (*semantic network*) adalah sebuah jaringan yang mewakili hubungan antar konsep. Jaringan *semantic* biasa digunakan sebagai bentuk representasi pengetahuan. Jaringan *semantic* berupa grafik berarah atau tidak yang terdiri dari simpul dan garis. Simpul mewakili konsep (Sowa, 1987).'

Pencarian dengan jaringan *semantic* berarti pencarian dokumen berdasarkan kata kunci penelusuran dan makna yang terkait dengan kata kunci tersebut. Pencarian semantik berusaha untuk meningkatkan akurasi pencarian dengan memahami maksud pencari dan makna kontekstual istilah seperti yang ditampilkan dalam data pencarian. Dengan kata lain, pencarian *semantic* adalah pencarian suatu konten berdasarkan konteks yang tepat. Konten adalah teks bertulis sedangkan konteks adalah kondisi keberadaan teks tersebut. Dikutip

penulis dari penelitian Rahutomo (2009), tujuan pencarian *semantic* adalah mencari konten yang sesuai dengan konteks yang diinginkan pengguna.

Ada dua jenis pencarian semantik menurut Pollock (2009). Jenis pertama adalah pencarian *semantic* dengan memberikan hasil berupa navigasi. Pengguna menggunakan mesin pencari sebagai alat navigasi untuk mengarahkan ke dokumen yang diinginkan. Navigasi ini dapat berupa link. Jenis yang kedua adalah dengan memaukkan frasa atau kalimat yang menunjukkan keinginan pengguna untuk mendapatkan informasi. Pada jenis yang kedua ini, pengguna akan mendapatkan keseluruhan dokumen yang akan memberikan informasi secara lengkap. Intinya, pencarian *semantic* memberikan saran bagi pengguna berdasarkan penarikan kesimpulan yang dilakukan oleh sistem berdasarkan batasan-batasan tertentu (Sarno, Yeni, & Fitri, 2012).

2.3 *Information Retrieval*

Information Retrieval merupakan sistem yang menerima *query* dari pengguna, kemudian dilakukan *ranking* terhadap dokumen berdasar kesesuaian terhadap *query*. Hasil rangking yang diberikan pada pengguna merupakan dokumen yang menurut sistem memiliki relevansi terhadap *query*, tetapi tingkat relevansi itu sendiri merupakan hal yang subjektif tergantung dari pengguna yang dipengaruhi oleh berbagai macam faktor seperti topik, pewaktuan, sumber informasi maupun tujuan pengguna. Model sistem temu kembali menentukan detail sistem temu yaitu meliputi representasi dokumen maupun *query*, *retrieval function*, dan *relevance notation* dokumen terhadap *query*.

Information Retrieval terbagi dari beberapa bagian yang dijabarkan sebagai berikut:

1. *Text Operations,* meliputi pemilihan kata-kata dalam *query* maupun dokumen (*term selection*) dalam proses transformasi dokumen atau *query* menjadi *term index* (indeks kata-kata).
2. *Query formulation*, memberi bobot pada indeks kata-kata *query*.
3. Rangking, mencari dokumen-dokumen yang relevan terhadap *query* dan mengurungkan dokumen tersebut berdasarkan kesesuaiannya dengan *query*.
4. *Indexing*, membangun basis data indeks dari koleksi dokumen dilakukan terlebih dahulu sebelum pencarian dokumen dilakukan (Wisnu, 2015).

2.4 *Data Mining*

Sistem manajemen basis data tingkat lanjut dan teknologi *yang* mampu untuk menampilkan "banjir" data dan untuk mentransformasikannya ke dalam basis data yang berukuran besar. Diperlukan teknik baru yang secara pintar dan otomatis mentransformasikan data-data yang diproses untuk menghasilkan pengetahuan dan informasi yang berguna.

Data mining adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual. Kata *mining* berarti usaha untuk mendapatkan sedikit barang berharga dari sejumlah besar material dasar (Pramudiono, 2003).

Secara sederhana *data mining* dapat dikatakan sebagai proses menyaring atau "menambang" pengetahuan suatu pengetahuan baru dari sejumlah data yang besar

menggunakan serangkaian proses matematika. Hal penting yang terkait *data mining* adalah:

a. *Data mining* merupakan proses otomatis terhadap data yang ada.

b. Data yang akan diproses berupa data yang sangat besar.

Tujuan *data mining* adalah mendapatkan hubungan atau pola yang mungkin memberikan indikasi bermanfaat.

2.5 *Text Mining*

Text mining merupakan salah satu bidang khusus dari *data mining*. *Text mining* dapat didefiniskan sebagai suatu proses menggali informasi dimana seorang *user* berinteraksi dengan sekumpulan dokumen menggunakan *tool* analisis yang merupakan komponen-komponen dalam *data mining* **(**Han dan Kamber, 2006**)**.

Dalam *text mining* berbeda dengan dengan *data mining* dimana *data mining* yang digunakan adalah *structured data* sementara dalam *text mining* umumnya data yang ditemui adalah *semi-structured* atau *unstructured*. Sementara keduanya memiliki permasalah yang sama yaitu jumlah data yang besar, dimensi yang tinggi, dan data juga struktur yang terus berubah.Struktur teks yang kompleks dan tidak lengkap, arti yang tidak jelas dan tidak standar, dan bahasa yang berbeda ditambah terjemahan yang tidak akurat memberikan tantangan tambahan pada *text mining.*

Text mining dalam prakeknya mencari pola-pola tertentu, mengasosiasikan satu bagian teks dengan lain berdasar aturan-aturan tertentu, kata-kata yang dapat mewakili sehingga dapat dilakukan analisa keterhubungan antar satu dengan lain,

dalam kumpulan dokumen yang sangat banyak. Dokumen yang ada bisa bersifat statis, yaitu dokumen yang tidak akan di perbarui lagi ataupun dinamis yaitu dokumen yang akan selalu diperbarui dalam rentang waktu tertentu.

1. Tahapan *Text Mining*

a. *Case Folding*

Mengubah semua huruf dalam dokumen menjadi huruf kecil (*lowercase*). Dalam tahap ini juga karakter selain huruf dihilangkan.

b. *Tokenizing*

Memotong tiap kata dalam kalimat atau *parsing* dengan menggunakan spasi sebagai delimiter yang akan menghasilkan token berupa kata.

c. *Filtering* (*Stopword*)

Menyaring kata yang didapat dari proses *tokenizing* yang dianggap tidak penting atau tidak memiliki makna dalam proses *text mining* yang disebut *stoplist*. *Stoplist* atau *stopword* berisi kata-kata umum yang sering muncul dalam sebuah dokumen dalam jumlah banyak namun tidak memiliki kaitan dengan dengan tema tertentu.

Tiap kata yang diperoleh dari *tokenizing* akan dicocokkan dalam kamus *stopword* di dalam *database*, jika kata tersebut cocok dengan salah satu kata dalam *stopword* maka kata tersebut akan dihilangkan, sementara yang tidak cocok akan dianggap cocok dan diproses ke tahap selanjutnya.

d. *Stemming*

Mengembalikan kata-kata yang diperoleh dari hasil *filtering* ke bentuk dasarnya, menghilangkan imbuhan awal (*prefix*) dan imbuhan akhir (*sufix*) sehingga di dapat kata dasar. Metode stemming memerlukan masukan berupa

kata yang terdapat dalam suatu dokumen, dengan menghasilkan keluaran berupa *root word*.

f. *Analyzing*

Keterhubungan antar kata dalam dokumen akan ditentukan dengan menghitung frekuensi *term* pada dokumen. Tahap *analyzing* lebih sering dikenal dengan tahap pembobotan.

Penelitian terkait oleh Dwija dan Anadini (2015) memanfaatkan *Information Retrieval* pada teks *mining* untuk menemukan ide pokok dalam teks pada artikel berbahasa Inggris, dapat membantu pembaca untuk lebih mudah memahami isi artikel dan menghemat waktu yang dibutukan untuk membaca secara garis besar dengan memberikan sebuah konten yang lebih ringkas dari artikel awal.

Basis pertama yang digunakan adalah *Term Frequency Inverse Document Frequency* (TF-IDF) untuk memberikan nilai dan menggunakan pembobotan *Vector Space Model* untuk menarik hasil dari pencarian ide pokok. Kata kunci yang digunakan dalam proses peringkasan adalah judul dari artikel.

2.6 *Web Services Description Languange* (WSDL)

WSDL merupakan sebuah bahasa berbasis XML yang digunakan untuk mendefinisikan *web service* dan menggambarkan bagaimana cara untuk mengakses *web service* tersebut. WSDL menjelaskan *method-method* apa saja yang tersedia dalam suatu *web service*, parameter apa saja yang diperlukan untuk memanggil suatu *method*, dan apa hasil atau tipe data yang dikembalikan oleh *method* yang akan dipanggil tersebut (http://kemenpora.go.id).

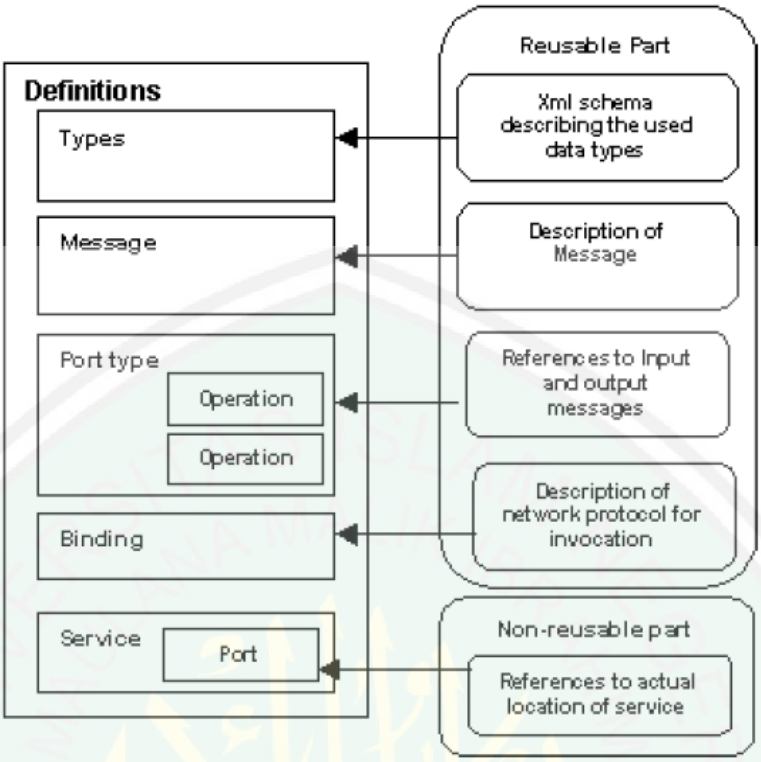
Dalam Manes disebutkan bahwa ada 5 elemen utama dalam sebuah dokumen WSDL yakni (Sycara, 2006):

1. Elemen <*type*>
   a. Menjelaskan semua jenis data yang akan digunakan antara klien dan *server*. Tidak termasuk skema *built-in* XML tipe-tipe sederhana (contoh : *string, integers*).
   b. Jika jenis WSDL tidak ditentukan maka *default*-nya adalah jenis Skema XML.
2. Elemen <*message*>

   Menjelaskan pesan satu arah (permintaan atau respon):
   a. Message nama
   b. Message parameter
   c. Message nilai kembali
3. Elemen <*portType*>
   a. Menggabungkan unsur beberapa pesan untuk membentuk perjalanan operasi "*round trip*". Contoh: Dapat menggabungkan 1 pesan permintaan dan 1 pesan respon menjadi permintaan tunggal / respon operasional.
   b. Dapat mendefinisikan beberapa operasi.
4. Elemen <*binding*>

   Spesifik tentang bagaimana layanan akan dilaksanakan pada kabel. Berfungsi untuk memetakan operasi-operasi dan pesan yang terdefinisikan pada *port type* ke protokol tertentu.
5. Elemen <*service*>

   Ditentukan alamat untuk memanggil layanan. Paling sering sebuah URL.



Gambar 2.1 Elemen WSDL

WSDL mendefinisikan *service* sebagai sebuah koleksi dari *endpoints network*. Sebuah definisi abstrak dari *endpoints* dan *messages* adalah bersifat terpisah dari pembangunan network atau penyatuan data format. Pembagian ini menyebabkan penggunaan kembali abstract description dari data yang akan dipertukarkan (*message exchange*) dan *abstract collection* dari operasi (*ports*).

Protokol konkret dan spesfikasi data format bagi tipe port tertentu menentukan *binding* yang dapat digunakan kembali (*reusable*). Sebuah port adalah sebuah *network address* yang dikombinasikan *reusable binding*, sebuah *service* adalah koleksi dari port-port.

2.7 *Vector Space Matching* (VSM)

Vector space matching menghitung kemiripan dengan mendefinisikan sebuah *vector* yang mempresentasikan tiap dokumen, dan sebuah *vector* yang mempresentasikan *query*. Jika kata-kata dalam dokumen dapat direpresentasikan dalam *vector*, sangat dimungkinkan untuk membandingkan dokumen dengan *query* untuk menunjukkan kemirapan konten keduanya.

Jika sebuah *query* dianggap mirip dengan sebuah dokumen, koefesien kemiripan yang mengukur kemiripan antara dokumen dan *query* dapat dihitung. Dokumen yang kontennya paling mirip dengan konten pada *query* dianggap paling relevan. Gambar mengilustraikan notasi dasar dari *vector space matching* yang menggambarkan beberapa *vector* yang merepresentasikan sebuah *query* dan tiga dokumen.

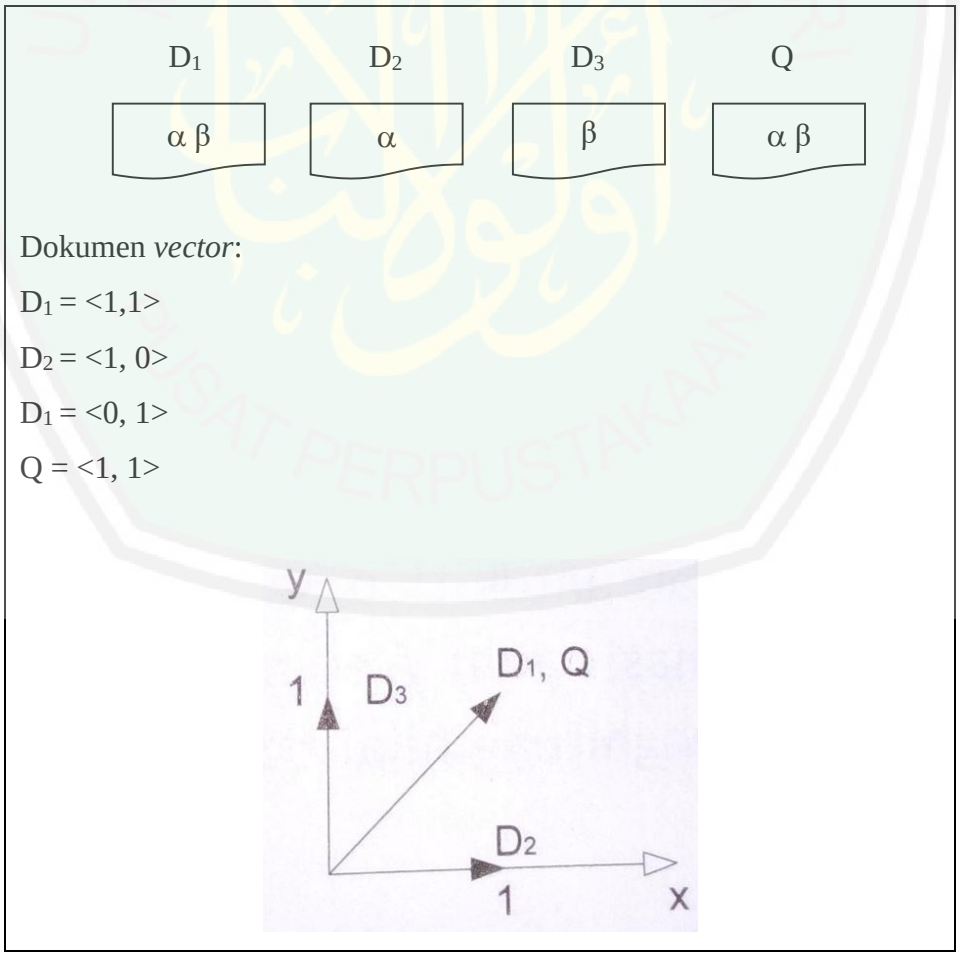
Pencocokan ini memerlukan pembentukan *vector* yang merepresentasikan kata-kata dalam dokumen dan *vector* lain yang merepresentasikan kata-kata dalam *query*. Kemudian, sebuah metode harus dipilih untuk mengukur kedekatan *vector* dokumen dan *vector query*.

Cara tradisional untuk menentukan kedekatan dua *vector* yang berbeda adalah dengan mengukur sudut di antara keduanya. Besar sudut dapat diperolah dari perhitungan perkalian silang (*inner product*) atau perkalian titik (*cross product*). Pada umumnya kata koefesien kemiripan digunakan untuk mengganti kata sudut.

Misalkan sekumpulan dokumen hanya terdiri dari dua kata yang berbeda, yakni α dan β. Semua *vector* hanya mengandung dua komponen. Komponen pertama merepresentasikan adanya α dan komponen kedua merepresentasikan adanya β. Arti paling sederhana dari pembentukan *vector* adalah menempatkan

angka 1 pada komponen *vector* yang sesuai jika terdapat α atau β. Dan 0 jika tidak ditemukan kata α atau β. Sebuah dokumen, D_1, mengandung dua kata α dan tidak ditemukan kata β. *Vector* dari dokumen tersebut adalah <2,0>.

Contoh sederhana ditunjukkan gambar berikut. Sebuah komponen pada tiap *vector* memerlukan kata berbeda pada kumpulan dokumen. Diberikan tiga buah dokumen dan sebuah *query* yang mengandung kata α dan/atau β. Koefien kemiripan antara *query* dan dokumen dapat dihitung sebagai jarak dari *query* ke tiga dokumen. Dari contoh ini dapat dilihat bahwa D_1 diwakili oleh vektor yang sama dengan *query* sehingga D_1 akan menduduki urutan pertama pada kumpulan hasil.



Gambar 2.2 *vector space model* dengan 2 kata

Terkadang seseorang menggunakan derajat kepentingan pada beberapa kata yang menunjukkan bahwa kata tersebut lebih penting daripada kata lain. Cara menggunakannya adalah dengan memberi bobot kata secara manual.

Cara otomatis juga dapat digunakan, biasanya berdasarkan frekuensi kemunculan kata pada sebuah dokumen dalam sekumpulan dokumen. Semakin sering suatu kata muncul pada dokumen, kata tersebut memiliki bobot yang semakin besar.

Vector untuk tiap dokumen memiliki n komponen dan mengandung sebuah masukan untuk tiap kata berbeda dalam sekumpulan dokumen. Komponen-komponen *vector* diisi dengan bobot yang dihitung untuk tiap kata dalam kumpulan dokumen. Kata-kata pada tiap dokumen secara otomatis diberi bobot berdasarkan kemunculan kata-kata tersebut dalam kumpulan dokumen dan seberapa sering kata tersebut muncul dalam dokumen tertentu.

Bobot kata untuk tiap kata dalam *vector* dokumen tidak bernilai nol hanya jika kata itu muncul pada dokumen. Untuk kumpulan dokumen yang besar dant erdiri dari beberapa dokumen kecil, dokumen *vector* akan bernilai mendekati nol. Misalkan sebuah kumpulan dokumen dengan 10.000 kata berbeda menghasilkan 10.000 vektor dimensi untuk tiap dokumen. Sebuah dokumen yang hanya memiliki 100 kata berbeda akan mempunyai 9900 vektor dokumen yang mengandung komponen kosong.

a. TF/IDF

Faktor pembobotan untuk tiap kata dalam dokumen didefinisikan sebagai kombinasi *term frequency* dan *inverse document frequency*. Untuk menghitung nilai bobot kata ke j dari dokumen I, digunakan rumus pada persamaan 1:

$$W_{ij} = TF_{ij} \text{ X } IDF_j \qquad \text{(Persamaan 1)}$$

Keterangan:

T : jumlah kata berbeda yang terdapat pada dokumen

TF_{ij} : *term frequency*, yakni jumlah kemunculan kata T_j dalam dokumen D_i

DF_j : *document frequency*, yakni jumlah dokumen yang mengandung T_j

IDF_j : $\log\left(\frac{D}{DF_j}\right)$ dengan d adalah jumlah semua dokumen dalam koleksi. IDF_j adalah *inverse document frequency*.

Pada persamaan 1, rumus pembobotan tersebut terdapat kelemahan, yakni jika sebuah kata muncul di semua kumpulan dokumen akan memiliki bobot nol. Untuk mengatasinya, menurut Konchady (2016) dalam Sulistyo (2008), dapat ditambahkan nilai 1 pada sisi IDF sehingga dapat dituliskan:

$$W_{ij} = TF_{ij} \text{ X } IDF_j \qquad \text{(Persamaan 2)}$$

$$W_{ij} = TF_{ij} \text{ X } \left(\log\left(\frac{D}{DF_j}\right)\right) \qquad \text{(Persamaan 3)}$$

$$W_{ij} = TF_{ij} \text{ X } \left(\log\left(\frac{D}{DF_j}\right)\right) + 1 \qquad \text{(Persamaan 4)}$$

Misalkan sebuah dokumen terdiri dari sejumlah T kata yang berbeda. *Vector* yang dibentuk adalah D (d1, d2, …, dt).

Diberikan sebuah *query* Q yang terdiri dari beberapa kata sejumlah T. Vektor yang dibangun adalah Q (q1, q2, …, qt). Beberapa persamaan untuk membandingkan *vector query* dengan *vector* dokumen telah dikembangkan yakni

Cosine coefficient. Dengan W adalah bobot dari *query* dan dokumen. Setelah mendapatkan nilai *cosine* tiap-tiap dokumen, maka hasil bobot dari kata kunci diurutkan. Bobot yang besar menjadi prioritas sebagai dokumen yang memiliki hubungan dengan kata kunci (Sarno, Yeni, & Fitri, 2012).

b. *Cosine Similarity*

Secara umum, fungsi *similarity* adalah fungsi yang menerima dua buah objek dan mengembalikan nilai kemiripan (*similarity*) antara kedua objek tersebut berupa bilangan riil. Umumnya, nilai yang dihasilkan oleh fungsi *similarity* berkisar pada interval [0...1]. Namun ada juga beberapa fungsi *similarity* yang menghasilkan nilai yang berada di luar interval tersebut. Untuk memetakan hasil fungsi tersebut pada interval [0...1] dapat dilakukan normalisasi. *Cosine similarity* adalah perhitungan kesamaan antara dua vektor n dimensi dengan mencari kosinus dari sudut diantara keduanya dan sering digunakan untuk membandingkan dokumen dalam *text mining*. Rumus c*osine similarity* adalah sebagai berikut (Triana, Saptono, & Sulistyo, 2014):

$$Similarity\ (x,y) = \cos(\theta) = \frac{x.y}{\|x\|\|y\|}$$

Dimana:

x.y : *vector dot product* dari x dan y, dihitung dengan $\sum_{j=1}^{t}(Q_j\ .D_{ij})$

||x|| : panjang *vector* x, dihitung dengan $\sum_{j=1}^{t} Q_j^2$

||y|| : panjang *vector* y, dihitung dengan $\sum_{j=1}^{t} {D_{ij}}^2$

Persamaan untuk *Cosine coefficient* adalah

$$Cosine(Q, D_i) = \frac{\sum_{j=1}^{t} q_j d_{ij}}{\sqrt{\sum_{j=1}^{t} (q_j)^2 \sum_{j=1}^{t} (d_{ij})^2}}$$

Robinson (2014) mengimplementasikan metode *generalized vector space model* pada *information retrieval* untuk pencarian informasi pada kumpulan dokumen teknik elektro di UPT BPI LIPI. *generalized vector space model* merupakan salah satu *algebraic model*. Proses yang dilakukan terbagi menjadi dua, yaitu proses *preprocessing* (*reading text*, *tokenization*, *filtration*, *stemming* dan *indexing*) dan proses yang kedua adalah menghitung relevansi antara kumpulan dokumen yang telah di-*preprocess* dengan *query* yang diinginkan pengguna.

Aishwarya dkk. (2008), melakukan pendekatan untuk meningkatkan akurasi *web service discovery* dengan mencari *web service* yang sama secara semantik atau *query* dari *user* menggunakan *support-based latent semantic kernel.*

Jadi, peneliti akan mengimplementasikan pencarian secara *semantic* atau *query* dari user menggunakan metode *vector space model*.

BAB 3

METODOLOGI PENELITIAN

3.1 Desain Penelitian

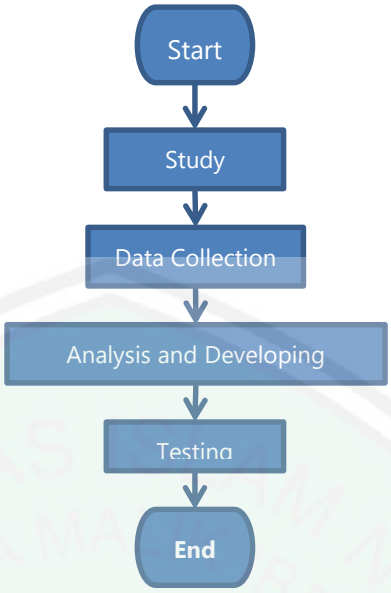
Desain penelitian menjabarkan bagaimana tahapan dan mekanisme penelitian dilakukan, meliputi tipe penelitian, prosedur penelitian, dan metode pengolahan data. Berikut penjelasan dari masing-masing mekanisme penelitian.

3.1.1 Tipe Penelitian

Penelitian ini menggunakan pendekatan kuantitatif. Metode penelitian kuantitatif yang akan digunakan untuk meneliti pada populasi atau sampel tertentu, pengumpulan data menggunakan instrumen penelitian, analisis data bersifat kuantitatif/statistik, dengan tujuan untuk menguji hipotesis yang telah ditetapkan.

3.2.1 Prosedur Penelitian

Prosedur penelitian menjabarkan bagaimana penelitian akan dilaksanakan. Tahapan pelaksanaan penelitian dimulai dari pengkajian literatur, pengumpulan data, analisa dan pengembangan sistem hingga tahap pengujian.



Gambar 3.1 Alur Prosedur Penelitian

Studi pustaka berfungsi untuk mengarahkan penelitian yang akan dikerjakan berdasar beberapa penelitian lain, tahap selanjutnya mengumpulkan data yang akan diimplementasikan pada pengembangan sistem dengan metode yang digunakan. Tahap terakhir pengujian sistem.

3.3.1 Pengumpulan Data

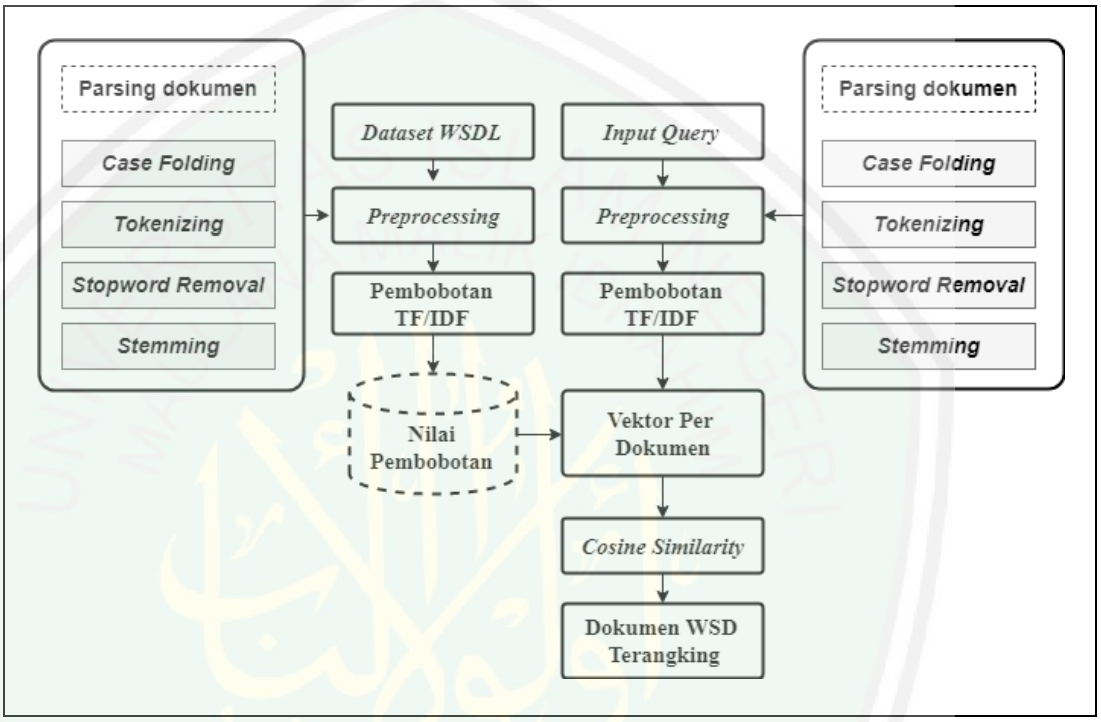
Penelitian ini menggunakan data sekunder. Data sekunder adalah data yang diperoleh dari sumber lain selain responden. Data sekunder dalam penelitian ini diambil dari penelitian (Suprianto, 2017) oleh saudara Eko. Data *Web Services* pondok pesantren tersebut berformat WSDL.

3.2 Perancangan Sistem

Perancangan sistem digambarkan untuk mempermudah implementasi, pengujian, dan analisis. Sistem dibangun menggunakan bahasa pemrograman PHP. Berikut adalah gambaran sistem yang akan dibangun mulai dari persiapan *dataset* WSDL, *preprocessing*, hingga metode VSM untuk melakukan proses *discovery web service*.

3.2.1 Metode Pengolahan Data

Metode pengolahan data memaparkan bagaimana data diolah untuk menghasilkan output yang diinginkan. Pada gambar 3.2 ditampilkan alur pengembangan sistem *web service discovery*:



Gambar 3.2 Alur pengembangan sistem

Berikut adalah sistematika tahapan dari alur pengembangan sistem:

1. Tahapan pertama yaitu mengolah data WSDL dari *web services* yang telah di-*Decompose* dan sudah di-*parsing* sehingga informasi dari WSDL tersebut dapat diolah ke tahap selanjutnya.
2. Tahap kedua dilakukan *preprocessing* terhadap dokumen WSDL. Meliputi, *case folding*, *tokenizing*, *stopword removal*, dan *stemming*.
3. Tahap ketiga, dihitung bobot dari dokumen WSDL tersebut. Selanjutnya, disimpan ke dalam database.
4. Tahap ke empat, dihitung nilai *vector* dari masing-masing dokumen.

5. Tahap ke lima, mengukur kemiripan dokumen menggunakan *cosine similarity*.
6. Terakhir pada aplikasi WS *Discovery* tersaji *list* WSDL yang diinginkan.

3.3 WSDL File

WSDL Bahasa XML yang digunakan untuk mendefinisikan *web service* dan bagaimana cara untuk mengakses *web service* tersebut. Gambar 3.3 adalah contoh data WSDL *atomic* yang digunakan sebagai *input parsing*. Data berupa WSDL proses rencana buku pondok pesantren. Pada penelitian ini menggunakan *web service* bagian akademik. Pada wsdl tersebut terdapat beberapa informasi yaitu parameter *web service* seperti *message*, *input*, *output*, *type*, dan *operation*.

```
▶ <message name="psbView.tingkatpenempatanRequest">...</message>
▶ <message name="psbView.tingkatpenempatanResponse">...</message>
▶ <message name="psbView.kelaspenempatanRequest">...</message>
▶ <message name="psbView.kelaspenempatanResponse">...</message>
▶ <message name="psbView.tempatkelasRequest">...</message>
▶ <message name="psbView.tempatkelasResponse">...</message>
▶ <message name="psbView.agamaRequest">...</message>
▶ <message name="psbView.agamaResponse">...</message>
▶ <message name="psbView.sukuRequest">...</message>
▶ <message name="psbView.sukuResponse">...</message>
▶ <message name="psbView.kondisiRequest">...</message>
▶ <message name="psbView.kondisiResponse">...</message>
▶ <message name="psbView.pendidikanRequest">...</message>
▶ <message name="psbView.pendidikanResponse">...</message>
▶ <message name="psbView.pekerjaanRequest">...</message>
▶ <message name="psbView.pekerjaanResponse">...</message>
▼ <portType name="PsbKeAkademikanViewPortType">
  ▼ <operation name="psbView.angkatanpsb">
      <documentation>Fetch array</documentation>
      <input message="tns:psbView.angkatanpsbRequest"/>
      <output message="tns:psbView.angkatanpsbResponse"/>

    </operation>
  ▶ <operation name="psbView.tingkatpenempatan">...</operation>
  ▶ <operation name="psbView.kelaspenempatan">...</operation>
  ▶ <operation name="psbView.tempatkelas">...</operation>
  ▶ <operation name="psbView.agama">...</operation>
  ▶ <operation name="psbView.suku">...</operation>
  ▶ <operation name="psbView.kondisi">...</operation>
  ▶ <operation name="psbView.pendidikan">...</operation>
  ▶ <operation name="psbView.pekerjaan">...</operation>
  </portType>
▼ <binding name="PsbKeAkademikanViewBinding"
  type="tns:PsbKeAkademikanViewPortType">
    <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  ▼ <operation name="psbView.angkatanpsb">
      <soap:operation soapAction="urn:server#angkatanpsb"
      style="rpc"/>
    ▼ <input>
        <soap:body use="encoded" namespace="urn:server"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      </input>

    ▼ <output>
        <soap:body use="encoded" namespace="urn:server"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      </output>
    </operation>
  ▶ <operation name="psbView.tingkatpenempatan">...</operation>
  ▶ <operation name="psbView.kelaspenempatan">...</operation>
  ▶ <operation name="psbView.tempatkelas">...</operation>
  ▶ <operation name="psbView.agama">...</operation>
  ▶ <operation name="psbView.suku">...</operation>
  ▶ <operation name="psbView.kondisi">...</operation>
  ▶ <operation name="psbView.pendidikan">...</operation>
  ▶ <operation name="psbView.pekerjaan">...</operation>
  </binding>
▼ <service name="PsbKeAkademikanView">
```

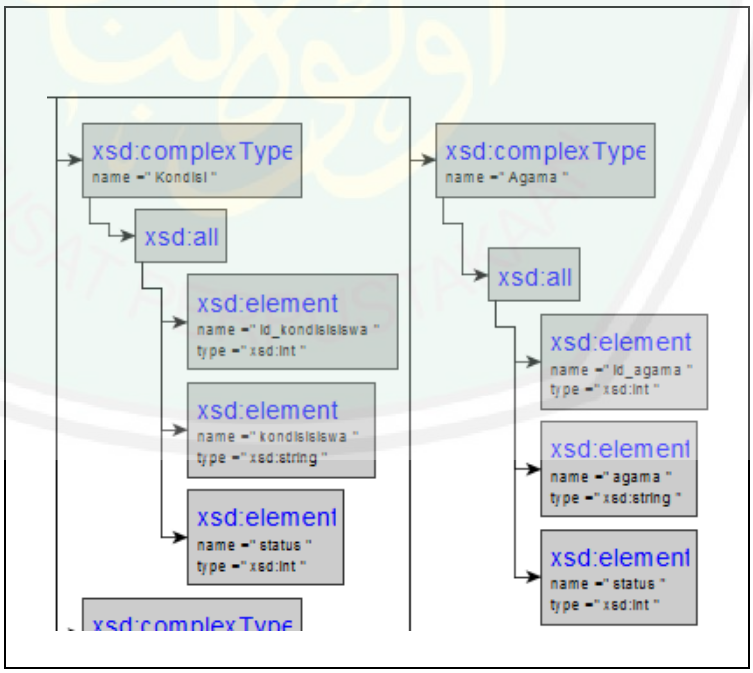
Gambar 3.3 Contoh dokumen WSDL

Data berupa WSDL proses pondok pesantren yang digunakan merupakan domain akademik (Suprianto, 2017).

3.4 Melakukan *Parsing*

Proses *parsing* file WSDL dari *web service atomic* untuk mengekstraksi informasi file tersebut. Selama tahap ini, informasi antara <*operation*>…</*operation*> diidentifikasi, misalnya *input* dan *output* yang berelasi dengan *tag* ini. WSDL adalah format XML yang menjelaskan *method-method*, *parameter*, dan tipe data apa saja yang tersedia dalam suatu *Web Service*.

Untuk ekstraksi WSDL *atomic* akan menggunakan *yWorks* yang merupakan *software* untuk mem-*parsing* dokumen XML ke dalam bentuk diagram *graph*.



Gambar 3.4 Hasil *parsing* WSDL ke *Graph*

Proses *parsing* file WSDL dari *web service atomic* untuk mengekstraksi informasi file tersebut. Untuk ekstraksi WSDL *atomic* menggunakan *yWorks* yang merupakan *software* untuk mem-*parsing* dokumen XML ke dalam bentuk diagram *graph*.

Data yang diparsing berupa varian proses bisnis yang telah diimplementasikan menjadi web service atau bisa disebut WSDL atomic dari *web service* akademik. Varian proses bisnis tersebut yaitu, proses bisnis referensi, psb, guru_pelajaran, dan jadwal_kalender.

Setelah di-*parsing* peneliti melakukan mapping semua *method* dan *elements* sesuai masing-masing *web service*. Pada element terdapat komponen "name" dan "type", keduanya merupakan *elements* Berikut adalah hasil *parsing* yang telah di-*mapping* ke dalam bentuk tabel untuk dilakukan analisis.

a. Proses Bisnis Referensi

Web service referensi berasal dari proses bisnis referensi yang memiliki berbagai *element* dari 10 *method*, yaitu angkatan, tingkatan, kelas, CariSiswaBerdasarkanDeptartemen, Semester, CekValidasi, LihatSiswaKelas, KelasByIdDepartemen, AngkatanByTahun, dan TerisiKelas.

Link: http://localhost/ws/_akademik_eko/1-referensi/referensiView.php?wsdl

Tabel 3.1 Hasil *mapping web service* referensi

No.	***Operation Name/Method***	***Element***	
		Name	***Type***
1.	Angkatan	id_departmen	int
		id_angkatan	int
		keterangan	string

No.	*Operation Name/Method*	*Element*	
		Name	***Type***
2.	Tingkatan	id_departmen	int
		keterangan	string
		id_tingkat	int
		tingkat	string
		aktif	int
		urutan	int
3.	Kelas	id_tahunajaran	int
		keterangan	string
		kapasitas	int
		id_tingkat	int
		id_kelas	int
		kelas	string
		nipwali	string
		aktif	int
4	CariSiswaBerdasarkanDeptartemen	namaayah	string
		tmp_lahir	string
		angkatan	string
		id_siswa	int
		nama	string
		kelamin	string
		tgl_lahir	string
5.	Semester	id_departemen	int
		id_semester	int
		keterangan	string
		semester	string
		aktif	int
6.	CekValidasi	id_dftvalidasi	int
		username	string
		password	string
		link	string
		id_santri	int
		status	int
7.	LihatSiswaKelas	keterangan	string
		id_siswa	int
		nis	string
		nisn	string
		nama	string
8.	KelasByIdDepartemen	id_kelas	int
		kapasitas	int
		kelas	string
		gender	string
9.	AngkatanByTahun	id_angkatan	int
		angkatan	string
10.	TerisiKelas	id_kelas	int

b. Proses Bisnis PSB

Web service PSB berasal dari proses bisnis PSB yang memiliki berbagai *element* dari 9 *method*, yaitu TempatKelas, Kondisi, Pendidikan, Pekerjaan, Angkatanpsb, Agama, TingkatPenempatan, Suku, KelasPenempatan.

Link: http://localhost/ws/_akademik_eko/2-psb/psbView.php?wsdl

Tabel 3.2 Hasil *mapping web service* PSB

No.	*Operation Name/Method*	*Element*	
		Name	***Type***
1.	TempatKelas	id_santri	int
		id_siswa	int
		nisn	string
		nis	string
		nama	string
2.	Kondisi	id_kondisisiswa	int
		kondisisiswa	string
		status	int
3.	Pendidikan	id_pendidikan	int
		pendidikan	string
		status	int
4	Pekerjaan	id_pekerjaan	int
		pekerjaan	string
		status	Int
5.	Angkatanpsb	id_angkatan	Int
		angkatan	String
6.	Agama	id_agama	Int
		agama	String
		status	Int
7.	TingkatPenempatan	id_tingkat	Int
		tingkat	String
8.	Suku	id_suku	Int
		suku	String
		status	Int
9.	KelasPenempatan	id_kelas	Int
		kelas	string

c. Proses Bisnis Guru Pelajaran

Web service PSB berasal dari proses bisnis PSB yang memiliki berbagai *element* dari 17 *method*, yaitu *DasarPenilaian*, *AturanGradingTabel*,

AspekPenilaianTabel, *MapelAll*, *MapelGuruAktif*, *DataGuruPerPelajaran*, *DataGuruPelajaran*, *DataGuruByPelajaran*, *AspekPerhitunganTabel*, *JenisUjian*, *AspekNilaiAll*, *StatusGuruAll*, *MapelAturanGradingNilai*, *GetGuruPelajaran-Login*, *TabelUpdateGrading*, *DataGuru*, dan *DataGuruAktif*.

Link: localhost/ws/_akademik_eko/3-guru-pelajaran/guruView.php?wsdl

Tabel 3.3 Hasil *mapping web service* guru_pelajaran

No	*Operation Name/Method*	*Element*	
		Name	*Type*
1.	DasarPenilaian	id_dasarpenilaian	int
		dasarpenilaian	string
		keterangan	string
		id_tingkat	int
2.	AturanGradingTabel	id_dasarpenilaian	int
		id_aturangrading	int
		id_tingkat	int
		nmin	double
		nmax	double
		grade	string
3.	AspekPenilaianTabel	id_dasarpenilaian	int
		id_aturanhb	int
		jenisujian	string
		id_tingkat	int
		bobot	int
		aktif	int
4	MapelAll	id_departemen	int
		id_pelajaran	int
		keterangan	string
		kode	int
		nama	string
		sifat	int
		aktif	int
5.	MapelGuruAktif	id_departemen	int
		id_pelajaran	int
		nama	string
		aktif	int
6.	DataGuruPerPelajaran	id_statusguru	int
		id_pelajaran	int
		keterangan	id_guru
		id_guru	int

No	*Operation Name/Method*	*Element*	
		Name	***Name***
7.	DataGuruPelajaran	id_statusguru	int
		id_pelajaran	int
		keterangan	string
		id_guru	int
		nip	string
		status	string
8.	DataGuruByPelajaran	id_statusguru	int
		id_pelajaran	int
		keterangan	string
		id_guru	int
		nip	string
9.	AspekPerhitunganTabel	id_aturanhb	int
		id_jenisujian	int
		jenisujian	string
		bobot	int
10.	JenisUjian	id_jenisujian	int
		id_pelajaran	int
		keterangan	string
		jenisujian	string
		info1	string
11.	AspekNilaiAll	id_dasarpenilaian	int
		dasarpenilaian	string
		keterangan	string
12.	StatusGuruAll	id_statusguru	int
		keterangan	string
		status	string
13.	MapelAturanGradingNilai	id_pelajaran	int
		nip	string
14.	GetGuruPelajaranLogin	id_pelajaran	int
		nip	string
15.	TabelUpdateGrading	nmin	double
		nmax	double
		grade	string
16.	DataGuru	nip	string
17.	DataGuruAktif	nip	string

d. Proses Bisnis Jadwal Kalender

Web service PSB berasal dari proses bisnis PSB yang memiliki berbagai *element* dari 17 *method,* yaitu Jadwal, JamBelajarAll, KelasAll,

LoadPelajaranJadwalKelas, LoadPelajaranJadwalGuru, RekapJadwalGuru, InfoJadwal, LoadJamJadwalGuru, TingkatAll,

Link:

http://localhost/ws/_akademik_eko/4-jadwal-kalender/jadwalView.php?wsdl

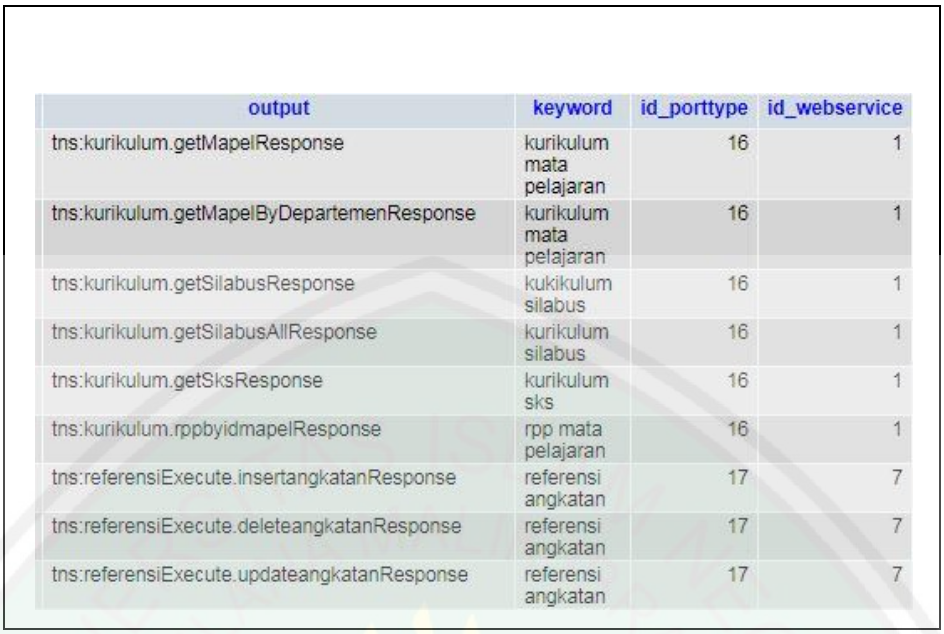
Tabel 3.4 Hasil *mapping web service* jadwal_kalender

No.	*Operation Name/Method*	*Element*	
		Name	***Type***
1.	Jadwal	Id_departmen	Int
		Id_statusguru	Int
		Id_infojadwal	Int
		Id_pelajaran	Int
		Jumlah_jam	Int
		Keterangan	string
		Id_jadwal	Int
		Id_kelas	Int
		nipguru	int
		Hari	Int
		Jam	Int
		Njam	Int
		Sifat	Int
		Jam1	String
		Jam2	string
		Id_jam1	Int
		Id_jam2	int
2.	JamBelajarAll	Id_departmen	Int
		menitmulai	String
		menitakhir	String
		jammulai	String
		Id_jam	String
		jamke	string
		jamakhir	string
3.	KelasAll	Id_tahunajaran	Int
		kapasitas	Int
		Id_kelas	int
		kelas	string
4.	LoadPelajaranJadwalGuru	Id_pelajaran	Int
		id	Int
		hari	Int
		jam	Int
		njam	Int
		ket	String
		kelas	string

No.	*Operation Name/Method*	*Element*	
		Name	***Type***
5.	LoadPelajaranJadwalKelas	Id_pelajaran	Int
		Id_jadwal	Int
		nip	String
		Hari	Int
		jam	int
		njam	Int
		ket	String
		status	string
6.	RekapJadwalGuru	nipguru	String
		asistensi	Int
		njam	Int
		hari	Int
		mengajar	Int
		tambahan	int
		Id_kelas	int
7.	InfoJadwal	Id_infojadwal	int
		deskripsi	string
		aktif	int
8.	LoadJamJadwalGuru	jamselesai	String
		jammulai	string
		jamke	string
9.	TingkatAll	Id_tingkat	int
		tingkat	string
		aktif	int

id_operation	nama_operation	input
141	kurikulum.getMapel	tns:kurikulum.getMapelRequest
142	kurikulum.getMapelByDepartemen	tns:kurikulum.getMapelByDepartemenRequest
143	kurikulum.getSilabus	tns:kurikulum.getSilabusRequest
144	kurikulum.getSilabusAll	tns:kurikulum.getSilabusAllRequest
145	kurikulum.getSks	tns:kurikulum.getSksRequest
146	kurikulum.rppbyidmapel	tns:kurikulum.rppbyidmapelRequest
147	referensiExecute.insertangkatan	tns:referensiExecute.insertangkatanRequest
148	referensiExecute.deleteangkatan	tns:referensiExecute.deleteangkatanRequest
149	referensiExecute.updateangkatan	tns:referensiExecute.updateangkatanRequest

Gambar 3.5 Hasil *parsing* WSDL ke database



output	keyword	id_porttype	id_webservice
tns:kurikulum.getMapelResponse	kurikulum mata pelajaran	16	1
tns:kurikulum.getMapelByDepartemenResponse	kurikulum mata pelajaran	16	1
tns:kurikulum.getSilabusResponse	kukikulum silabus	16	1
tns:kurikulum.getSilabusAllResponse	kurikulum silabus	16	1
tns:kurikulum.getSksResponse	kurikulum sks	16	1
tns:kurikulum.rppbyidmapelResponse	rpp mata pelajaran	16	1
tns:referensiExecute.insertangkatanResponse	referensi angkatan	17	7
tns:referensiExecute.deleteangkatanResponse	referensi angkatan	17	7
tns:referensiExecute.updateangkatanResponse	referensi angkatan	17	7

Gambar 3 6 Hasil *parsing* WSDL ke database

Sebagai contoh terdapat 3 dokumen dimisalkan D1, D2, dan D3. Seluruh dokumen berhubungan dengan proses bisnis referensi dan difokuskan untuk mendapatkan nilai *similarity*.

Tabel 3.5 Hasil *parsing* dokumen

Dokumen	**Isi Dokumen *Web Service***
D1	Referensi angkatan.
D2	Referensi angkatan yang aktif.
D3	Referensi semester yang aktif.

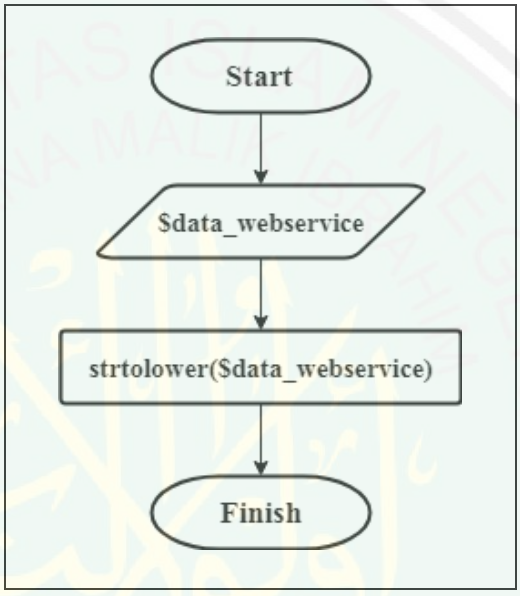
Selanjutnya, dilakukan tahap *preprocessing* terhadap *query*, yaitu angkatan aktif, setelah diolah menjadi seperti di bawah ini:

Tabel 3.6 Hasil *parsing query*

Query	**Isi *Query***
Q	referensi angkat aktif

3.5 *Case Folding*

Tahapan *preprocessing* dimulai dari tahap *case folding* dari meta dokumen *web service*, yaitu mengolah dokumen dengan mengubah semua huruf dalam dokumen menjadi huruf kecil (*lowercase*). Dalam tahap ini juga karakter selain huruf dihilangkan.



Gambar 3.7 *Flowchart code case folding*

Source code pada tahap *preprocessing* yaitu *case folding* diilustrasikan pada gambar 3.8 :

```
include "koneksi.php";

function preprocessing($data_webservice){
    $listtanda = array(".",",","-",":",";","?","!","(",")
    foreach ($listtanda as $i => $value) {
        $data_webservice = str_replace($listtanda, "
    }

    $data_webservice = strtolower($data_webservice);
```

Gambar 3.8 *Source code case folding*

Hasil dari proses pengolahan dokumen pada tahap *case folding* dapat dilihat pada tabel 3.7:

Tabel 3.7 Hasil *case folding dokumen*

Dokumen	**Isi Dokumen *Web Service***
D1	referensi angkatan
D2	referensi angkatan yang aktif
D3	referensi semester aktif

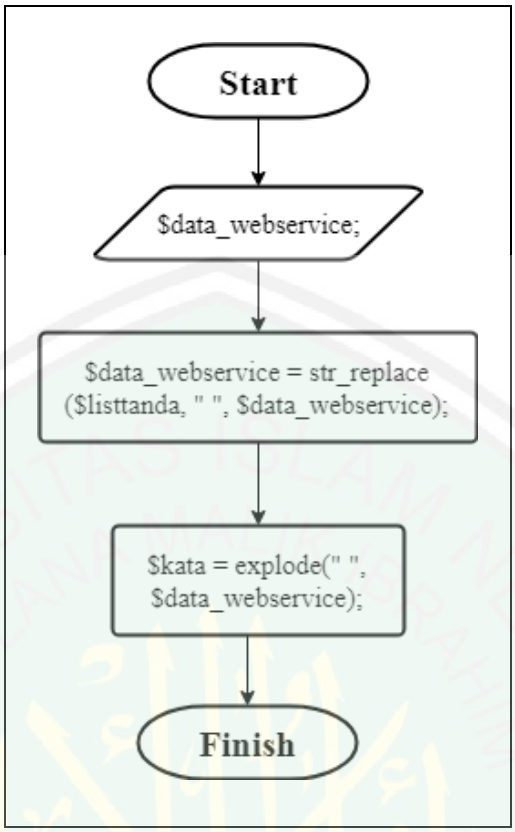
Selanjutnya, dilakukan tahap *preprocessing* terhadap *query*, yaitu angkatan aktif, setelah diolah menjadi seperti tabel 3.8:

Tabel 3.8 Hasil *case folding query*

Query	**Isi *Query***
Q	referensi angkat aktif

3.6 *Tokenizing*

Pada proses *preprocessing* setelah melakukan *case folding* adalah *tokenizing*. Yakni memotong tiap kata dalam kalimat atau *parsing* dengan menggunakan spasi sebagai *delimiter* yang akan menghasilkan token berupa kata. Jadi, pada tahap ini menghilangkan tanda baca dan memecah tiap kata (belum dibobotkan). Selain itu melakukan pemotongan *query* yang akan di-*input*-kan menjadi kata-kata tunggal berdasarkan spasi, seperti pada gambar 3.9. Pada gambar 3.9, *input* berupa kata yang terdapat di dalam dokumen web service, kemudian kata tersebut dipotong menggunakan spasi sehingga beberapa kata yang terdapat pada dokumen *web service* menjadi kata-kata tunggal. Pemotongan kata menggunakan fungsi *explode*.



Gambar 3.9 *Flowchart tokenizing*

Tahap *tokenizing* diimplementasikan pada program, sebagai berikut:

```
//List tanda baca
$listtanda = array(".",",","-",":",";","?","!","(",")");

//Menghilangkan tanda baca
foreach ($listtanda as $i => $value) {
    $data_webservice = str_replace($listtanda, " ", $data_webservice);
}

//Memotong kata
$kata = explode(" ", $data_webservice);

//print_r($kata); Menghitung Jumlah Kata
$jml_kata = count($kata)-1;
for ($i=0; $i<=$jml_kata; $i++){
    if (in_array($kata[$i], $stoplist)) {
    // print_r($kata);
        unset($kata[$i]);
    }
    // print_r($kata);
}
```

Gambar 3.10 *Source code tokenizing*

Setelah dilakukan proses *case folding* (mengubah menjadi huruf kecil) maka dilanjutkan proses *tokenizing* untuk menghilangkan tanda baca yang mengacu pada deklarasi *list* tanda baca. Selanjutnya, dokumen akan dipecah dari kalimat menjadi per kata. Begitu juga *query*, diproses tokenisasi seperti dokumen. Hasil tokenisasi dapat dilihat pada tabel:

No.	*Query*
1.	referensi
2.	angkatan
3.	yang
4.	aktif
5.	semester

Tabel 3.9 Hasil *tokenizing* dokumen

No	*Query*
1.	referensi
2.	angkat
3.	aktif

Tabel 3.10 Hasil *tokenizing* dokumen

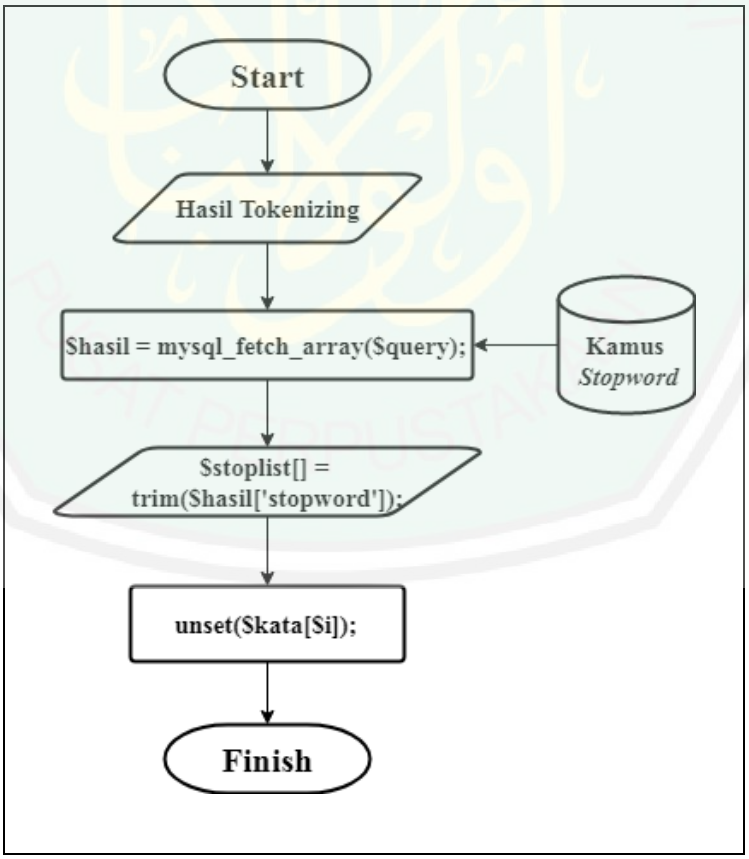
3.7 *Stopword Removal*

Pada proses *stopword removal* yaitu menyaring kata yang didapat dari proses *tokenizing* yang dianggap tidak penting atau tidak memiliki makna. Tiap kata yang diperoleh dari *tokenizing* akan dicocokkan dalam kamus *stopword* di dalam *database*, jika kata tersebut cocok dengan salah satu kata dalam *stopword* maka kata tersebut akan dihilangkan, sementara yang tidak cocok akan dianggap cocok dan diproses ke tahap selanjutnya.

Tabel 3.11 adalah isi dari kamus *stopword*. Kamus *stopword* tersebut terdiri dari kata baku Bahasa Indonesia yang tidak penting atau tidak memiliki makna.

Tabel 3.11 Kamus *stopword*

Stopword	
yang	sudah
mampu	tetapi
tentang	oleh
di	bisa
setelah	tidak
semua	sayang
hampir	melakukannya
juga	lakukan
am	memang
antara	baik
dan	lain
ada	pernah
seperti	setiap
jadi	untuk
karena	dari



Gambar 3.12 *Flowchart stopword removal*

```php
//panggil tabel stopword
$query = mysql_query("SELECT * FROM tb_stopwords");

//memanggil stopword
while($hasil = mysql_fetch_array($query)){
    $stoplist[] = trim($hasil['stopword']);
}
//memisah per kata
$kata = explode(" ", $data_webservice);
//print_r($kata); Menghitung jumlah kata
$jml_kata = count($kata)-1;
//menghapus kata yang sama dengan stopword
for ($i=0; $i<=$jml_kata; $i++){
    if (in_array($kata[$i], $stoplist)) {
    // print_r($kata);
        unset($kata[$i]);
    }
    // print_r($kata);
}
//menggabungkan per kata
$data_webservice = implode(" ",$kata);
$data_webservice = strtolower(trim($data_webservice));

return $data_webservice;
```

Gambar 3.13 *Source code stopword removal*

Pada tabel 3.11 dan 3.12 merupakan hasil proses *stopword removal*:

Tabel 3.11 hasil *stopword removal* dokumen

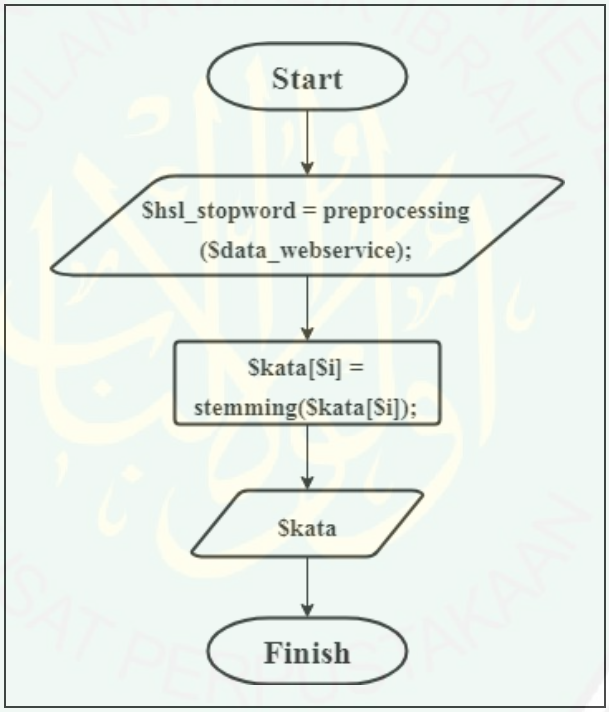
No	*Query*
1.	referensi
2.	angkatan
3.	aktif
4.	semester

Tabel 3.12 hasil *stopword removal query*

No	*Query*
1.	referensi
2.	angkat
3.	aktif

3.8 *Stemming*

Proses *stemming* bertujuan untuk menghasilkan keluaran berupa *root word* (kata dasar). Dengan cara mengembalikan kata-kata yang diperoleh dari hasil proses sebelumnya yaitu *stopword removal* ke bentuk dasarnya, menghilangkan imbuhan awal (*prefix*) dan imbuhan akhir (*sufix*) sehingga di dapat kata dasar. Ini merupakan tahapan terakhir dari *preprocessing*. Berikut *flowchart* dari proses *stemming*:



Gambar 3.14 *Flowchart stemming*

Proses *stemming* seperti pada gambar 3.14, menggunakan *input* dari hasil *stopword* kemudian akan di proses *stemming* untuk mendapatkan kata dasar sehingga diperoleh *output* berupa kata dasar. Hasil *stemming* ini akan diberikan bobot nilai per kata pada tahap berikutnya yaitu pembobotan TF.IDF.

Berikut implementasi proses *stemming* ke dalam *source code*:

```php
function cari($data_webservice){
    include "koneksi.php";
    $hasil = mysql_num_rows(mysql_query("SELECT * FROM
    tb_katadasar WHERE katadasar='$data_webservice'"));
    return $hasil;
}

//langkah 1 - hapus partikel
function hapuspartikel($data_webservice){
if(cari($data_webservice)!=1){
    if((substr($data_webservice, -3) == 'kah' )||
    ( substr($data_webservice, -3) == 'lah' )||
    ( substr($data_webservice, -3) == 'pun' )){
        $data_webservice = substr($data_webservice, 0, -3);
        }
    }
    return $data_webservice;
}

//langkah 2 - hapus possesive pronoun
function hapuspp($data_webservice){
if(cari($data_webservice)!=1){
    if(strlen($data_webservice) > 4){
    if((substr($data_webservice, -2)== 'ku')||
    (substr($data_webservice, -2)== 'mu')){
        $data_webservice = substr($data_webservice, 0, -2);
    }else if((substr($data_webservice, -3)== 'nya')){
        $data_webservice = substr($data_webservice,0, -3);
    }
  }
}
    return $data_webservice;
}
```

Gambar 3.15 *Source code stemming*

Setelah kata diolah menjadi kata dasar pada prosess *stemming*, maka hasilnya adalah sebagai berikut ini:

Tabel 3.13 Hasil *stemming* dokumen

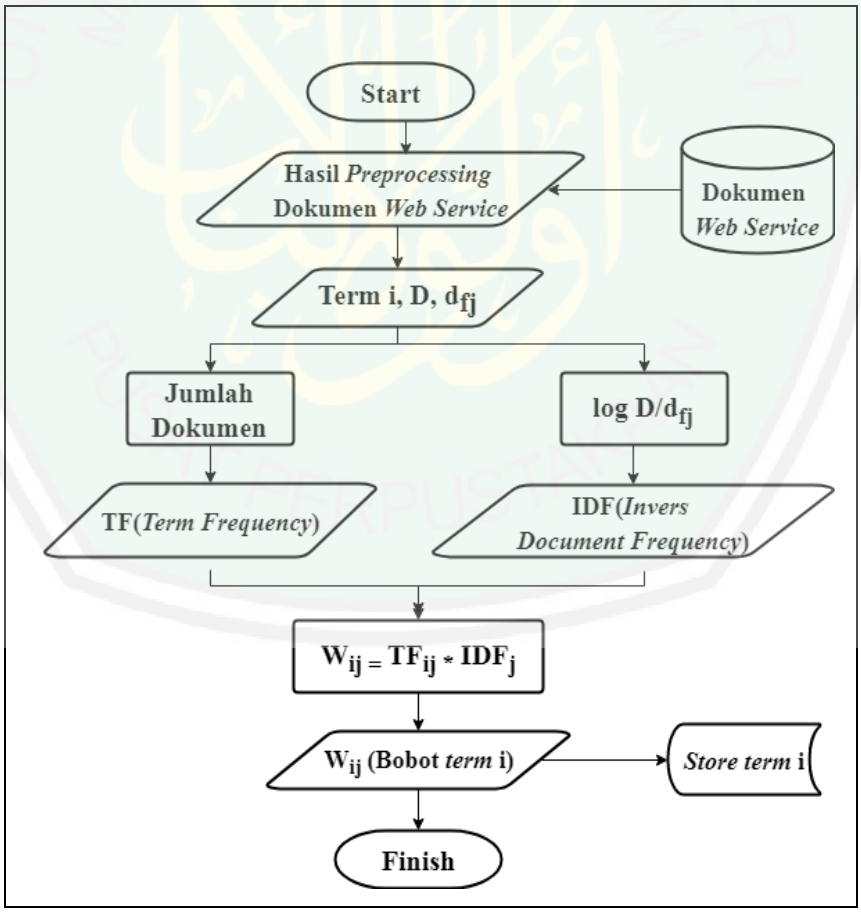
1.	referensi
2.	angkat
4.	aktif
5.	semester

Tabel 3.14 Hasil *stemming query*

1.	referensi
2.	angkat
3.	aktif

3.9 TF/IDF

Pada tahap ini dilakukan analisa terkait dokumen *web service* dengan *query* yang di-*input*-kan *user*. Peneliti menggunakan faktor pembobotan untuk tiap kata (*term*) dalam dokumen didefinisikan sebagai kombinasi *term frequency* dan *inverse document frequency*.



Gambar 3.16 *Flowchart* pembobotan TF/IDF

TF/IDF digunakan untuk menghitung nilai bobot kata. Bobot kata i W_{ij} diformulakan menggunakan rumus 3.1.

$$W_{ij} = TF_{ij} \text{ X } IDF_j \qquad (3.1)$$

Berikut *flowchart* pembobotan dokumen *web service* menggunakan TF/IDF pada gambar 3.16:

Selanjutnya *source code* untuk menghitung bobot kata pada dokumen dan *query* yaitu *term frequency* (TF) dan *inverse document frequency* (IDF).

Tabel 3.15 *Source code* pembobotan TF/IDF

```
function indexData(){
    //hapus index sebelumnya
    mysql_query("TRUNCATE TABLE tb_index_opt");
    //ambil semua berita (teks)
    $sql_opt = mysql_query("SELECT * FROM tb_operation ORDER BY
    id_operation");
    $data = mysql_num_rows($sql_opt);
    print("Mengindeks sebanyak " . $data . " berita. <br />");

    while($row = mysql_fetch_array($sql_opt)) {
        $id_opt = $row['id_operation'];
        $data_webservice = $row['keyword'];
        //terapkan preprocessing
        $data_webservice = kataDasar($data_webservice);

        //simpan ke inverted index (tb_index_opt)
        $opt = explode(" ", trim($data_webservice));
        foreach ($opt as $j => $value) {
        //hanya jika Term tidak null atau nil, tidak kosong
        if ($opt[$j] != "") {
        //berapa baris hasil yang dikembalikan query tersebut?
        $hitung_term=mysql_query("SELECT count_keyword FROM
tb_index_opt WHERE term_keyword = '$opt[$j]' AND id_operation =
$id_opt");
        $jml_term = mysql_num_rows($hitung_term);
        //jika sudah ada id_operation dan Term tersebut , naikkan
```

```
          // Count (+1)
          if ($jml_term > 0) {
          $data_term = mysql_fetch_array($hitung_term);
          $count = $data_term['count_keyword'];
          $count++;
          mysql_query("UPDATE tb_index_opt SET count_keyword = $count
WHERE
           term_keyword = '$opt[$j]' AND id_operation = $id_opt");
         }
          //jika belum ada, langsung simpan ke tb_index_opt
          else {
          mysql_query("INSERT                                    INTO
tb_index_opt(id_index_opt,id_operation,

term_keyword,count_keyword)VALUES(NULL,'$id_opt','$opt[$j]',1)");}}}}
}
```

Tabel 3.16 *Source code* perhitungan bobot

```
function hitungBobot(){
     //berapa jumlah id_operation total?, n
       $index=mysql_query("SELECT DISTINCT id_operation FROM
     tb_index_opt");
     $n = mysql_num_rows($index);
    //ambil setiap record dalam tabel tb_index_opt
    //hitung bobot untuk setiap Term dalam setiap id_operation
    $sql_index = mysql_query("SELECT * FROM tb_index_opt ORDER BY
    id_index_opt");
    $hasil = mysql_num_rows($sql_index);
    print("Terdapat " . $hasil . " Term yang diberikan bobot. <br />");
    while($dt_index = mysql_fetch_array($index)) {
    //$w = tf * log (n/N)
    $term = $dt_index['term_keyword'];
    $tf = $dt_index['count_keyword'];
    $id = $dt_index['id_index_opt'];
    //berapa jumlah dokumen yang mengandung term itu?, N
    $hitung_term = mysql_query("SELECT Count(*) as N FROM tb_index_opt
    WHERE term_keyword = '$term'");
    $hasil_term = mysql_fetch_array($hitung_term);
    $NTerm = $hasil_term['N'];
    $w = $tf * log($n/$NTerm);
    //update bobot dari term tersebut
    $updateBobot = mysql_query("UPDATE tb_index_opt SET bobot_keyword
=
    $w WHERE id_index_opt = $id");
    } //end while $rowbobot}
```

Perhitungan TF/IDF untuk mendapatkan bobot dari hasil *processing* dimana n adalah jumlah seluruh dokumen yaitu 3 dokumen, seperti pada tabel 3.17:

Tabel 3.17 Menghitung TF/IDF

Term	**Q**	**D1**	**D2**	**D3**	**df**	**n/df**	**Idf=log(n/df)**
referensi	1	1	1	1	3	1	**0**
angkat	1	1	1		2	1.5	**0.17609**
aktif	1		1	1	2	1.5	**0.17609**
semester				1	1	3	**0.47712**

Kemudian, setelah menghitung TF/IDF nilai-nilai tersebut dimasukkan ke dokumen dan *query*. Selanjutnya dihitung bobotnya seperti pada tabel 3.18:

Tabel 3.18 Menghitung bobot

		W			
Term	**Idf=log(n/df)**	**Q**	**D1**	**D2**	**D3**
referensi	0	0	0	0	0
angkat	0.17609	0.17609	0.17609	0.17609	0
aktif	0.17609	0.17609	0	0.17609	0.17609
semester	0.47712	0	0	0	0.47712

Setelah dihitung bobot dokumen dan *query*, dilanjutkan dengan menghitung *similarity* antara *query* dengan dokumen. Beberapa persamaan untuk membandingkan *vector query* dengan *vector* dokumen telah dikembangkan yakni *Cosine coefficient*. Dengan W adalah bobot dari *query* dan dokumen.

Tabel 3.19 Hasil perhitungan *cosine similarity*

	Q	**D1**	**D2**	**D3**
Jumlah Kuadrat Bobot	0.062015	0.031008	0.062015	0.258651
Panjang Vektor	0.249029	0.17609	0.249029	0.508578
Jumlah Perkalian Bobot *Query* dengan Bobot Dokumen		0.031008	0.062015	0.031008
Cosine Similarity		0.707107	1	0.244829

Nilai *cosine* dihitung dengan persamaan berikut:

Cosine (D_i) = sum($W_{ij} \times W_{ik}$) / sqrt(W_{ij}^2) × sqrt(W_{ik}^2)

Kemudian untuk mendapatkan hasil dari perhitungan *cosine similarity* dimasukan ke dalam hasil dari persamaan berikut:

$\text{Cos}\ (q, d1) = \frac{0.031008}{0.249029 * 0.17609} = 0.707107$
$\text{Cos}\ (q, d2) = \frac{0.062015}{0.249029 * 0.249029} = 1$
$\text{Cos}\ (q, d3) = \frac{0.031008}{0.249029 * 0.508578} = 0.244829$

Setelah mendapatkan nilai *cosine similarity* tiap-tiap dokumen, maka hasil bobot *query* diurutkan. Bobot yang besar menjadi prioritas sebagai dokumen yang memiliki hubungan dengan *query*, yaitu terurut menjadi D2, D1, dan D3.

Tabel 3.20 Hasil *similarity* dokumen

Dokumen Terangking	**Isi Dokumen *Web Service***	**Hasil *Preprocessing***
D2	referensi angkatan yang aktif	referensi angkat aktif
D1	referensi angkatan	referensi angkat
D3	referensi semester aktif	referensi aktif

Dokumen diatas merupakan hasil dari pencarian yang sudah terangking berdasarkan *query* yang dimasukan pada tabel di bawah ini:

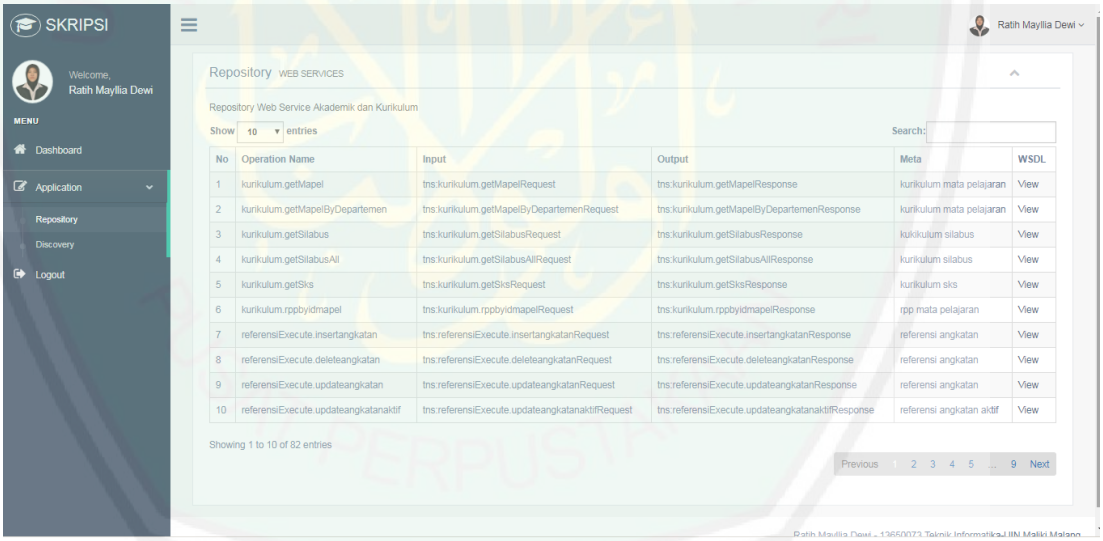
Tabel 3.21 *Query*

Query	**Isi *Query***
Q	referensi angkat aktif

3.10 Aplikasi *Web Service Discovery*

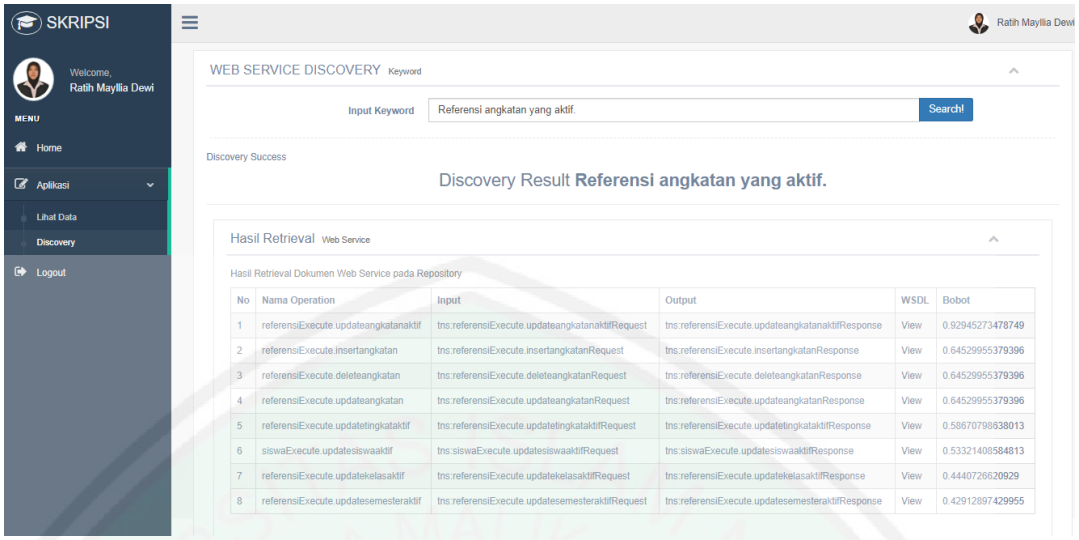
Perancangan sistem aplikasi *web service discovery* memiliki output berupa dokumen *web service* yang terangking berdasarkan nilai *similarity*-nya. Desain interface untuk aplikasi ini memiliki menu utama yaitu menyajikan isi *repository* dan menu *discovery* untuk menemukan *web service*. Berikut merupakan tampilan aplikasi *web service discovery* dan bagaimana pengoperasiannya:

1. Menu ***Application***, memiliki 2 sub menu yaitu yang pertama menu ***Repository***. Menu ***Repository*** menyajikan seluruh data *web service* dalam bentuk tabel yang terdiri dari kolom *Operation Name*, *Input*, *Output*, *Meta*, dan *file* WSDL.



Gambar 3.17 GUI menu *repository*

2. Menu ***Discovery***, pada menu ini *user* meng-*input*-kan *query* untuk mendapatkan *web service* yang ingin dicari. Contoh: “Guru yang mengajar.”



Gambar 3.18 *GUI menu discovery*

3.11 Pengujian Sistem

Akurasi adalah *weight arithmetic mean* dari *Precision* dan *Inverse Precision* (dibobot berdasarkan Bias) serta *weight arithmetic mean* dari *Recall* dan *Inverse Recall* (dibobot berdasarkan Prevalensi). *Inverse Precision* dan *Recall* (Powers, 2011) hanyalah *Precision* dan *Recall* dari masalah *invers* dimana label positif dan negatif dipertukarkan (untuk *real classes* dan *prediction label*). *Recall* dan *Inverse Recall*, dapat disebut sebagai *true positif rate* dan *false positif rate*.

Sistem yang dibuat akan diuji untuk diketahui tangkat keakuratan atau akurasi *query*-nya. Adapun bentuk pengujiannya yaitu menggunakan ukuran akurasi. Jadi, pengujian akurasi dihitung menggunakan teori *accuracy* untuk mengetahui tingkat akurasi *query* pada *web service discovery*. Pada tabel 3.22, diberikan konsep mengenai perhitungan akurasi berdasarkan empat parameter, yaiut TP, FP, TN, dan FN.

Berikut ini adalah tabel aturan yang digunakan untuk mengukur nilai akurasi:

Outcome of query results	*Condition* (*e.g discovery web service*) *as determined by the Standard of Truth*		
	Positive	*Negative*	Row Total
Positive	TP	FP	TP+FP *(Total number of subjects with positive test)*
Negative	FN	TN	*FN + TN (Total number of subjects with negative test)*
Column total	TP+FN (*Total number of subjects with given condition*)	FP+TN (*Total number of subjects without given condition*)	N = TP+TN+FP+FN (*Total number of subjects in study*)

Tabel 3.22 Aturan mengukur akurasi

Di luar *information retrieval*, penerapan *Recall*, *Precision* dan F-*measure* dianggap cacat karena mengabaikan sel negatif tabel kontinjensi yang sebenarnya, dan mudah dimanipulasi dengan *biasing the preditcions*. Masalah ini (Powers, 2011) dapat dipecahkan dengan menggunakan Akurasi.

Untuk mengevaluasi akurasi *query* yang diusulkan, maka akan digunakan pengujian sistem menggunakan ukuran akurasi.

Terdapat beberapa aturan yang umum digunakan dalam menghitung akurasi, yaitu *true positive* (TP), *true negative (*TN), *false negative* (FN), dan *false positive* (FP). Jika, hasil prediksi *query* bernilai *positive* dan nilai sebenarnya juga *positive*, maka dapat disebut *true positive* (TP), sedangkan jika nilai sebenarnya adalah *negative* maka disebut *false positive* (FP). Disisi lain, *true negative* (TN) dikatakan jika prediksi dan nilai sebenarnya adalah *negative*, dan *false negative* (FN) adalah jika hasil prediksi *negative* sementara nilai sebenarnya *positive* (Zhu, 2010).

Dibawah adalah formula untuk menghitung nilai akurasi yang mengacu pada ketentuan tabel berikut:

Tabel 3.23 Rumus akurasi

Accuracy = (TN + TP)/(TN+TP+FN+FP) = (*Number of correct assessments*)/*Number of all assessments*)

3.12 Evaluasi

Evaluasi dibutuhkan untuk menganalisis hasil pengujian yang telah dibangun sehingga dapat diketahui tingkat akurasinya. Pengujian dilakukan dengan menganalisa nilai akurasinya.

3.13 Dokumentasi

Dokumentasi disertakan selama proses penyusunan laporan hasil penelitian. Dokumentasi ini berupa laporan penelitian.

BAB 4

HASIL DAN PEMBAHASAN

Dalam bab ini akan dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibangun. Uji coba dilakukan untuk mengetahui apakah sistem telah berjalan sebagaimana mestinya sesuai dengan tahapan penelitian yang telah dilakukan pada bab 3.

4.1 Prosedur Pengujian

Prosedur pengujian dilakukan dengan mengimplementasikan metode yang digunakan dalam penelitian *web service discovery* berdasarkan sistematika yang sudah peneliti susun sehingga diperoleh hasil uji coba yang terukur tingkat akurasinya. Adapun langkah-langkah dalam pengujian sistem yang dilakukan untuk mengetahui tingkat akurasinya, adalah sebagai berikut:

4.1.1 Pengindeksan

Pengindeksan dokumen *web service* dilakukan dalam beberapa tahapan. Tahap pertama yaitu mengambil data indeks *web service* dari hasil *parsing* dekomposisi *web service* yang selanjutnya tersimpan di *repository*. Tahap selanjutnya adalah *preprocessing*. *Preprocessing* dilakukan melalui empat tahapan yaitu, *case folding*, tokenisasi, *stopword*, dan *stemming*. Hasil *preprocessing* adalah *term* atau *meta* dari seluruh dokumen *web service* yang terkait. Setelah itu dilakukan tahap pembobotan. Pembobotan menggunakan TF.IDF terhadap masing-masing *term web service* tersebut.

4.1.2 Pemanggilan Dokumen *Web Service* dari *Repository*

Dokumen *web service* yang digunakan sebagai data pada penelitian ini tersimpan di dalam *repository web service*. *Repository* tersimpan di dalam MysQL terdiri dari beberapa tabel yang diproses pada yaitu tabel *operation, index operation, vector, cache, stopword*, dan kata dasar. Untuk memanggil data dari *repository* tersebut dibutuhkan koneksi php dengan nama file koneksi.php. *File* koneksi ini terhubung di setiap proses fungsi PHP, seperti fungsi *select, insert, update,* dan *delete*.

Semua fungsi menggunakan bahasa pemrograman PHP. PHP adalah singkatan dari *Hypertext Prepocessor*, digunakan secara luas untuk penanganan pembuatan dan pengembangan aplikasi berbasis situs web dan digunakan bersamaan dengan HTML.

Di dalam file koneksi terdapat atribut connect untuk menjalankan query pada setiap fungsi manajemen basis data. Beberapa method pada koneksi yang sering digunakan untuk memproses database yaitu, method new mysqli(), digunakan untuk membuat sebuah koneksi. Method mysqli_query() digunakan untuk memproses perintah pada database MySQL seperti *select, update*, *insert*, dan *delete*. *Method mysqli_num_rows()* digunakan untuk menghitung jumlah data yang dipanggil. *Method mysqli_fetch_array()* digunakan untuk menyajikan data yang sudah dipanggil dari basis data. Pada *file* koneksi terdapat beberapa parameter yaitu alamat *host*, nama *database*, *username* dan *password*.

4.1.3 *Preprocessing*

Dokumen *web service* sebelum dibuat indeks datanya diproses dahulu melalui tahapan *preprocessing* untuk menghasilkan sebuah *set term index* yang bisa mewakili dokumen. Proses preprocessing dilakukan melalui beberapa tahapan yaitu, *case folding*, *tokenizing*, *stopword* dan *stemming*. Berikut adalah proses rinci pada preprocessing:

a. *Case Folding*

Mengubah semua huruf dalam dokumen menjadi huruf kecil (*lowercase*). Dalam tahap ini juga karakter selain huruf dihilangkan.

b. *Tokenizing*

Memotong tiap kata dalam kalimat atau *parsing* dengan menggunakan spasi sebagai delimiter yang akan menghasilkan token berupa kata.

c. *Filtering* (*Stopword*)

Menyaring kata yang didapat dari proses *tokenizing* yang dianggap tidak penting atau tidak memiliki makna dalam proses *text mining* yang disebut *stoplist*. *Stoplist* atau *stopword* berisi kata-kata umum yang sering muncul dalam sebuah dokumen dalam jumlah banyak namun tidak memiliki kaitan dengan dengan tema tertentu. Tiap kata yang diperoleh dari *tokenizing* akan dicocokkan dalam kamus *stopword* di dalam *database*, jika kata tersebut cocok dengan salah satu kata dalam *stopword* maka kata tersebut akan dihilangkan, sementara yang tidak cocok akan dianggap cocok dan diproses ke tahap selanjutnya.

d. *Stemming*

Mengembalikan kata-kata yang diperoleh dari hasil *filtering* ke bentuk dasarnya, menghilangkan imbuhan awal (*prefix*) dan imbuhan akhir (*sufix*) sehingga di dapat kata dasar. Metode *stemming* memerlukan masukan berupa kata yang terdapat dalam suatu dokumen, dengan menghasilkan keluaran berupa *root word*.

Proses *preprocessing* diatas tersebut hasilnya disimpan ke dalam *repository* dalam bentuk database MySQL seperti pada gambar 4.1.

id_index_opt	id_operation	term_keyword
19	150	referensi
20	150	angkat
21	150	aktif

Gambar 4.1 Hasil *preprocessing*

4.1.4 Pembobotan TF.IDF

Pembobotan TF.IDF merupakan tahapan setelah *preprocessing*. Pada tahap ini dilakukan analisa terkait dokumen *web service* dengan *query* yang di-*input*-kan *user*. Peneliti menggunakan faktor pembobotan untuk tiap kata (*term*) dalam dokumen didefinisikan sebagai kombinasi *term frequency* dan *inverse document frequency*. TF/IDF digunakan untuk menghitung nilai bobot kata. Hasil dari pembobotan disimpan pada *repository* pada tabel *index_opt* di database MySQL seperti gambar 4.2:

id_index_opt	id_operation	term_keyword	count_keyword
19	150	referensi	1
20	150	angkat	1
21	150	aktif	1

Gambar 4.2 Jumlah nilai per *term_keyword*

id_index_opt	id_operation	term_keyword	count_keyword	bobot_keyword
19	150	referensi	1	1.63413
20	150	angkat	1	3.02042
21	150	aktif	1	2.79728

Gambar 4.3 Jumlah nilai bobot per *term_keyword*

Diambil setiap *record* dalam tabel *tb_index_opt*. Setelah diketahui nilai *ter_keyword* yaitu *count_keyword* pada setiap dokumen (*id_operation*) *web service* selanjutnya dihitung bobot untuk setiap *term_keyword* pada setiap dokumen (*id_operation*). Dan hasilnya disimpan pada *bobot_keyword*.

4.1.5 Implementasi *Cosine Similarity*

Discovery terhadap *query* dihitung menggunakan *cosine similarity* antara *query* dengan seluruh dokumen *web service*. Perhitugngan *cosine similarity* mengacu pada nilai pembobotan TF.IDF. Kemudian, untuk menghitung nilai *similarity* dilakukan dengan cara membandingkan nilai kemiripan antara nilai *vector query* dan *vector* tiap dokumen pada *repository web service*. VSM menghitung kemiripan dengan mendefinisikan sebuah *vector* yang mempresentasikan tiap dokumen *web service*, dan sebuah *vector* yang mempresentasikan *query* dari *keyword* untuk menemukan *web service* tersebut di dalam *repository*.

Kata-kata dalam dokumen *web service* direpresentasikan dalam *vector*, sehingga sangat dimungkinkan untuk membandingkan dokumen *web service* dengan *query* untuk menunjukkan kemiripan konten keduanya. Sebuah *query* dianggap mirip dengan sebuah dokumen *web service*, maka koefesien kemiripan yang mengukur kemiripan antara dokumen *web service* dan *query*

dapat dihitung. Dokumen *web service* yang kontennya paling mirip dengan konten pada *query* dianggap paling relevan.

Pencocokan ini memerlukan pembentukan *vector* yang merepresentasikan kata-kata dalam dokumen *web service* dan *vector* lain yang merepresentasikan kata-kata dalam *query*. Kemudian, sebuah metode harus dipilih untuk mengukur kedekatan *vector* dokumen dan *vector query*, yaitu *cosine similarty*.

Sebelum menghitung nilai *similarity* menggunakan *cosine similarity* antara *query* dengan dokumen *web service*, dilakukan tahapan *preprocessing* terhadap *query* yang dimasukkan oleh *user* dan diberikan bobot tiap kata dalam *query* tersebut, sama seperti tahapan *preprocessing* pada dokumen *web service*.

Hasil *preprocessing* dan perhitungan nilai *similarity* disimpan pada tabel *tb_cache*. Misalkan *query* yang dimasukkan oleh *user* adalah: "Referensi angkatan yang aktif.". Selanjutnya kalimat tersebut akan diproses melalui beberapa tahapan dalam *preprocessing* dan *term*/kata yang dihasilkan dari proses tersebut adalah "*referens* angkat aktif" dengan masing-masing nilai *similarity*-nya yang diurutkan berdasarkan nilai *similarity* terbesar. Sehingga hasil disajikan berdasarkan dokumen *web service* yang paling relevan.

Berikut adalah hasil perhitungan *similarity* antara dokumen dengan *query* yang tersimpan pada tabel *tb_cache*. Tabel tb_cache tersimpan di database MySQL seperti pada gambar 4.4:

id	query	docId	value
1	referens angkat aktif	147	0.645299553794
2	referens angkat aktif	148	0.645299553794
3	referens angkat aktif	149	0.645299553794
4	referens angkat aktif	150	0.929452734787
5	referens angkat aktif	154	0.58670798638
6	referens angkat aktif	158	0.4291289743
7	referens angkat aktif	162	0.444072662093
8	referens angkat aktif	211	0.533214085848

Gambar 4.4 Hasil *similarity query* dengan dokumen

4.2 Hasil dan Uji Coba

Pada langkah-langkah sebelumnya telah dilakukan implementasi metode dari *preprocessing* hingga mendapatkan nilai *similarity* antara *query* dengan dokumen *web service*. Langkah-langkah tersebut telah dilakukan secara sistematis sehingga dapat diketahui nilai akurasi pada penelitan ini.

Pada tahap ini dilakukan pengujian dari penelitan yang dikerjakan. Pengujian dilakukan menggunakan *query* dari *user* atau *developer* di bidang *web service* untuk mendapatkan hasil *discovery* berupa dokumen *web service* yang relevan.

4.3 Hasil dan Analisa

Tahap pengujian aplikasi untuk mengukur akurasi query yang di-input-kan oleh user. *Query* tersebut diukur dan dilakukan analisa untuk mendapatkan nilai akurasinya sehingga diketahui nilai kemampuan aplikasi yang telah dibuat. Tabel 4.1 merupakan daftar *query* yang diuji coba untuk dihitung nilai akurasinya:

Tabel 4.1 *List query* pengujian

No.	*Query*
1	Guru yang mengajar.
2	Peraturan menghitung bobot nilai
3	bekerja
4	Tervalidasi.
5	Referensi angkatan yang aktif.
6	SISWA BARU YANG LULUS
7	aspek penilaian untuk siswa
8	Kurikulum pelajaran siswa
9	jadwal untuk pelajaran
10	kenaikan siswa
11	silabus pelajaran
12	presensi keaktifan
13	grading siswa
14	Jumlah SKS pelajaran di kelas
15	guru
16	siswa belajar
17	siswa belajar mengajar
18	hitung bobot
19	Penjadwalan
20	data guru

Dokumen-dokumen *web service* tersimpan di tabel *tb_operation* memiliki beberapa *field* dan berisi *record* dokumen web *service* seperti pada tabel 4.2 dan 4.3. Berikut adalah tabel tb_operation dokumen *web service*:

Tabel 4.2 Dokumen *web service*

No	Nama *Operation*	*Input*
1	kurikulum.getMapel	tns:kurikulum. getMapelRequest
2	kurikulum.getMapelByDepartemen	tns:kurikulum.getMapelBy DepartemenRequest
3	kurikulum.getSilabus	tns:kurikulum.getSilabus Request
4	kurikulum.getSilabusAll	tns:kurikulum.get SilabusAllRequest
5	kurikulum.getSks	tns:kurikulum.getSksRequest
6	kurikulum.rppbyidmapel	tns:kurikulum.rppby idmapelRequest

No	Nama *Operation*	*Input*
7	referensiExecute.insertangkatan	tns:referensiExecute. insertangkatanRequest
8	referensiExecute.deleteangkatan	tns:referensiExecute. deleteangkatanRequest
9	referensiExecute.updateangkatan	tns:referensiExecute. updateangkatanRequest
10	referensiExecute.updateangkatanaktif	tns:referensiExecute. updateangkatanaktifRequest

Tabel 4.3 Dokumen *web service*

Output	*Meta*	*WSDL*
tns:kurikulum. getMapelResponse	kurikulum mata pelajaran	View
tns:kurikulum.getMapel ByDepartemenResponse	kurikulum mata pelajaran	View
tns:kurikulum.getSilabusResponse	kukikulum silabus	View
tns:kurikulum.getSilabusAllResponse	kurikulum silabus	View
tns:kurikulum.getSksResponse	kurikulum sks	View
tns:kurikulum.rppbyidmapelResponse	rpp mata pelajaran	View
tns:referensiExecute. insertangkatanResponse	referensi angkatan	View
tns:referensiExecute. deleteangkatanResponse	referensi angkatan	View
tns:referensiExecute. updateangkatanResponse	referensi angkatan	View
tns:referensiExecute.update angkatanaktifResponse	referensi angkatan aktif	View

Query yang telah disiapkan pada tabel 4.3 diujikan terhadap dokumen *web service* yang telah ada di dalam *repository web service*. Seperti pada *query* pada tabel 4.4 diuji coba untuk diketahui akurasi querynya:

Tabel 4.4 *Query* pengujian

Query
Kurikulum pada pelajaran siswa.

Hasil dari *web service discovery* menggunakan *query* 4.4 ditampilkan pada tabel 4.5, dilakukan analisa dengan "nama *operation*" sebagai *parameter* hasil *discovery* dokumen *web service*. Diberikan analisa *true positive*, yaitu

dokumen *web service* yang harusnya tertangkap dan hasil *discovery*-nya pun tertangkap berdasarkan *query* yang diberikan. Nilai "*YES*" jika hasil *discovery* dokumen tersebut bernilai benar sebaliknya jika dokumen tidak relevan terhadap *query* yang diberikan maka nilainya "*NO*".

Tabel 4.5 Hasil *web service discovery*

No	Nama *Operation*	*True Positive*
1	kurikulum.getMapel	YES
2	kurikulum.getMapelByDepartemen	YES
3	kenaikanExecute.updatesiswanaik	YES
4	kenaikanExecute.updatesiswatidaknaik	YES
5	guruExecute.updatemapel	YES
6	guruExecute.deletemapel	YES
7	guruExecute.insertmapel	YES
8	kurikulum.getSilabusAll	YES
9	psbExecute.insertsiswa2	NO
10	psbExecute.insertsiswa	NO
11	psbExecute.deletesiswabaru	NO
12	kurikulum.getSks	YES
13	siswaExecute.updatesiswaaktif	YES
14	kurikulum.rppbyidmapel	YES
15	mutasiExecute.updatemutasisiswa	NO
16	presensiExecute.insertphsiswa	YES
17	kenaikanExecute.updatesiswalulus	YES

Tabel 4.6 Hasil akurasi *query*

Parameter Akurasi	Nilai
True Positive (**TP**)	13
False Positive (**FN**)	4
True Negative (**TN**)	63
False Negative (**FN**)	2
Correct Assessment (**TN**+**TP**)	76
All Assessment (**TN**+**TP**+**FN**+**FP**)	82
Accuracy (Correct Assessment/All Assessment)	**92.68%**

Keterangan:

TP: Dokumen yang harusnya tertangkap hasilnya tertangkap

FP: Dokumen yang harusnya tidak tertangkap namun tertangkap

TN: Dokumen yang harusnya tidak tertangkap tapi tidak tertangkap

FN: Dokumen yang harusnya tertangkap namun tidak tertangkap

Selanjutnya dilakukan analisa hasil pengujian terhadap *query* pada tabel 4.4. Berdasarkan hasil *discovery* pada tabel 4.5 telah dilakukan analisa akurasi *query*-nya pada tabel 4.6. Diketahui jumlah dokumen yang relevan berjumlah 13 (**TP**), dokumen tidak relevan berjumlah 4 (**FN**), sisa dokumen yang tidak terkait berjumlah 63 (**TN**), dan dokumen yang harusnya tertangkap pada saat *discovery* namun tidak tertangkap berjumlah 2 (**FN**).

Empat *parameter* tersebut dihitung ke dalam rumus akurasi untuk memperoleh nilai akurasinya. Pertama, menjumlahkan nilai yang benar (**TN+TP)**, kedua menjumlahkan seluruh nilai (**TN**+**TP**+**FN**+**FP)**, terakhir mengimplementasi rumus untuk mendapatkan nilai akurasi dengan nilai yang benar dibagi jumlah seluruh nilai (**TN+TP/ TN**+**TP**+**FN**+**FP)**. Diperoleh hasil akurasi sebesar **92.68%** terhadap *query* tersebut.

Kemudian dilakukan perhitungan akurasi menggunakan *cosine similarity* dengan SQL *query* yaitu pada tabel 4.7 dan 4.8. Pada tabel 4.7 diketahui hasil akurasi rata-rata terhadap 20 *query* yang berbeda menghasilkan sebesar **97.20%**.

Tabel 4.7 Akurasi *query* dengan *cosine similarity*

No.	*Query*	TP	FP	TN	FN	A	B	C
1	Guru yang mengajar.	12	3	63	4	75	82	91.46%
2	Peraturan menghitung bobot nilai	4	0	78	0	82	82	100.00%
3	bekerja	3	0	79	0	82	82	100.00%
4	Tervalidasi.	1	0	81	0	82	82	100.00%
5	Referensi angkatan yang aktif.	8	0	74	0	82	82	100.00%
6	SISWA BARU YANG LULUS	8	1	71	2	79	82	96.34%
7	aspek penilaian untuk siswa	8	4	69	1	77	82	93.90%
8	Kurikulum pada pelajaran siswa.	13	4	63	2	76	82	92.68%
9	jadwal untuk pelajaran	13	3	64	2	77	82	93.90%
10	kenaikan siswa	5	4	71	2	76	82	92.68%
11	silabus pelajaran	8	0	74	0	82	82	100.00%
12	presensi keaktifan	5	0	75	2	80	82	97.56%
13	grading siswa	7	4	71	0	78	82	95.12%
14	Jumlah SKS pelajaran di kelas	7	0	75	0	82	82	100.00%
15	guru	15	0	67	0	82	82	100.00%
16	siswa belajar	8	1	73	0	81	82	98.78%
17	siswa belajar mengajar	8	1	70	3	78	82	95.12%
18	hitung bobot	2	0	80	0	82	82	100.00%
19	Penjadwalan	10	0	72	0	82	82	100.00%
20	data guru	12	3	67	0	79	82	96.34%
Avarage								**97.20%**

Keterangan:

TP : *True Positive*

FP : *False Positive*

TN: *True Negative*

FN : *False Negative*

A : *Correct Assessment* (**TN**+**TP**)

B : *All Assessment* (**TN**+**TP**+**FN**+**FP**)

C : *Correct Assessment/All Assessment* (**A/B**)

Kemudian diketahui hasil akurasi tanpa menggunakan *cosine similarity* (SQL *query*) terhadap 20 *query* yang sama seperti pada tabel 4.8, menghasilkan presentase rata-rata sebesar 90.18%.

Berikut adalah hasil akurasi tanpa menggunakan *cosine similarity* yaitu SQL *query*:

No.	*Query*	TP	FP	TN	FN	A	B	C
1	Guru yang mengajar.	0	0	66	16	66	82	80.49%
2	Peraturan menghitung bobot nilai	0	0	78	4	78	82	95.12%
3	bekerja	0	0	79	3	79	82	96.34%
4	Tervalidasi.	0	0	81	1	81	82	98.78%
5	Referensi angkatan yang aktif.	0	0	74	8	74	82	90.24%
6	SISWA BARU YANG LULUS	0	0	72	10	72	82	87.80%
7	aspek penilaian untuk siswa	0	0	73	9	73	82	89.02%
8	Kurikulum pada pelajaran siswa.	0	0	67	15	67	82	81.71%
9	jadwal untuk pelajaran	0	0	67	15	67	82	81.71%
10	kenaikan siswa	3	0	75	4	78	82	95.12%
11	silabus pelajaran	0	0	74	8	74	82	90.24%
12	presensi keaktifan	0	0	75	7	75	82	91.46%
13	grading siswa	0	0	75	7	75	82	91.46%
14	Jumlah SKS pelajaran di kelas	0	0	75	7	75	82	91.46%
15	guru	15	4	63	0	78	82	95.12%
16	siswa belajar	0	0	74	8	74	82	90.24%
17	siswa belajar mengajar	0	0	71	11	71	82	86.59%
18	hitung bobot	0	0	80	2	80	82	97.56%
19	Penjadwalan	0	0	72	10	72	82	87.80%
20	data guru	3	0	67	12	70	82	85.37%
Avarage								**90.18%**

Tabel 4.8 Akurasi *query* dengan SQL *query*

Berdasarkan hasil pengujian akurasi *query* yang menggunakan dengan SQL *query* dengan *cosine similarity* pada tabel 4.8 dan 4.7, akurasi meningkat sebesar **7.01%** dari **90.18%** menjadi **97.20%**. Jadi, tingkat akurasi *query* yang dihasilkan lebih relevan menggunakan *cosine similarity* daripada SQL *query*.

Parameter akurasi *query* yang diuji juga mempengaruhi hasil *discovery*. Semakin tinggi **nilai kebenaran** (**TP+TN**) yaitu nilai dokumen yang bernilai

benar dari seluruh dokumen (lihat pada tabel 4.5) maka nilai akurasinya semakin tinggi. Seperti pada *query* nomor 2,3,4,5,11,14,15 bernilai 100.00%. Perbandingan *query* pada nomor 1 yaitu nilai TP sebesar 16 dan TN 63, maka jumlah TP+TN yaitu 79, sedangkan *query* pada nomor 2 nilai TP sebesar 4 dan TN 78, maka jumlah TP+TN yaitu 82 dari total seluruh dokumen berjumlah 82. Jadi, tingkat akurasi *query* ke 2 lebih tinggi daripada *query yang* ke 1.

Penelitian *web service discovery* menghasilkan daftar dokumen *web service* yang relevan dan terangking sesuai nilai *similarity*-nya. Proses *discovery* dilakukan dengan membandingkan nilai *similarity* dokumen *web service* dengan *query* yang diberikan dengan perhitungan-perhitungan yang terukur sesuai dengan firman Allah:

وَلاَ تَقْرَبُواْ مَالَ الْيَتِيمِ إِلاَّ بِالَّتِي هِيَ أَحْسَنُ حَتَّى يَبْلُغَ أَشُدَّهُ وَأَوْفُواْ الْكَيْلَ وَالْمِيزَانَ بِالْقِسْطِ لاَ نُكَلِّفُ نَفْسًا إِلاَّ وُسْعَهَا وَإِذَا قُلْتُمْ فَاعْدِلُواْ وَلَوْ كَانَ ذَا قُرْبَى وَبِعَهْدِ اللّهِ أَوْفُواْ ذَلِكُمْ وَصَّاكُم بِهِ لَعَلَّكُمْ تَذَكَّرُونَ ﴿١٥٢﴾

Artinya:

"*Dan janganlah kamu dekati harta anak yatim, kecuali dengan cara yang lebih bermanfaat, hingga sampai ia dewasa.* ***Dan sempurnakanlah takaran dan timbangan dengan adil.*** *Kami tidak memikulkan beban kepada sesorang melainkan sekedar kesanggupannya. Dan apabila kamu berkata, maka hendaklah kamu berlaku adil, kendatipun ia adalah kerabat(mu), dan penuhilah janji Allah. Yang demikian itu diperintahkan Allah kepadamu agar kamu ingat.*" (QS. Al-An'am: 152)

Tafsir **Quraish Shihab** terhadap surat **Al-An'am Ayat 152** yakni jangan mengurangi timbangan atau ukuran saat kalian memberi dan jangan meminta lebih atau tambahan saat kalian menerima. Lakukanlah timbangan itu secara adil semampu kalian. Allah tidak membebani manusia kecuali sesuatu yang sesuai dengan kemampuannya, tanpa merasa terpaksa.

Sedangkan menurut tafsir **Jalalayn** yakni (**Dan sempurnakanlah takaran dan timbangan dengan adil**) secara adil dan tidak curang. (Kami tidak memikulkan beban kepada seseorang melainkan sekadar kesanggupannya) sesuai dengan kemampuannya dalam hal ini; apabila ia berbuat kekeliruan di dalam menakar atau menimbang sesuatu, maka Allah mengetahui **kebenaran** niat yang sesungguhnya, oleh karena itu maka ia tidak berdosa, sebagaimana yang telah disebutkan dalam hadis Nabi saw. (Dan apabila kamu berkata) dalam masalah hukum atau lainnya (maka hendaklah kamu berlaku **adil**) jujur (kendatipun dia) orang yang bersangkutan (adalah kerabatmu) famili (dan penuhilah janji Allah. Yang demikian itu diperintahkan Allah kepadamu agar kamu ingat) dengan memakai *tasydid* agar menjadikannya sebagai pelajaran dan juga dibaca dengan sukun.

Perhitungan yang telah peneliti lakukan juga tetap berpegang teguh pada surat **Al-An'am ayat 152**, ***"Jangan mengurangi timbangan atau ukuran saat kalian memberi dan jangan meminta lebih atau tambahan saat kalian menerima"***, pengujian sistem dilakukan dengan menghitung nilai akurasi ***query*** berdasar hasil yang ada menggunakan rumus matematika yaitu ***accuracy***. Sehingga dapat dimanfaat oleh *developer web service* untuk *discovery web service*.

Keterangan lebih jelas mengenai aspek matematika untuk menghitung akurasi adalah sebagai berikut:

وَيَا قَوْمِ أَوْفُواْ الْمِكْيَالَ وَالْمِيزَانَ بِالْقِسْطِ وَلاَ تَبْخَسُواْ النَّاسَ أَشْيَاءهُمْ وَلاَ تَعْثَوْاْ فِي الأَرْضِ مُفْسِدِينَ ﴿٨٥﴾

Artinya:

"*Wahai kaumku, cukupkanlah **takaran** dan timbangan dengan **adil**, dan janganlah kamu merugikan manusia terhadap hak-hak mereka dan janganlah kamu membuat kejahatan di bumi dengan berbuat kerusakan."* (QS. Al-Hud: 85)

Dari tafsir **Quraish Shihab** yakni "Wahai kaumku, laksanakanlah **timbangan** dan **takaran** barang yang kalian jual secara **adil** dan seimbang dengan tidak mengurangi atau melebihkannya. Jangan kalian kurangi hak-hak orang lain yang ada pada barang itu. Dan jangan kalian membuat kejahatan dan kerusakan di muka bumi dengan merampas harta, menyerang dan membegal mereka, serta menjadikan kejahatan sebagai cara memperoleh harta secara tidak benar".

Sedangkan tafsir **Jalalayn** yaitu (Dan Syuaib berkata, "Wahai kaumku!) (Cukupkanlah **takaran** dan **timbangan**) sempurnakanlah keduanya (dengan **adil**) secara tetap (dan janganlah kalian merugikan manusia terhadap hak-hak mereka) janganlah kalian mengurangi hak-hak mereka sedikit pun (dan janganlah kalian membuat kejahatan di muka bumi dengan membuat kerusakan) dengan melakukan pembunuhan dan kejahatan lainnya. *Lafal ta'tsau* berasal dari *fi'il madhi `atsiya,* artinya mengadakan kerusakan.

Sedangkan lafal *mufsidiina* berkedudukan menjadi hal atau keterangan yang mengukuhkan makna *amil*-nya yaitu *ta'tsau*.

Diperjelas lagi dalam surat **Al-Hadid ayat 25**:

لَقَدْ أَرْسَلْنَا رُسُلَنَا بِالْبَيِّنَاتِ وَأَنزَلْنَا مَعَهُمُ الْكِتَابَ وَالْمِيزَانَ لِيَقُومَ النَّاسُ بِالْقِسْطِ وَأَنزَلْنَا الْحَدِيدَ فِيهِ بَأْسٌ شَدِيدٌ وَمَنَافِعُ لِلنَّاسِ وَلِيَعْلَمَ اللَّهُ مَن يَنصُرُهُ وَرُسُلَهُ بِالْغَيْبِ إِنَّ اللَّهَ قَوِيٌّ عَزِيزٌ ﴿٢٥﴾

Artinya:

"*Sesungguhnya Kami telah mengutus rasul-rasul Kami dengan membawa bukti-bukti yang nyata dan telah* ***Kami turunkan bersama mereka Al Kitab dan neraca (keadilan) supaya manusia dapat melaksanakan keadilan***. *Dan Kami ciptakan besi yang padanya terdapat kekuatan yang hebat dan berbagai manfaat bagi manusia, (supaya mereka mempergunakan besi itu) dan supaya Allah mengetahui siapa yang menolong (agama)Nya dan rasul-rasul-Nya padahal Allah tidak dilihatnya. Sesungguhnya Allah Maha Kuat lagi Maha Perkasa*". (QS. Al-Hadid: 25)

Dari **Quraish Shihab** "Kami benar-benar telah mengutus para rasul yang Kami pilih dengan membawa beberapa mukjizat yang kuat. Bersama mereka juga Kami turunkan kitab suci-kitab suci yang mengandung hukum, syariat agama, dan **timbangan** yang mewujudkan **keadilan** dalam hubungan antarmanusia.

Sedangkan dari tafsir **Jalalayn** yaitu (Sesungguhnya Kami telah mengutus rasul-rasul Kami) yaitu malaikat-malaikat-Nya kepada nabi-nabi (dengan membawa bukti-bukti yang nyata) **hujah-hujah yang jelas dan akurat** (dan

telah Kami turunkan bersama mereka Alkitab) lafal Alkitab ini sekalipun bentuknya *mufrad* tetapi makna yang dimaksud adalah jamak, yakni *al-kutub* (**dan neraca**) yakni keadilan (**supaya manusia dapat melaksanakan keadilan)**.

Ayat-ayat diatas seperti **Al-Hud ayat 85**, **Al-Hadid ayat 25** jelas sekali meletakkan dasar **keadilan** bagi para ahli matematika dan statiska. Mereka harus bekerja keras **menghitung bilangan-bilangan secara tepat**, sehingga semua pihak yang berkepentingan bisa merasakan keadilan. Tidak boleh ada selisih atau inkonsistensi dalam hitungan. Semuanya harus dilakukan secara cermat dan **akurat** sehingga menghasilkan **kebenaran** yang *sahih*.

Hal inilah yang ditekakan dalam Al-Quran. Ketepatan serta ***accuracy*** perhitungan yang dilakukan oleh para ahli matematika bahkan subyek diluar matematika termasuk teknik informatika juga mengaplikasikan perhitungan. Hal tersebut dilakukan demi menjamin keadilan kepada siapa saja yang berkepentingan, melainkan juga demi memperoleh informasi yang benar berdasarkan bilangan dan angka yang disajikan kepada mereka dan demi menjaga keadilan terhadap semua pihak dalam segala keadaan.

Dengan demikian,dapat kita nyatakan di sini bahwa Al-Quran telah banyak mendorong manusia untuk melakukan penelitian tentang persamaan matematis. Al-quran bukan saja telah mendorong mereka untuk menghitung bilangan-bilangan secara tepat berdasarkan data-data yang mereka miliki menurut kaidah-kaidah saintifik, melainkan juga mendorong mereka memelihara hubungan yang erat dengan Allah yaitu Sang Pencipta melalui hasil-hasil perhitungan yang

dilakukannya. Itulah sebabnya, mengapa matematika diklaim memiliki kedudukan yang istimewa dalam sains **Islam**.

Peneliti mengacu pada surat-surat diatas menghasilkan ukuran ***accuracy*** dalam bentuk bilangan yang **terukur**. Ada ayat lain dalam Al-Quran untuk memperjelas dan memperkuat pandangan di atas dan mendorong manusia untuk melakukan penelitian dengan cara yang serupa. Ayat yang dimaksud adalah:

"*Kami tidak memikulkan beban kepada sesorang melainkan sekedar kesanggupannya. Dan apabila kamu berkata, maka hendaklah kamu berlaku* ***adil****, kendatipun ia adalah kerabat(mu), dan penuhilah janji Allah. Yang demikian itu diperintahkan Allah kepadamu agar kamu ingat.*" (QS. Al-An'am: 152)

Pada surat **Al-Baqarah ayat 286** berkaitan dengan usaha-usaha yang dilakukan oleh peneliti melewati serangkaian proses untuk mengembangkan *web service repository* demi tercapainya keberhasilan sistem, berikut ayatnya:

لاَ يُكَلِّفُ اللّهُ نَفْسًا إِلاَّ وُسْعَهَا لَهَا مَا كَسَبَتْ وَعَلَيْهَا مَا اكْتَسَبَتْ رَبَّنَا لاَ تُؤَاخِذْنَا إِن نَّسِينَا أَوْ أَخْطَأْنَا رَبَّنَا وَلاَ تَحْمِلْ عَلَيْنَا إِصْرًا كَمَا حَمَلْتَهُ عَلَى الَّذِينَ مِن قَبْلِنَا رَبَّنَا وَلاَ تُحَمِّلْنَا مَا لاَ طَاقَةَ لَنَا بِهِ وَاعْفُ عَنَّا وَاغْفِرْ لَنَا وَارْحَمْنَآ أَنتَ مَوْلاَنَا فَانصُرْنَا عَلَى الْقَوْمِ الْكَافِرِينَ ﴿٢٨٦﴾

Artinya:

"***Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya. Ia mendapat pahala (dari kebajikan) yang diusahakannya dan ia mendapat siksa (dari kejahatan) yang dikerjakannya. (Mereka***

berdoa)*: "Ya Tuhan kami, janganlah Engkau hukum kami jika kami lupa atau kami tersalah. Ya Tuhan kami, janganlah Engkau bebankan kepada kami beban yang berat sebagaimana Engkau bebankan kepada orang-orang sebelum kami. Ya Tuhan kami, janganlah Engkau pikulkan kepada kami apa yang tak sanggup kami memikulnya. Beri maaflah kami; ampunilah kami; dan rahmatilah kami. Engkaulah Penolong kami, maka tolonglah kami terhadap kaum yang kafir*" (QS. Al-Baqarah: 286).

Dari tafsir **Quraish Shihab** pada surat **Al-Baqarah ayat 286**, Allah tidak membebani hamba-hamba-Nya kecuali dengan sesuatu yang dapat dilaksanakan. Maka, setiap orang yang *mukallaf*, amalnya akan dibalas: yang baik dengan kebaikan, dan yang jelek dengan kejelekan.

Dari tafsir **Jalalayn** yaitu (**Allah tidaklah membebani seseorang melainkan sesuai dengan kemampuannya**), artinya sekadar kesanggupannya. (**Ia mendapat dari apa yang diusahakannya**) berupa kebaikan artinya pahalanya (dan ia beroleh pula dari hasil kejahatannya), yakni dosanya. Maka seseorang itu tidaklah menerima hukuman dari apa yang tidak dilakukannya, hanya baru menjadi angan-angan dan lamunan mereka.

Surat **Al-Baqarah ayat 286** memiliki gambaran yang sama dengan penelitian ini yakni apa yang telah diusahakan akan didapatkan hasil setimpal. Penelitian ini dilakukan untuk **menemukan *web service*** dari *repository web service*. Peneliti melakukan serangkaian rencana dan tahapan yang dikerjakan mulai dari memanfaatkan hasil ***parsing web service*** sebagai data untuk dipreproses. Dalam tahapan ***preprocessing*** dilakukan lagi serangkaian tahapan untuk menghasilkan dokumen yang mudah untuk diolah yaitu dengan

casefolding, ***tokenizing***, ***stopword***, dan ***stemming***. Selanjutnya dilakukan tahapan lagi untuk memberikan bobot ***TF.IDF*** pada setiap kata dalam dokumen *web service* sehingga dapat dihitung nilai ***similarity*** nya menggunakan ***cosine similarity***. Hasilnya dokumen *web service* dapat ditemukan oleh *user* atau *web service developer* dengan meng-*input*-kan *query*. Dokumen yang telah ditemukan diuji kembali oleh peneliti untuk dihitung **akurasi** *query*-nya dengan hasil akurasi meningkat. Pengujiannya yaitu menggunakan perhitungan akurasi yang terukur sesuai firman Allah pada surat-surat yang telah dipaparkan di atas. Dengan demikian, aplikasi ini bisa dimanfaatkan dengan bijak dan sebaik-baiknya serta memudahkan *user* untuk mengembangkan *web service* yang sudah tersedia.

BAB 5

KESIMPULAN DAN SARAN

Bab ke lima pada penelitian ini merupakan bab terakhir yang berisi kesimpulan dari hasil penelitian yang telah dilakukan dan saran untuk pengembangan penelitian berikutnya.

5.1 Kesimpulan

Aplikasi *web service discovery* menggunakan *cosine similarity* untuk meningkatkan akurasi *query* pada *web service repository* sebagai media bagi para *developer web service* yang ingin membangun *web service* dengan memanfaatkan *web service* yang telah ada untuk dimodifikasi atau *reuse*. *Developer web service* dapat memanfaatkan aplikasi ini untuk menemukan *web service*. Adapun hasil pengujian yang telah dilakukan peneliti menghasilkan beberapa *point* kesimpulan sebagai berikut:

1. Penelitian ini dilakukan untuk membangun *web service discovery* pada *web service repository*. Peneliti melakukan serangkaian rencana dan tahapan mulai dari memanfaatkan hasil *parsing web service* sebagai data untuk dipreproses. Dalam tahapan ***preprocessing***, bagian dari dokumen *web service* yang dapat diolah yaitu bagian *meta*, karena hanya *field meta* yang bersifat *standard*. Selanjutnya, mengolah *meta* pada setiap dokumen *web service* untuk memudahkan proses komputasi *similarity* yaitu dengan ***casefolding***, ***tokenizing***, ***stopword***, dan ***stemming***. Memberikan bobot ***TF.IDF*** pada setiap kata dalam dokumen *web service* sehingga dapat

dihitung nilai ***similarity***-nya menggunakan ***cosine similarity***. Hasilnya dokumen *web service* dapat ditemukan oleh *user* atau *web service developer* dengan meng-*input*-kan *query*. Dokumen *web service* yang telah ditemukan diuji kembali oleh peneliti untuk dihitung akurasi *query*-nya.

2. *Web service discovery* menggunakan metode *cosine similarty* memiliki tingkat akurasi lebih tinggi jika dibandingkan dengan *discovery* menggunakan SQL *query* saja. Pengujian yang telah dilakukan menggunakan *accuracy* nilai kebenaran menghasilkan *query* dapat lebih berkembang dan kompleks jika menggunakan *cosine similarity*. Berdasarkan pengujian dan analisa yang telah dilakukan terhadap 20 *query* pengujian dengan membandingkan antara SQL *query* dan *cosine similarity*, diperoleh peningkatan akurasi *query* sebesar sebesar 7.01% dari 90.18% menjadi 97.20%. Jadi, tingkat akurasi *query* yang dihasilkan lebih relevan menggunakan *cosine similarity* daripada menggunakan SQL *query*, karena komputasi dari nilai *similarity web service*, fungsi *cosine* memiliki kinerja lebih efisien daripada SQL *query*.

5.2 Saran

Pengembangan penelitian *web service discovery* perlu dilanjutkan untuk:

1. Penambahan domain *web service* menjadi lebih kompleks, pada penelitian ini hanya mencakup domain *web service* akademik dan kurikulum.
2. Variasi metode *similarity* seperti *dice, jaccard* dan lain sebagainya.
3. Standarisasi nama *web service*.
4. Pengembangan aplikasi dengan fungsi komposisi *web service* dari hasil *discovery web service* yang telah didekomposisi.

DAFTAR PUSTAKA

Antoniou, G. d. (2008). *A Semantic web Primer.* MIT Press.

ASIA, D. W. (2015, Februari). Perancangan Information Retrieval (IR) untuk Pencarian Ide Pokok Teks Artikel Berbahasa Inggris dengan Pembobotan *Vector Space Model*. Jurnal Ilmiah Teknologi dan Informasi ASIA, *09*, 01.

Bell, D. a. (2005). *Semantic transformation of web services*. Springer, (pp. 856--865).

Bose, A. a. (2008). *Improving web service discovery by using semantic models*. *International Conference on Web Information Systems Engineering*, (pp. 366--380).

Cheng, B. Z. (2016). *Web Services discovery approach based on mining interface semantics*. *In Web Services (ICWS) IEEE International Conference*, (pp. pp. 332-339).

Djeni, C. A., Riyanarto, S., & Sunaryono, D. (2013). Rancang Bangun *Workflow Management System pada Kasus Enterprise Resource Planning* (ERP). Jurnal Teknik POMITS, II (1), ISSN: 2337-3539.

Hamzah, A., Soesianto, F., Susanto, A., & Istiyanto, J. E. (2008). Studi Kinerja Fungsi-Fungsi Jarak dan Similaritas dalam *Clustering* Dokumen Teks Berbahasa Indonesia. Seminar Nasional Informatik*a*, 1979-2328.

http://kemenpora.go.id. (n.d.). Bakuan Format dan Ontologi Data & Interoperabilitas Layanan SIM Kemenpora RI.

Ma, Z., & Leymann, F. (2008). *A Lifecycle Model for Using Process Decompose in Business Process Modeling*. *Institute of Architecture of Application Systems (IAAS) University of Stuttgart*.

Mutiara, A. B., Muslim, A., Putri, T., Oswari, T., & Silfianti, W. (2013). Aplikasi Pencarian Jadwal Dokter dan Fasilitas Rumah Sakit *e-Doctor Schedule & Hospital* Info Berbasis Web Semantik. Depok: Gunadrama.

Nava'atul, F. a. (2011). Penerapan Teknologi *Semantic* Web pada Aplikasi Pencarian Koleksi Perpustakaan (Studi Kasus: Perpustakaan Fti Upn" Veteran" Yogyakarta). Telematika (16).

Powers, D. M. (2011). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. Bioinfo *Publications*.

R.K, B. K. (2016). *Ontologies: A Review of Web Service Discovery Techniques*. IJAIC, Vol.7 (Issue 6), pp.1-12.

Sarno, R., Yeni, A., & Fitri, R. (2012). *Semantic Search Pencarian Berdasarkan Konten.* Yogyakarta: Penerbit ANDI.

Suprianto, M. E. (2017). Integrasi Sistem Informasi Akademik pada *Enterprise Resource Planning* Pondok Pesantren Tipe D menggunakan *Service Oriented Architecture Web Services.* Malang: UIN Maliki Malang.

Sycara, K. a. (2006). *Tools and Technologies for semantic web services: An OWL-S perspective*. ISWC.

T.P., B. P., & Gunawan, I. (2015). Sistem *Information Retrieval* Pencarian Kesamaan Ayat Terjemahan Al Quran Berbahasa Indonesia dengan *Query Expansion* Dari Tafsirnya. *IDeaTech.* Surabaya.

Thomas, O. a. (2007). *Semantic EPC: Enhancing Process Modeling Using Ontology Languages*. SBPM, 251.

Triana, A., Saptono, R., & Sulistyo, M. E. (2014). Pemanfaatan Metode *Vector Space Model* dan Metode *Cosine Similarity* pada Fitur Deteksi Hama dan Penyakit Tanaman Padi. *Research Gate.* Surakarta.

Wulan, F. R. (2018). *Clustering* Hasil Dekomposisi *Web Service* pada *Enterprise Resource Planning* Pondok Pesantren Menggunakan *Parsing Web Services Decription Language* dan *K-Means Clustering.* Malang: Teknik Informatika UIN Maliki Malang.

Zhu, W. a. (2010). *Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations. NESUG proceedings: health care and life sciences, Baltimore, Maryland*, 1-9.

LAMPIRAN

Data *operation name* dan *meta* pada *web service* akademik:

No.	*Operation Name*	*Meta*
1	kurikulum.getMapel	kurikulum mata pelajaran
2	kurikulum.getMapelByDepartemen	kurikulum mata pelajaran
3	kurikulum.getSilabus	kukikulum silabus
4	kurikulum.getSilabusAll	kurikulum silabus
5	kurikulum.getSks	kurikulum sks
6	kurikulum.rppbyidmapel	rpp mata pelajaran
7	referensiExecute.insertangkatan	referensi angkatan
8	referensiExecute.deleteangkatan	referensi angkatan
9	referensiExecute.updateangkatan	referensi angkatan
10	referensiExecute.updateangkatanaktif	referensi angkatan aktif
11	referensiExecute.inserttingkat	referensi tingkat
12	referensiExecute.deletetingkat	referensi tingkat
13	referensiExecute.updatetingkat	referensi tingkat
14	referensiExecute.updatetingkataktif	referensi tingkat aktif
15	referensiExecute.insertsemester	referensi semester
16	referensiExecute.deletesemester	referensi semester
17	referensiExecute.updatesemester	referensi semester
18	referensiExecute.updatesemesteraktif	referensi semester aktif
19	referensiExecute.insertkelas	referensi kelas
20	referensiExecute.deletekelas	referensi kelas
21	referensiExecute.updatekelas	referensi kelas
22	referensiExecute.updatekelasaktif	referensi kelas aktif
23	referensiExecute.inserttabelvalidasi	tabel validasi
24	psbExecute.addagama	psb agama
25	psbExecute.addsuku	psb suku
26	psbExecute.addkondisi	psb kondisi
27	psbExecute.addpendidikan	psb pendidikan
28	psbExecute.addpekerjaan	psb pekerjaan
29	psbExecute.updateagama	psb agama
30	psbExecute.updatesuku	psb suku
31	psbExecute.updatekondisi	psb kondisi
32	psbExecute.updatependidikan	psb pendidikan
33	psbExecute.updatepekerjaan	psb pekerjaan
34	psbExecute.deleteagama	psb agama
35	psbExecute.deletesuku	psb suku
36	psbExecute.deletekondisi	psb kondisi
37	psbExecute.deletependidikan	psb pendidikan
38	psbExecute.deletepekerjaan	psb pekerjaan

39	psbExecute.deletesiswabaru	psb siswa baru
40	psbExecute.insertsiswa	psb siswa
No.	***Operation Name***	***Meta***
41	psbExecute.insertsiswa2	psb siswa
42	guruExecute.insertmapel	guru mata pelajaran
43	guruExecute.deletemapel	guru mata pelajaran
44	guruExecute.updatemapel	guru mata pelajaran
45	guruExecute.insertjenisuji	guru jenis uji
46	guruExecute.deletejenisuji	guru jenis uji
47	guruExecute.updatejenisuji	guru jenis uji
48	guruExecute.insertaspeknilai	guru aspek nilai
49	guruExecute.deleteaspeknilai	guru aspek nilai
50	guruExecute.updateaspeknilai	guru aspek nilai
51	guruExecute.insertstatusguru	status guru
52	guruExecute.deletestatusguru	status guru
53	guruExecute.updatestatusguru	status guru
54	guruExecute.insertdataguru	data guru
55	guruExecute.deletedataguru	data guru
56	guruExecute.updatedataguru	data guru
57	guruExecute.insertaturangrading	aturan grading
58	guruExecute.deleteaturangrading	aturan grading
59	guruExecute.insertaturanhasilbobot	aturan hasil bobot
60	guruExecute.deleteaturanhasilbobot	aturan hasil bobot
61	jadwalExecute.insertjam	jadwal jam
62	jadwalExecute.deletejam	jadwal jam
63	jadwalExecute.updatejam	jadwal jam
64	jadwalExecute.insertinfojadwal	jadwal info
65	jadwalExecute.deleteinfojadwal	jadwal info
66	jadwalExecute.updateinfojadwal	jadwal info
67	jadwalExecute.updateinfojadwalaktif	jadwal info
68	jadwalExecute.insertjadwal	jadwal
69	jadwalExecute.updatejadwal	jadwal
70	jadwalExecute.deletejadwal	jadwal
71	siswaExecute.updatesiswaaktif	siswa aktif
72	kenaikanExecute.angkatan	naik kelas
73	kenaikanExecute.updatesiswatidaknaik	siswa tidak naik
74	kenaikanExecute.updatesiswanaik	siswa naik
75	kenaikanExecute.updatesiswalulus	siswa lulus
76	mutasiExecute.insertjenismutasi	jenis mutasi
77	mutasiExecute.updatejenismutasi	jenis mutasi
78	mutasiExecute.deletejenismutasi	jenis mutasi
79	mutasiExecute.updatemutasisiswa	mutasi siswa
80	presensiExecute.insertpresensiharian	presensi hari
81	presensiExecute.insertphsiswa	presensi siswa
82	presensiExecute.deletepresensiharian	presensi hari