TERM WEIGHTING BERBASIS INDEKS KELAS MENGGUNAKAN METODE TF.IDF.ICS$_\delta$F UNTUK PERANGKINGAN DOKUMEN AL-QUR'AN

SKRIPSI

Oleh :
KURNIAWATI
12650009

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

TERM WEIGHTING BERBASIS INDEKS KELAS MENGGUNAKAN METODE TF.IDF.ICS$_\delta$F UNTUK PERANGKINGAN DOKUMEN AL-QUR'AN

SKRIPSI

Diajukan Kepada :
Dekan Fakultas Sains Dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim (UIN) Malang
untuk Memenuhi Salah Satu Persyaratan dalam Memperoleh
Gelar Sarjana Komputer (S.Kom)

Oleh :
KURNIAWATI
12650009

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

LEMBAR PENGESAHAN

TERM WEIGHTING BERBASIS INDEKS KELAS MENGGUNAKAN METODE TF.IDF.ICS$_{\delta}$F UNTUK PERANGKINGAN DOKUMEN AL-QUR'AN

SKRIPSI

Oleh :
KURNIAWATI
NIM. 12650009

Telah Dipertahankan Di Depan Dewan Penguji Skripsi Dan Dinyatakan Diterima
Sebagai Salah Satu Persyaratan Untuk Memperoleh
Gelar Sarjana Komputer (S.Kom)

Tanggal, Juni 2016

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	**: Irwan Budi Santoso, M.Kom** **NIP. 19770103201101 1 004**	**(……)**
2. Ketua	**: Dr. M. Amin Hariyadi, M.T** **NIP. 19670118 200501 1 001**	**(……)**
3. Sekretaris	**: A'la Syauqi, M.Kom** **NIP. 19771201 200801 1 007**	**(……)**
4. Anggota	**: Zainal Abidin, M.Kom** **NIP. 19760613 200501 1 004**	**(……)**

Mengetahui dan Mengesahkan
Ketua jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424200901 1008

LEMBAR PERSETUJUAN

TERM WEIGHTING BERBASIS INDEKS KELAS MENGGUNAKAN METODE TF.IDF.ICS$_\delta$F UNTUK PERANGKINGAN DOKUMEN AL-QUR'AN

SKRIPSI

Oleh :

KURNIAWATI

NIM. 12650009

Telah disetujui oleh:

Pembimbing I	Pembimbing II
A'la Syauqi, M.Kom	**Zainal Abidin, M.Kom**
NIP. 19771201 200801 1 007	**NIP. 10760613 200501 1 004**

Juni 2016
Mengetahui,
Ketua jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424200901 1008

SURAT PERNYATAAN
ORISINILITAS PENELITIAN

Saya yang bertanda tangan dibawah ini :

Nama	:	KURNIAWATI
NIM	:	12650009
Fakultas / Jurusan	:	Sains dan Teknologi / Teknik Informatika
Judul Penelitian	:	***TERM WEIGHTING* BERBASIS INDEKS KELAS MENGGUNAKAN METODE TF.IDF.ICS$_\delta$F UNTUK PERANGKINGAN DOKUMEN AL-QUR'AN**

Menyatakan dengan sebenar-benarnya 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 jiplakan, maka saya bersedia untuk mempertang jawabankan, serta diproses sesuai peraturan yang berlaku.

Malang, 21 Juni 2016

Yang membuat Pernyataan

Kurniawati

NIM. 12650009

HALAMAN PERSEMBAHAN

Skripsi ini didedikasikan kepada kedua orang tuaku, sungguh apa yang mereka berikan melebihi daripada apa yang pernah aku inginkan.

Dari titisan Ilmu-Mu kupersembahkan goresan tanganku bagi ilmu pengetahuan Indonesia

MOTTO

الأَدَبُ فَوْقَ الْعِلْمِ

"Adab lebih tinggi derajatnya dari pada ilmu "

UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM
PUSAT PERPUSTAKAAN

KATA PENGANTAR

Segala puji syukur kehadirat Allah SWT sehingga skripsi ini dapat diselesaikan dengan baik. Meski dalam menyelesaikan skripsi ini banyak ditemui kesulitan, namun berkat bantuan dan bimbingan berbagai pihak, akhirnya Penulis berhasil menyelesaikan skripsi ini. Dengan segala kerendahan hati melalui halaman persembahan ini, Penulis menghaturkan terima kasih kepada pihak-pihak yang membantu Penulis dalam menyelesaikan skripsi ini baik secara langsung maupun tidak langsung.

1. Kepada Bapak dan Ibu yang telah mendidik dan membesarkan Penulis hingga 22 tahun yang tiada henti memebrikan do'a, pengertian, dukungan, dan pengorbanan yang besar selama Penulis menyelesaikan studi ini.
2. Kepada Dosen Pembimbing Bapak A'la Syauqi, M.Kom dan Bapak Zainal Abidin, M.Kom. yang dengan sabar membimbing penulis, sehingga Penulis dapat menyelesaikan skripsi ini.
3. Kepada Bapak Fatchurrochman, M.Kom selaku Dosen wali.
4. Kepada Teman-teman S1 seangkatan yang berusaha bersama dan saling mendukung dalam menjalani studi.
5. Kepada semua pihak yang tidak bisa Penulis sebutkan satu persatu, terima kasih atas segala bantuan, baik berupa ide, gagasan, pemikiran, atau bahkan sekedar kesediaan mendengarkan keluh kesah.

Akhirnya, Penulis berharap, laporan skripsi ini dapat memberikan kontribusi ilmiah bagi khasanah riset di bidang informatika. Mohon maaf atas segala kelebihan dan kekurangannya.

Malang, 4 Ramadhan 1437 H

9 Juni 2016 M

Kurniawati

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Kurniawati, 2016. ***Term Weighting*** **Berbasis Indeks Kelas Menggunakan metode TF.IDF.ICS$_\delta$F untuk Perangkingan Dokumen Al-Qur'an.** Pembimbing : (1) A'la Syauqi, M.Kom (2) Zainal Abidin, M.Kom.

Kata Kunci : Perangkingan Dokumen, Pembobotan Kata, TF.IDF, ICF, ICS$_\delta$F, Indeks Kelas

Information Retrieval berdasarkan query tertentu sudah biasa ditemukan pada sistem komputer saat ini. Salah satu metode yang popular digunakan adalah perangkingan dokumen menggunakan *Vector Space Model* (VSM) berbasis pada nilai pembobotan kata TF.ID. Penelitian ini menggunakan pendekatan pembobotan kata berbasis pengindeksan kelas dan membandingkan perngaruhnya pada dua metode pembobotan lain yang berbeda yang dijadikan pendekatan dasar. Metode pembobotan akan diterapkan pada dataset Al-Qur'an yang dijadikan tolak ukur koleksi. Al-Qur'an memiliki banyak ayat, masing-masing ayat dari Al-Qur'an tersebut adalah sebuah dokumen yang akan diranking berdasarkan *query* pengguna. TF.IDF hanya melakukan pembobotan berbasis dokumen tanpa memperhatikan kelas yang merupakan induk dokumen tersebut. Sementara pendekatan menggunakan TF.IDF.ICF hanya memperhatikan indeks kelas belum menghiraukan anggota dari setiap kelas yang menyebabkan sulit untuk membedakan frekuensi *term* yang jarang muncul. Oleh sebab itu diaujukan fungsi ICF yang mengimplementasikan *inverse class density frequency* (ICS$_\delta$F), dan menghasilkan metode TF.IDF.ICS$_\delta$F yang memberikan nilai diskriminasi positif pada *term* yang sering dan jarang muncul. Hasil penelitian menunjukkan bahwa metode yang diusulkan menghasilkan kinerja yang lebih bagus dibanding metode sebelumnya dengan nilai akurasi 93%.

ABSTRACT

Kurniawati, 2016. ***Term Weighting*** **Based Class Indexes Using TF.IDF.ICS$_\delta$F Method For Al-Qur'an Document Ranking.** Pembimbing : (1) A'la Syauqi, M.Kom (2) Zainal Abidin, M.Kom.

Keywords : Document Ranking, Term Weighting, TF.IDF, ICF, ICS$_\delta$F, Class Index

Information Retrival based on specific queries is common to the current computer systems. One of the popular methods used in the document ranking method using vector space models based on TF.IDF term weighting. In this research, we introduce class-indexing-based term-weighting approaches and judges their effect two other different term weighing approaches that are considered as the baseline approaches. In the experiment, we investigate the effects of method over the Al-Qur'an datasets as benchmark collection. Al-Qur'an contain many verse, each verse of the Al-Qur'an is a single document that will be ranked based on the user query. TF.IDF method only performs term weighting based on document without regard to the indexes of the class of the document. While TF.IDF.ICF approach without any prior knowledge of the class space. Therefore, inverse class space density frequency (ICS$_\delta$F) based category mapping is proposed and multiplied by TF.IDF to generate TF.IDF.ICS$_\delta$F, which provide positive discrimination on rare terms in the vector space. The experimental result show that the proposed method can be implemented on document ranking and the performance are better than some previous methods with accurate value 93%.

ملخص

كورنييواتي 2016 الوزن على المدى مؤشرات استنادا طريق ت ف.ادف.اج س ة ف أسلوب فئة لآل القرآن وثيقة الترتيب

المشرف: (1) أعلى شوقى الماجيستر (2) زين العابدين الماجيستر

كلمات البحث: الترتيب ثيقة ، ترجيح المدى ، مؤشر الطبقة.

إحدى الطرق الشائعة .استرجاع المعلومات وفقا لطلبات محددة من الشائع أن أنظمة الكمبيوتر الحالية المستخدمة في الأسلوب وثيقة الترتيب باستخدام نموذج فضاء المتجه بناء على ترجيح المدى ت ف.ادف. في هذا البحث، ونحن نقدم النهج والقضاة تأثيرها نهجين المدى وزنها المختلفة الأخرى التي تعتبر مع اقتراب في التجربة، ونحن التحقيق في الآثار الطريقة .خط الأساس الأجل الترجيح على أساس الدرجة الفهرسة آل القرآن تحتوي على العديد من الآية، كل آية من .على مجموعات البيانات آل القرآن كما جمع المعيار سورة القرآن الكريم هي وثيقة واحدة من شأنها أن يكون في المرتبة بناء على طلب المستخدم. طريقة ت ف.ادف يؤدي فقط الترجيح المدى استنادا إلى الوثيقة دون النظرإلى الأرقام القياسية لفئة من الوثيقة. في حين وبالتالي، مساحة الطبقة معكوس تردد كثافة (اج .ت ف . ادف. اج ف نهج دون أي معرفة مسبقة من سعة الفصول س ة ف) ويقترح رسم الخرائط الفئة بناء ومضروبا ت ف . ادف لتوليد ت ف . ادف. اج س ة ف ،التي توفر التمييز وتظهر نتيجة التجريبي أن الطريقة المقترحة يمكن تنفيذها على .الإيجابي بشروط نادرة في الفضاء ناقلات وثيقة الترتيب وأداء أفضل من بعض الأساليب السابقة مع قيمة دقيقة 93%.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini semakin banyak kebutuhan untuk menemukan informasi tertentu dari data-data yang banyak secara cepat pada internet. Komputer desktop saat ini juga dapat menyimpan data dalam jumlah yang sangat besar hingga *multi-tera-byte*. Mebuka *file* satu persatu untuk mencari informasi jelas bukan merupakan tindakan yang efektif. Pencarian data sederhana untuk mendapatkan informasi berdasarkan kata dan memasangkannya dengan suatu dokumen sudah umum dilakukan pada sistem komputer saat ini. Proses ini bisa memberikan hasil pencarian dokumen yang ditemukan pada sistem baik hasil yang relevan ataupun tidak. Namun pemrosesan ini memiliki banyak kelemahan seperti waktu proses yang lama, (Manning, 2008).

Berdasarkan keterbatasan tersebut diperlukan suatu metode untuk pencarian informasi yang efektif. Metode-metode pencarian yang efektif dipelajari dalam bidang temu kembali informasi (Manning, 2008). Temu kembali informasi merupakan tindakan, metode dan prosedur untuk menemukan kembali data yang tersimpan sesuai subyek yang dibutuhkan. Tindakan tersebut mencakup teks pembentukan indeks, *inquiry analysis* dan *relevance analysis*. Dengan data mencakup teks, tabel, gambar, suara, dan video (Salton, 1989).

Objek pembahasan temu kembali semakin luas. Tidak hanya pada dokumen berbahasa Inggris yang telah mendominasi temu kembali informasi selama lebih dari 50 tahun (Salton, 1989). Hal ini karena pembahasan temu kembali informasi

dalam bahasa Arab memiliki tantangan dari aspek perbedaan struktur bahasa Arab dengan bahasa latin seperti bahasa Inggris (Ibrahim, 2007).

Perangkingan dokumen merupakan salah satu pembahasan temu kembali informasi yang biasa diteliti. Perangkingan dokumen ini dilakukan untuk menyediakan informasi dokumen yang sesuai dengan data yang diinginkan pengguna dari *query* pengguna (Manning, 2008) (Esraa, 2010) .

Salah satu cara untuk melakukan perangkingan pada dokumen bahasa Arab adalah dengan menggunakan *vector space model* (Harrag, 2008). Pada metode ini dokumen direpresentasikan sebagai sebuah vektor yang dibentuk dari nilai-nilai *term* yang menjadi indeksnya. Nilai-nilai tersebut dihitung dengan menggunakan *term weighting* TF.IDF. metode tersebut mengkombinasikan *term frequency* (TF) yang mengukur kepadatan *term* dalam sebuah dokumen dikalikan dengan *inverse document frequency* (IDF) yang mengukur kepentingan sebuah *term* (kelangkaannya pada keseluruhan korpus) (Salton, 1989). Metode TF.IDF hanya berbasis pada *term* dalam satu dokumen tidak cukup untuk menentukan indeks dokumen. Penentuan indeks yang akurat juga bergantung pada kepentingan *term* tehadap kelas (kelangkaan *term* pada keseluruhan kelas). Sehingga dibutuhkan *term weighting* berbasis kelas yang dinamakan *iverse class frequency* (ICF). Namun ICF hanya memperhatikan *term* yang ada pada kelas tanpa memperhatikan jumlah *term* dalam dokumen yang menjadi anggota kelas.

Oleh karena itu dalam penelitian ini, diusulkan metode *term weighting* berbasis kelas dengan menambahkan faktor *space density*, metode ini secara lengkap disebut *inverse class space density frequency* ($ICS_{\delta}F$). Metode $ICS_{\delta}F$

memperhatikan kemunculan *term* pada kumpulan dokumen yang menjadi anggota *class/category* sehingga dapat meningkatkan bobot *term* dalam dokumen yang menjadi anggota *class*. Pembobotan ini digabungkan dengan pembobotan yang telah ada sebelumnya dengan cara mengalikannya, sehingga diperoleh metode *term weighting* TF.IDF.ICS$_\delta$F. Metode ini diharapkan akan perangkingan dokumen dengan urutan yang tepat dan sesuai yang diharapkan pengguna.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya maka terdapat permasalahan yang diangkat dalam penelitian ini yaitu : Seberapa besar pengaruh nilai *Space Density* pada *term weighting* TF.IDF.ICS$_\delta$F terhadap perangkingan dokumen Al-Qur'an?

1. Bagaimana melakukan perangkingan dengan menggunakan metode penbobotan TF.IDF.ICS$_\delta$F pada dokumen Al-Qur'an?
2. Bagaimana tingkat akurasi pencarian pada metode pembobotan TF.IDF.ICS$_\delta$F?

1.3 Tujuan

Adapun maksud dan tujuan yang didapat dari penelitian ini adalah sebagai menemukan dokumen yang relevan dengan *keyword* topik yang diinginkan pengguna.

1. Mendapatkan hasil perangkingan dokumen dengan menggunakan metode TF.IDF.ICS$_\delta$F pada dokumen Al-Qur'an.
2. Memperoleh nilai akurasi dari metode pembobotan TF.IDF.ICS$_\delta$F.

1.4 Manfaat

Manfaat yang didapat dari penelitian ini dapat dipandang dari tiga aspek yaitu peneliti, keilmuan dan manfaat dari sisi pembaca. Adapun manfaat tersebut adalah sebagai berikut :

1) Peneliti :
   a. Dapat memperdalam pemahaman dan implementasi konsep/teori tentang perangkingan dokumen dan *term weighting*.
   b. Sebagai sarana untuk mengetahui perbandingan kinerja metode *term weighting* yang diusulkan peneliti terhadap beberapa metode yang ada sebelumnya.
2) Keilmuan :
   a. Turut serta dalam rangka mengintegrasikan ilmu pengetahuan dan tekonlogi dengan Islam yang berupa penerapan metode *term weighting* TF.IDF.ICS$_{\delta}$F terhadap perangkingan dokumen Al-Qur'an.
   b. Menghasilkan metode *term weighting* TF.IDF.ICS$_{\delta}$F untuk melakukan perangkingan pada dokumen teks berbahasa Arab.
3) Pembaca : metode yang dihasilkan dapat di jadikan sebagai metode acuan untuk melakukan *term weighting*.

1.5 Batasan Masalah

Pembahasan dalam penelitian ini dibatasi pada beberapa hal berikut ini :

1) Metode *term weighting* yang diusulkan diterapkan pada dokumen berbahasa Arab.

2) Kinerja dari metode yang diusulkan dibandingkan dengan metode TF.IDF dan TF.IDF.ICF.

BAB II

TINJAUAN PUSTAKA

Pada bagian ini membahas tentang penelitian yang terkait dan konsep tentang teori yang digunakan dalam melakukan penelitian ini.

2.1 Penelitian Terkait

Harrag dkk (2008) menggunakan *vector space model* untuk melakukan perangkingan dokumen berbahasa Arab. Harrag membuat aplikasi pengindeksan hadist menurut derajat kemiripannya. Pada metode ini dokumen direpresentasikan sebagai sebuah vektor yang dibentuk dari nilai-nilai *term* yang menjadi indeksnya. Nilai-nilai *term* tersebut dihitung dengan mengunakan *term weighting* TF.IDF dan mengukur nilai kemiripan menggunakan *cosine similiarity*.

El Emary (2005) membuat sistem temu kembali informasi otomatis untuk mengatasi data berbahasa Arab dan merepresentasikan perbandingan antara hasil *retrieval* menggunakan *vector space model* dalam 2 metode pengindeksan yang berbeda yaitu pengindeksan seluruh kata dan pengindeksan akar kata. Sistem temu kembali informasi otomatis dibangun menggunakan teknik pemodelan tradisional yaitu *vetor space model* yang menggunakan pengukuran *cosine similiarity*. Dimana pemodelan tersebut memberikan kinerja *retrieval* yang lebih baik melalui perangkingan data yang telah ditemukan secara *descending* menurut derajat kemiripan. Hasil *output* menunjukkan bahwa pengindeksan akar kata meningkatkan kinerja *retrieval* lebih baik dari pada pengindeksan seluruh kata (*full-word indexing*) pada dokumen berbahasa Arab, dan mengurangi kapasitas penyimpanan data dan meminimumkan waktu pemrosesan.

Al-Taani (2010) telah meneliti pencarian tentang konsep dan *keywords* pada Al-Quran. menggunakan algoritma *light stemming* untuk menemukan *stem* kata. Menggunakan pendekatan berbasis teks, stem, dan sinonim yang masing-masing pendekatan tersebut digunakan untuk menganalisa *query*. Pada pendekatan sistem berbasis *stem*, metode pencarian yang digunakan berdasarkan akar kata, *query* sebelumnya melalui langkah *preprocessing* (*stopword*, *normalization*, dan *stemming*). Hasil pengujian menunjukkan bahwa menggunakan sistem berbasis sinonim memiliki nilai presisi yang lebih tinggi dari pada menggunakan sistem berbasis *stem* dan sistem berbasis teks. Ketika mencari beberapa kata, nilai rata-rata presisi sebesar 0.92, sedangkan nilai rata-rata presisi sistem berbasis stem sebesar 0.80 dan nilai rata-rata presisi sebesar 0.77.

Mahmoud (2009) menerapkan metode pengindeksan dan sistem *Retrieval* berdasarkan N - gram untuk hukum bahasa Arab yang digunakan dalam dokumen jurnal resmi pemerintah Lebanon. Dalam hal ini N–gram berfungsi sebagai metode representasi berdasarkan kata-kata dan karakter dan kemudian membandingkan hasil menggunakan *vector space model* dengan tiga langkah pengukuran kemiripan: pembobotan TF*IDF, koefisien *Dice* dan *Cosine* Koefisien. Percobaan menunjukkan penggunaan trigram dokumen indeks Arab adalah pilihan yang optimal untuk pencarian informasi Arab menggunakan N-gram.

Murugesan dkk (2009) telah meneliti *term weighting* berbasis *cluster* untuk klusterisasi dokumen berbasis *Term Frequency - Inverse Document Frequency* (TF-IDF). Metode ini menetapkan *term weighting* menggunakan informasi yang di peroleh dari cluster yang produksi dan koleksi. Metode tersebut

dapat mengenali kata pada suatu *clester* dan menambah *term weighting* berdasarkan kepentingannya. Pembobotan ini dapat meningkatkan hasil *entropy* rata-rata dari algoritma *K-means*.

Ren (2013) memperkenalkan pembobotan berdasarkan indeks kelas TF.IDF.ICF yang menggabungkan *Inverse Class Frequency* (ICF). Pada percobaan ini menggunakan dataset Reuters-21578, 20 *News Groups* dan RCV1-v2 yang memberikan nilai pembeda positif pada *term* yang jarang muncul dalam ruang vektor. Kemudian merevisi fungsi ICF dan menerapkan metode *Inverse Class Space Density Frequency* ($ICS_{\delta}F$), dan menghasilkan metode TF.IDF.$ICS_{\delta}F$ yang menyediakan diskriminasi positif pada *term* yang jarang dan sering muncul. Hasil eksperimen menunjukkan metode pengindeksan berbasis kelas TF.IDF.$ICS_{\delta}F$ dapat mengatasi masalah yang berdimensi tinggi.

2.2 *Information Retrieval*

Istilah *Information Retrieval* diciptakan oleh Mooers pada tahun 1951. *Information Retrival* merupakan bidang persimpangan antara ilmu informasi dan ilmu komputer (Hersh, 2003). ISO 2382/1 mendefinisikan *Information Retrieval* (IR) sebagai tindakan, metode dan prosedur untuk menemukan kembali data yang tersimpan, kemudian menyediakan informasi mengenai subyek yang dibutuhkan. Tindakan tersebut mencakup *text indexing*, *inquiry analysis*. Data mencakup teks, tabel, gambar, ucapan dan video. Informasi *term*asuk pengetahuan terkait yang dibutuhkan untuk mendukung penyelesaian masalah dan akuisi pengetahuan (Cios, 2007). Tujuan dari sistem *information retrieval* adalah memenuhi kebutuhan informasi pengguna dengan me-*retrieve* semua dokumen yang relevan, pada waktu

yang sama me-*retrieve* sesedikit mungkin dokumen yag tak relevan. Sistem ini menggunakan fungsi heuristik untuk mendapatkan dokumen-dokumen yang relevan dengan *query* pengguna. Sistem *information retrieval* yang baik memungkinkan menentukan secara cepat dan akurat apakah isi dari dokumen yang diterima memenuhi kebutuhannya. Agar representasi dokumen lebih baik, dokumen-dokumen dengan topik atau isi yang mirip dikelompokkan bersama-sama (Murrad dkk, 2002).

Secara garis besar, dua pekerjaan yang ditangani oleh sistem *information retrieval* yaitu melakukan *preprocessing* terhadap *database* dan kemudian menerapkan metode tertentu untuk menghitung kedekatan antara dokumen didalam *database* yang telah di *preprocess* denga *query* pengguna. Pada tahapan dasar *preprocessing*, sistem ini berurusan dengan dokumen semi-*structured* biasanya memberikan *tag term-term* atau bagian dari dokumen, sedangkan pada dokumen yang tidak terstruktur proses ini dilewati dan memberikan *term* tanpa imbuhan *tag*. *Query* yang dimasukkan pengguna dikonversi sesuai aturan tertentu untuk mengekstrak *term-term* penting. *Term-term* yang sebelumnya telah di ekstrak dari dokumen dan menghitung relevansi antara *query*. Dan dokumen berdasarkan pada *term-term* tersebut. Sebagai hasilnya, sistem mengembalikan suatu daftar dokumen terurut *descensing* (*rangking*) sesuai nilai kemiripannya dengan *query* pengguna.

Setiap dokumen direpresentasikan menggunakan model *bag-of-words* yang mengabaikan urutan dari kata-kata didalam dokumen, struktur sintaksis dari dokumen dan kalimat. Dokumen ditransformasi kedalam suatu wadah berisi kata-kata independen. *Term* disimpan dalam suatu *database* pencarian khusus yang

ditata sebagai *inverted index*. Indeks ini merupakan konversi dari dokumen asli yang mengandung sekumpulan kata kedalam daftar kata yang berasosiasi dengan dokumen terkait dimana kata-kata tersebut muncul.

Begitu juga pada penggunaan data teks berbahasa Arab, proses-proses yang dilakukan juga sama seperti pada penggunaan data dalam bahasa lain. Namun dalam tahap *preprocessing*, dokumen-dokumen tersebut memerlukan penanganan khusus. Bahasa Arab terdiri dari 28 huruf yang terdiri dari kanan ke kiri serta memiliki morfologi yang sangat kompleks. Ada banyak masalah dalam *information retrieval* bahasa Arab seperti variasi pengucapan kata-kata tertentu, bentuk kata yang *irregular* dan hasil turunan, panjang pendek pengucapan huruf serta hampir semua kata-katanya mengandung imbuhan.

2.3 Morfologi Bahasa Arab

Bahasa Arab (عربي - ʻArabī) termasuk dalam rumpun bahasa Semitik dan berkerabat dengan bahasa Ibrani dan bahasa-bahasa Neo Arami. Bahasa Arab memiliki lebih banyak penutur daripada bahasa-bahasa lainnya dalam rumpun bahasa Semitik. Bahasa ini dituturkan oleh lebih dari 300 juta orang sebagai bahasa utama (Khresiat, 2009), yang sebagian besar tinggal di Timur Tengah dan Afrika Utara. Bahasa ini adalah bahasa resmi dari 25 negara, dan merupakan bahasa peribadatan dalam Islam karena merupakan bahasa yang dipakai oleh Al-Qur'an. Berdasarkan penyebaran geografisnya, bahasa Arab percakapan memiliki banyak variasi (dialek), beberapa dialeknya bahkan tidak dapat saling mengerti satu sama lain. Bahasa Arab modern telah diklasifikasikan sebagai satu makrobahasa dengan 27 sub-bahasa dalam ISO 6393. Bahasa Arab Baku (kadang- kadang disebut Bahasa

Arab Sastra) diajarkan secara luas di sekolah dan universitas, serta digunakan di tempat kerja, pemerintahan, dan media massa. Jazirah Arabia merupakan tempat lahirnya bahasa Arab. Ia terbagi atas dua bagian yaitu bahasa Arab fasih dan bahasa Arab sehari-hari atau dialek lahjatun. Bahasa Arab fasih dapat dibagi lagi menjadi dua bagian yaitu bahasa Arab klasik dan bahasa Arab modern. Bahasa Arab klasik adalah bahasa formal yang digunakan di kawasan Hejaz. Sampai saat ini masih terdapat catatan tertulis yang berkaitan dengan penggunaan bahasa Arab klasik, termasuk di dalamnya syair-syair Arab yang amat terkenal pada masa pra Islam. Al-Qur`an juga diturunkan dalam bahasa Arab klasik tersebut, dan hal inilah yang menjadi alasan utama mengapa bahasa ini terjaga keasliannya sepanjang masa.

Bahasa Arab modern sama dengan bahasa klasik, dan merupakan bahasa resmi 22 negara Arab, baik untuk percakapan maupun tulisan. Perbedaannya hanya terletak pada perkembangan pembendaharaan kata, di mana pada bahasa Arab modern perkembangan pembendaharaan kata mengiringi perkembangan zaman, sedangkan bahasa Arab klasik mengacu pada adat kebiasaan lama, dan lebih sering digunakan dalam penyampaian berita atau dalam penulisan koran.

Bahasa Arab Baku berasal dari bahasa Arab klasik, satu-satunya anggota rumpun bahasa Arab Utara Kuna yang saat ini masih digunakan, sebagaimana terlihat dalam inskripsi peninggalan Arab pra-Islam yang berasal dari abad ke-4. Bahasa Arab Klasik juga telah menjadi bahasa kesusasteraan dan bahasa peribadatan Islam sejak lebih kurang abad ke-6. Abjad Arab ditulis dari kanan ke kiri. Penulisan abjad Arab sendiri berbeda dengan bahasa latin, dalam bahasa Arab

terdapat beberapa huruf yang penulisannya jika bertemu dengan huruf lain akan mengalami perubahan. Lebih jelasnya dapat dilihat pada Gambar 2.1.

final	medial	initial	isolated	IPA
ـط	ـطـ	طـ	ط	tˤ
ـع	ـعـ	عـ	ع	ʕ
ـف	ـفـ	فـ	ف	f
ـق	ـقـ	قـ	ق	q
ـك	ـكـ	كـ	ك	k
ـگ	ـگـ	گـ	گ	g
ـل	ـلـ	لـ	ل	l

Gambar 2.1 Perubahan penulisan abjad Arab

Secara gramatikal, kata benda dalam bahasa Arab bisa mengalami keadaan rofa', nashob, dan jar. Contoh keadaan rofa' dari suatu kata benda adalah pada saat berkedudukan sebagai subyek, sedangkan nashob misalnya pada saat jabatan kata tersebut adalah sebagai obyek penderita. Keadaan jar salah satunya dialami suatu kata jika didahului oleh sebuah preposisi. Kata benda dalam bahasa Arab juga dibedakan menjadi kata benda tertentu (ma'rifah), dan umum (nakirah)(Dayyab dkk, 1993).

Dalam struktur bahasa Arab, dikenal 2 disiplin ilmu utama, yaitu : nahwu dan shorof/ tashrif. Nahwu adalah metode analisis jenis kata dalam suatu kalimat. Dengan nahwu, kedudukan / jabatan kata dalam kalimat dapat diidentifikasi. Sedangkan ilmu tashrif adalah ilmu yang mempelajari kemungkinan perubahan suatu kata sebelum kata tersebut masuk dalam suatu kalimat.

Ilmu tashrif mempelajari perubahan bentuk dan pola suatu kata. Teks kamus bahasa Arab seringkali menempatkan susunan suatu kata dalam urutan berdasarkan kata dasarnya. Kata dasar dalam kamus tersebut biasanya adalah kata kerja bentuk lampau (*fi'il madhi*) atau kata benda bentukan dari kata kerja (*isim mashdar*). Sebagai contoh, untuk mencari kata يَغْسِلُ dalam kamus, seseorang tidak bisa mencarinya dalam deretan kata berawalan ي (huruf ya'), karena susunan kata diurutkan menurut kata dasarnya. Kata يَـغْسِـــلُ bermakna '(dia laki-laki) sedang mencuci' berasal dari kata dasar غَسَــلَ yang artinya 'dia laki-laki telah mencuci'. Karena itu, kata tersebut dalam teks tertulis kamus Arab biasanya diletakkan dalam deretan kata berawalan huruf غ (ghain), bukan huruf ي (ya'). Untuk menganalisis suatu kata menjadi kata dasarnya ini diperlukan pengetahuan dalam ilmu tashrif. Ilmu *tashrif* .

Ilmu tasrif mengelompokkan mayoritas kata kerja dalam bahasa Arab memiliki 3 huruf utama. Karena itu, 3 huruf utama inilah yang dianggap sebagai dasar susunan huruf bagi susunan huruf kata kerja yang lain. Dasar susunan huruf susunan huruf tersebut adalah فَعَلَ . Merujuk pada dasar susunan huruf ini, huruf inti pertama dianggap sebagai ف, huruf inti ke-2 dianggap sebagai huruf ع , dan huruf inti ke-3 dianggap sebagai ل. Sebagai contoh, kata: كَتَبَ yang artinya : 'telah menulis' merupakan morfologi turunan dari dasar susunan huruf فَعَلَ (Dayyab dkk, 1993).

Gambar 2.2 Contoh pemasangan dasar susunan huruf

Pada Gambar 2.2 menjelaskan pensejajaran masing-masing huruf pembentuk kata dengan huruf pada dasar susunan hurufnya. Huruf ك adalah huruf pertama, sehingga merupakan turunan dari huruf ف . Huruf ت adalah huruf kedua dalam susunan huruf, merupakan representasi dari dasar susunan huruf ع. Sedangkan huruf terakhir ب merupakan pasangan dari huruf dasar susunan huruf ل. Dasar susunan huruf huruf penyusun tersebut dalam istilah tashrif disebut sebagai wazan, yang bermakna timbangan. Artinya, semua huruf penyusun suatu kata akan direferensikan pada wazan/ timbangannya yang sesuai.

Jumlah *wazan* sebagai referensi dalam *tashrif* ada 35 buah (Maksum, 1965). Kata yang dijadikan sebagai acuan dalam *wazan* adalah kata kerja bentuk lampau (*fi'il madhi*) dan kata kerja sekarang/akan datang (*fi'il mudhari'*). Dua jenis kata yang dijadikan acuan sebagai *wazan* adalah kata kerja lampau (*fi'il madhi*) dan kata kerja masa sekarang atau akan datang (*fi'il mudhari'*). Berdasarkan jumlahnya, kata dalam bahasa Arab dibedakan menjadi *mufrad* (tunggal), *mutsannaa* (dua), dan jama' (lebih dari 2). Suatu kalimat atau frase dalam bahasa Arab akan tersusun dari tiga unsur pembentuknya. Ketiga macam itu adalah kata benda (*isim*), kata kerja (*fi'il*), dan huruf. *Isim* adalah kata yang menunjukkan pada sesuatu namun tidak

terikat dengan waktu, sedangkan *fi'il* adalah kata yang menunjukkan sesuatu pekerjaan yang terikat dengan waktu tertentu (telah selesai, sedang berlangsung, atau akan dilakukan). Huruf sebagai unsur pembentuk suatu kata, yang dimaksud bukanlah huruf *hijaiyyah*. Unsur pembentuk kata ini dibedakan dari *isim* dan *fi'il* karena tidak memiliki karakteristik seperti dua unsur pembentuk tersebut (Chen, 2002).

2.4 *Preprocessing* Teks Bahasa Arab

Preprocessing merupakan tahap yang penting dalam pemrosesan teks untuk memperoleh fitur yang akan diproses pada tahap selanjutnya. Tugas pokok pada tahap ini adalah pembangunan indeks dari koleksi dokumen. *Indexing* adalah proses membangun representasi suatu dokumen dengan memberikan suatu pengenal pada item-item teks. Kualitas indeks mempengaruhi efektifitas dan efisiensi sistem *information retrieval*. Elemen dari *indexing* adalah fitur (*keyword*, *term*, dan istilah), yang dapat diperoleh dari ekstraksi teks suatu dokumen yang akan dimodelkan, atau bisa juga diperoleh secara independen. Indeks dokumen adalah himpunan *term* yang menunjukkan isi topik yang dikandung oleh dokumen. Indeks akan membedakan antara satu dokumen dengan satu dokumen lain yang berada didalam koleksi. Ukuran indeks yang kecil dapat memberikan hasil buruk dan mungkin beberapa item yang relevan justru terabaikan. Indeks yang besar memungkinkan ditemukannya banyak dokumen yang relevan tetapi sekaligus menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pemrosesan (Machnik, 2004).

Untuk dokumen teks, setiap kata didalam dokumen dapat dianggap sebagai *term*, setelah kata tersebut mengalami *preprocessing*. *Preprocessing* berkaitan erat dengan penerapan proses *indexing* pada suatu *information retrieval*. Suatu skema *preprocessing* biasanya tidak hanya bagaimana mempresentasikan suatu dokumen teks, tapi juga bagaimana menghasilkan solusi yang efektif dan efisien. Pembuatan indeks melibatkan konsep *linguistic preprocessing* yaitu pemrosesan *term* sesuai dengan struktur bahasa yang bertujuan mengekstrak *term-term* penting dari dokumen yang direpresentasikan sebagai *bag of words*. Urutan langkah langkah pembangunan indeks dengan implementasi *linguistic processing* dengan menggunakan sebuah dokumen teks Arab adalah sebagai berikut (Mesleh, 2008) (Hmeidi, 1997)

قُلْ هُوَ اللَّهُ أَحَدٌ اللَّهُ الصَّمَدُ لَمْ يَلِدْ وَلَمْ يُولَدْ وَلَمْ يَكُنْ لَهُ كُفُوًا أَحَدٌ

"Katakanlah, Dialah Allah Yang Maha Esa. Allah tempat bergantung bagi segala sesuatu. Dia tidak beranak dan tidak diperanakkan. Dan tidak ada sesuatupun yang menyamai-Nya"

1. Pemisah rangkaian kata (*tokenizing*)

Tahap ini berfungsi memisahkan deretan kata didalam kalimat, paragraf, maupun halaman menjadi *token* atau potongan kata tunggal atau *termed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti digit, angka, tanda hubungan dan tanda baca. Maka dari dokumen tersebut akan menghasilkan.

– (akhad) **أَحَدٌ** – (Allahu) **اللَّهُ** – (huwa) **هُوَ** – (qul) **قُلْ**

– (yalid) **يَلِدْ** – (lam) **لَمْ** – (Shomada) **الصَّمَدُ** – (Allahu) **اللَّهُ**

– (yakun) **يَكُنْ** – (walam) **وَلَمْ** – (yulad) **يُولَدْ** – (walam) **وَلَمْ**

(akhad) **أَحَدٌ** – (kufuwan) **كُفُوًا** – (lahu) **لَهُ**

2. *Normalization* dan *filtration*

Tahap ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain didalam koleksi. Selain itu, juga dilakukan penghilangan karakter-karakter yang tidak *term*asuk huruf hijaiyah (**ا, ب, ... ,ت**), kemudian dilakukan penghapusan *diartic* (harokat), dan juga menormalkan teks Arab ke bentuk dasar. Dan semua teks-teks bukan Arab dihapus. Dalam tahapan ini, hasil dari *tokenization* akan diproses dan menghasilkan:

قل – هو – الله – احد – الله – الصمد – لم –

يلد – لم – يولد – يكن – له – كفوا – احد

Setelah dilakukan proses *normalization* dan *filtration* data yang dihasilkan berupa potongan kata tunggal tanpa harokat, tanpa tanda baca, dan dikembalikan ke bentuk normal huruf *hijaiyah*.

3. *Stopword Removal*

Term yang efektif dalam pemisahan dokumen yang relevan dari dokumen yang tidak relevan kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. Ini berarti bahwa *term* dengan frekuensi kemunculan tinggi tidak memberikan nilai informasi yang tinggi. Kedua, *term* yang muncul dalam banyak dokumen tidak mencerminkan definisi dari topik atau sub-topik dokumen. Seperti kata fungsi bahasa Arab (... ,**أحد** ,**أبدا** ,**آخر**), kata ganti, artikel dan preposisi di hapus. Karena itu, *term-term* ini dimasukkan dalam daftar *stopword* atau *stoplist* dan dihapus. Strategi umum penentuan *stoplist* adalah mengurutkan *term* berdasarkan

frekuensi koleksi (jumlah total kemunculan setiap *term* di dalam koleksi dokumen) dan memasukkan *term* yang paling sering muncul sebagai *stopword*. Proses *stopword removal* adalah sebagai berikut :

قل – هو – الله – احد – الله – الصمد – لم –

يلد – لم – يولد – لم – يكن – له – كفوا – احد

Kata yang bergaris bawah diatas termasuk kata dalam daftar *stopword*, pada proses ini kata-kata tersebut akan dihapus dan menghasilkan *output* sebagai berikut :

قل– الصمد – يلد – يولد – كفوا

Setelah dilakukan penghapusan *stopword* tinggal menyisakan kata-kata yang memiliki pengaruh dalam suatu dokumen seperti hasil proses *stopword removal* diatas.

4. Konversi *term* ke bentuk dasar (*stemming*)

Stemming adalah proses konversi *term* ke bentuk dasarnya. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi *term-term* tertentu di dalamnya. Sinonim adalah kata-kata yang mempunyai pengertian serupa tetapi berbeda dari sudut pandang morfologis. Dalam tahapan ini setiap kata yang telah dihasilkan dari serangkaian proses sebelumnya kemudian dirubah menjadi kata dasar, dan akan dihasilkan *output* sebagai berikut :

قال – صمد – ولد – ولد – كفا

Setelah proses stemming selesai dilakukan, maka didapat kata-kata dalam bentuk dasar seperti diatas. Proses *stemming* ini mempunyai peranan penting dalam *information retrieval*, karena proses ini tidak hanya mengumpulkan kata yang sama, tetapi bisa mengurangi jumlah *term* yang merepresentasikan suatu dokumen

tertentu, sehingga penyimpanan data *term* akan lebih sedikit dan proses yang dijalankan juga akan lebih cepat. *Light stemming* merupakan salah satu *stemmer* teks Arab (Larkey, 2007). *Stemmer* inilah yang digunakan pada *library Lucene* apabila menggunakan *class ArabicAnalyzer*.

2.5 Perangkingan Dokumen Teks Bahasa Arab

Perangkingan dokumen menggunakan representasi *vector space model* dari kumpulan dataset. Dokumen dalam *vector space model* di representasikan dalam matriks yang berisi bobot kata dalam dokumen. Bobot tersebut menyatakan kepentingan atau kontribusi kata terhadap suatu dokumen dan kumpulan dokumen. Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya terhadap dokumen. Biasanya kata yang berbeda memiliki frekuensi yang berbeda.

Dari pembobotan tersebut diperoleh bobot kata pada dokumen yang digunakan sebagai representasi vektor. Dari representasi bobot tersebut dapat dihitung nilai kemiripan suatu dokumen dengan *query*. Nilai kemiripan ini biasanya dihitung dengan rumus *cosine similiarity*, perhitungan tingkat kemiripan ini dibuat dengan berdasarkan besar sudut kosinus antara dua vektor, dalam hal ini adalah vektor dokumen. Hasil dari perhitungan jarak kosinus antara tiap dokemen terhadap *query* inilah yang digunakan untuk merangking dokumen.

2.4.1 *Term Weighting*

Dibawah ini adalah beberapa *term weighting* yang digunakan (Ren, 2013):

- *Term Frequency*

  Term frequency merupakan metode yang paling sederhana dalam membobotkan setiap *term*. Setiap *term* diasumsikan memiliki kepentingan yang proporsional

terhadap jumlah kemunculan *term* pada dokumen. Bobot dari *term* t pada dokumen *d* dengan *f(dj,ti)* adalah frekuensi kemunculan *term t* ke-*i* pada dokumen *d* ke-*j*.

$$TF(t_i, d_j) = f(d_j, t_i) , \quad (2.1)$$

- *Inverse Document Frequency (IDF)*

  Bila *term frequency* memperhatikan kemunculan *term* di dalam dokumen, maka IDF memperhatikan kemunculan *term* pada kumpulan dokumen. Latar belakang pembobotan ini adalah *term* yang jarang muncul pada kumpulan dokumen sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah dokumen yang mengandung *term*. Perhitungan bobot IDF dari *term* t.

$$IDF(t_i, d_j) = 1 + log \frac{D}{d(ti)} , \quad (2.2)$$

- *Inverse Class Frequency (ICF)*

  Dalam pengindeksan dokumen klasik terdapat fungsi yang sering digunakan yaitu ICF. Pemetaan *category* bernilai 1 ketika *term* sesuai dengan *category* tertentu c_k , dan bernilai 0 jika term tidak sesuai. Dalam ICF term yang ditemukan pada banyak kelas tidak bisa memberikan nilai pembeda yang baik, yang menyebabkan fungsi tersebut memberikan nilai yang rendah pada *term* yang muncul pada banyak kelas.

$$W_{c_k}(t_i) = \begin{cases} 1, jika\ term\ muncul\ dalam\ class \rightarrow\ t_i\ \in |c_k| \\ 0, selainya \end{cases} \quad (2.3)$$

  Perhitungan bobot ICF dari *term* t adalah sebagai berikut:

$$W_{ICS_\delta F}(t_i, d_j, c_k) = 1 + log \frac{C}{c(t_i)} \quad (2.4)$$

Dimana C mengindikasikan kesuluruhan kelas dalam koleksi, $c(t_i)$ adalah jumlah kelas dalam koleksi yang didalamya terdapat *term* t_i, $\frac{c(t_i)}{C}$ sebagai CF, dan $\frac{C}{c(t_i)}$ sebagai ICF dari t_i.

- *Inverse Class Space Density Frequency ($ICS_\delta F$)*

  Dalam ICF term yang ditemukan pada banyak kelas tidak bisa memberi nilai pembeda yang baik, akibatnya fungsi ICF memberikan nilai yang rendah pada term yang muncul pada banyak kelas. Bila ICF memperhatikan $ICS_\delta F$ memperhatikan kemunculan *term* pada kumpulan dokumen yang menjadi anggota *class/category*. *Term* yang jarang muncul pada banyak dokumen anggota *class* adalah *term* yang sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah dokumen pada class yang mengandung *term*. Perhitungan $ICS_\delta F$ ini dapat diadopsi langsung dari ICF, akan tetapi ICF hanya menghiraukan *term* yang ada pada *class* tanpa menghiraukan jumlah *term* yang terdapat dalam dokumen yang menjadi anggota class. Perhitungan bobot $ICS_\delta F$ dari *term t* tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah *term* pada dokumen anggota kelas yang mengandung *term*. Dalam class density (C_δ) pengambilan nilai direpresentasikan seperti berikut:

$$C_\delta(t_i) = \frac{n_{c_k}(t_i)}{N_{c_k}} \tag{2.5}$$

  Dimana $n_{c_k}(t_i)$ mengindikasikan jumlah keseluruhan dokumen yang di dalam-nya terdapat *term* t_i dan merupakan anggota kelas c_k, dan N_{c_k} mengindikasikan jumlah keseluruhan dokumen yang menjadi anggota kelas c_k. Sedangkan nilai

space class density merupakan nilai dari hasil penjumlahan kelas yang didapat dari class density (C_δ).

$$CS_\delta(t_i) = \sum_{c_k} C_\delta(t_i) \quad (2.6)$$

Perhitungan bobot $ICS_\delta F$ dari *term* t_i adalah sebagai berikut:

$$W_{ICS\delta F}(t_i, d_j, c_k) = 1 + log \frac{C}{cs\delta(ti)} \quad (2.7)$$

Dimana $\frac{CS_\delta(t_i)}{C}$ merujuk sebagai *class space density frequency* ($CS_\delta F$) dan $\frac{C}{CS_\delta(t_i)}$ merupakan inverse class space density frequency ($ICS_\delta F$) dari *term* t_i.

Gabungan dari masing-masing pembobotan tersebut memberikan variasi pembobotan kata. Dalam penelitian ini menggunakan pembobotan $TF.IDF.ICS_\delta F$ yang merupakan perkalian dari bobot TF, IDF dan $ICS_\delta F$ dari suatu *term* terhadap dokumen dan kelompok kepadatan kelas tertentu, sebagaimana yang didefinisikan pada persamaan berikut:

$$W_{TF.IDF.ICS\delta F}(t_i, d_j, c_k) = W_{TF}(t_i, d_j) \text{ x } W_{ICS\delta F}(t_i, d_j, c_k)$$

atau,

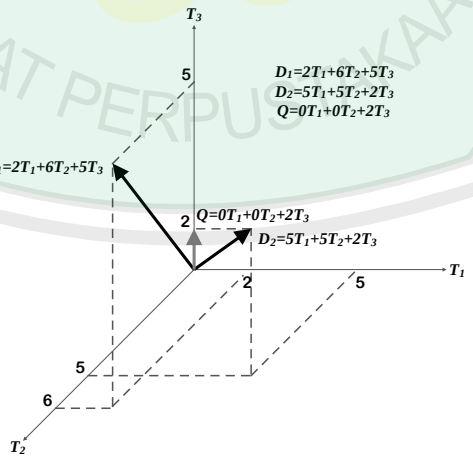
$$W_{TF.IDF.ICS\delta F}(t_i, d_j, c_k) = tf(ti, dj)\left(1 + log \frac{D}{d(ti)}\right) x \left(1 + log \frac{C}{cs\delta(ti)}\right) \quad (2.8)$$

Dengan perumusan tersebut maka bobot akan semakin tinggi saat lebih banyak ditemukan didalam satu dokumen (indikasi frekuensi *term*). *Term* yang semakin muncul pada satu dokumen, tapi jarang muncul pada seluruh dataset akan diberikan nilai bobot yang lebih tinggi (Salton,1989).

2.4.2 *Vector Space Model* (VSM)

Pada *Vector space model*, setiap dokumen di dalam database dan *query* pengguna direpresentasikan oleh suatu vektor multi-dimensi (Polettini, 2004; Cios, 2007). Dimensi sesuai dengan jumlah *term* dalam dokumen yang terlibat. Contoh dari model ruang vektor tiga dimensi untuk dua dokumen D_1 dan D_2, satu *query* pengguna Q_1, dan tiga *term* T_1, T_2 dan T_3 diperlihatkan pada Gambar 2.1. Pada model ini:

a. *Vocabulary* merupakan kumpulan semua *term* berbeda yang tersisa dari dokumen setelah *preprocessing* dan mengandung *t term* indeks. *Term-term* ini membentuk suatu ruang vektor.
b. Setiap *term i* di dalam dokumen atau *query j*, diberikan suatu bobot (*weight*) bernilai *real* w_{ij}.
c. Dokumen dan *query* diekspresikan sebagai vektor *t* dimensi *dj* = (w_1, w_2,..., w_{tj}) dan terdapat n dokumen di dalam koleksi, yaitu *j* = 1, 2,..., n.



Gambar 2.3 Model ruang vektor (Polettini, 2004; Cios, 2007)

Dalam model ruang vektor, koleksi dokumen direpresntasikan oleh matriks *term-document* (atau matriks *term-frequency*). Setiap sel dalam matriks bersesuaian dengan bobot yang diberikan dari suatu *term* dalam dokumen yang ditentukan. Nilai nol berarti bahwa *term* tersebut tidak hadir di dalam dokumen (Cios, 2007). Pada Gambar 2.2 ditunjukkan contoh matriks *term-document* untuk *database* dengan *n* dokumen dan *t term*.

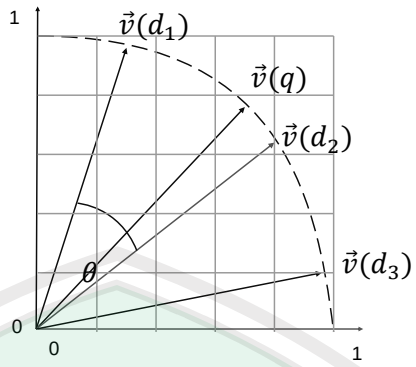
$$\begin{bmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{11} & w_{21} & \dots & w_{t2} \\ \dots & \dots & \dots & \dots & \dots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{bmatrix}$$

Gambar 2.4 Matriks *term-document* (Cios, 2007)

Keberhasilan dari model VSM ini ditentukan oleh skema pembobotan terhadap suatu *term* baik untuk cakupan lokal maupun global, dan faktor normalisasi. Pembobotan lokal hanya berpedoman pada frekuensi munculnya *term* dalam suatu dokumen dan tidak melihat frekuensi kemunculan *term* tersebut didalam dokumen lainnya.

2.4.3 *Cosine Similiarity*

Salah satu ukuran kemiripan teks yang populer (Tata, 2007) adalah *cosine similiarity*. Ukuran ini menghitung nilai kosinus sudut antara dua vektor. Dalam gambar 2.3 terdapat tiga vektor dokumen *d1*, *d2*, *d3* dan satu vektor *query q*. *Cosine similiarity* menghitung nilai kosinus θ dari *query* dan tiga dokumen lain. Nilai ini menunjukkan derajat kemiripan dokumen dengan *query*.



Gambar 2.5 Representasi perumusan *Cosine Similiarity* (Tata, 2007)

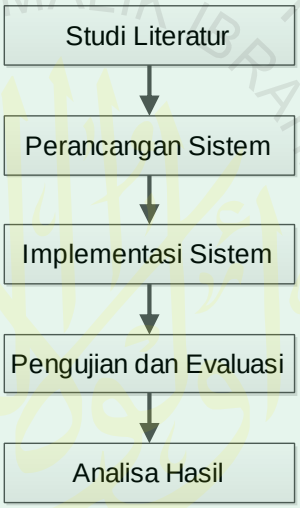
Karena berdasarkan sudut antar dua vektor, maka nilainya berkisar pada 0 sampai dengan 1, dimana 0 menandakan bahwa kedua dokumen tidak mirip sama sekali, dan 1 menandakan bahwa antara *query* dan dokumen benar-benar identik. *Cosine* dinyatakan sebagai persamaan 2.9 berikut. Pada persamaan tersebut cos(*q*, d_j) adalah nilai *cosinus* anatara *query* dan dokumen *j*. $TF.IDF(t_k, q)$ dan $TF.IDF(t_k, d_j)$ masing-masing menunjukkan pembobotan *TF.IDF* kata t_k pada query dan pembobotan *TF.IDF* kata t_k pada dokumen *j*, sedangkan $|TF.IDF_q|$dan $|TF.IDF_d|$ masing-masing merupakan panjang dari vektor *query q* dan panjang dari dokumen *j*.

$$\cos(q, dj) = \frac{\sum_{t_i}[TF.IDF(t_k,q)].[TF.IDF(t_k,d_j)]}{\sqrt{\sum |TF.IDF_q|^2}.\sqrt{\sum |TF.IDFd_j|^2}} \quad (2.9)$$

BAB III

METODE PENELITIAN

Adapun tahapan-tahapan yang akan dilakukan pada penelitian ini meliputi (1) Studi Literatur, (2) Perancangan sistem, (3) Implementasi Sistem, (4) pengujian dan Evaluasi, serta (5) Analisa Hasil. Alur tahapan-tahapan tersebut dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Metode Penelitian

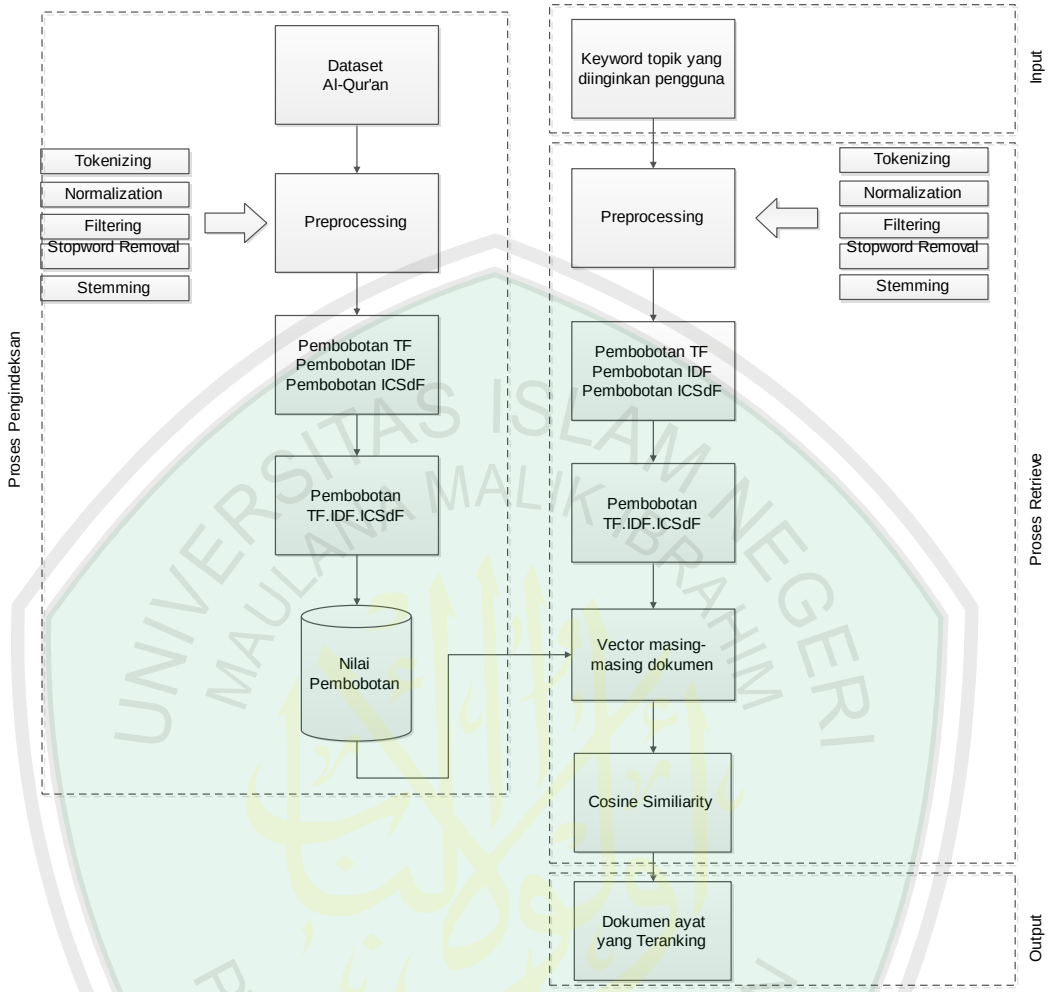
3.1 Studi Literatur

Studi literatur dilakukan untuk mendapatkan informasi yang berkaitan dengan lingkup pembahasan dalam penelitian, perkembangan keilmuan terkait, serta metode yang telah ada sebelumnya. Studi literatur yang dilakukan diharapkan dapat memberikan data, informasi, dan fakta mengenai perangkingan dokumen berbahasa Arab yang akan dikembangkan. Studi literatur yang dilakukan mencakup pencarian dan mempelajari referensi-referensi yang terkait, seperti:

1. *Text preprocessing* yaitu *tokenizing*, *stopword removal* dan *stemming* kata Arab menggunakan *Light Stemmer* (Larkey, 2002) pada *library* lucene (http://lucene.apache.org)
2. Metode *term weighting* TF.IDF. $ICS_{\delta}F$
3. Metode pengukuran kemiripan dokumen menggunakan *cosine similiarity*.
4. Evaluasi hasil perangkingan dokumen dengan perhitungan *recall*, *precission*, dan *f-measure*.

3.2 Perancangan Sistem

Alur proses dalam penelitian yang akan dilakukan terdiri dari beberapa tahapan sebagaimana yang tertera pada gambar 3.2. Dokumen al-Qur'an sebagai dataset melalui tahapan awal yaitu *preprocessing*. Pada tahap *preprocessing* dokumen dipotong menjadi potongan-potongan kata (*tokenizing*), menghilangkan kata yang *term*asuk *stoplist* (*stoppword removal*), dan dirubah menjadi akar kata (*stemming*). Setelah itu, melakukan pembobotan kata terhadap *term* hasil tahap *preprocessing*, mulai dari perhitungan nilai TF (*Term Frequency*) menggunakan persamaan 3.1, IDF (*Inverse Document Frequency*) menggunakan persamaan 3.2, dan $ICS_{\delta}F$ (*Inverse Class Space Density Frequency*) menggunakan persamaan 3.3. Masing-masing bobot TF, IDF, dan $ICS_{\delta}F$ dikalikan menggunakan persamaan 3.4. bobot tiap *term* dimodelkan dalam *vector space model* hingga dilakukan perhitungan jarak *cosinus* antar sumbu bobot *term* menggunakan ukuran kemiripan *cosine similiarity* menggunakan persamaan 3.5.



Gambar 3.2 Diagram Rancangan Sistem

3.2.1 Dataset

Data yang digunakan pada penelitian ini adalah dokumen Al-Qur'an berbahasa Arab yang diperoleh dari situs http://www.qurandatabase.org/, dimana dokumen tersebut berisi keterangan surat, ayat, dan teks isi, seperti yang ditunjukkan pada tabel 3.1.

Tabel 3.1 Dataset dokumen Al-Qur’an

id_doc	id_class	Surat	Ayat	Text
1	1	1	1	بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ
2	7	1	2	الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ
3	1	1	3	الرَّحْمَٰنِ الرَّحِيمِ
4	1	1	4	مَالِكِ يَوْمِ الدِّينِ
5	1	1	5	إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ
6	7	1	6	اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ
7	5	1	7	صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ
8	1	2	1	بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ الم
9	1	2	2	ذَٰلِكَ الْكِتَابُ لَا رَيْبَ ۛ فِيهِ ۛ هُدًى لِلْمُتَّقِينَ
10	1	2	3	الَّذِينَ يُؤْمِنُونَ بِالْغَيْبِ وَيُقِيمُونَ الصَّلَاةَ وَمِمَّا رَزَقْنَاهُمْ يُنْفِقُونَ

Adapun data yang digunakan sebagai indeks klasifikasi berasal dari index Al-Qur’an yang berisi topik-topik berupa pembahasan bangsa terdahulu, ibadah, sejarah, mu’amalat, makanan dan minuman, peradilan dan hakim, iman dll. Sedagkan dalam kelas tersebut terdapat rincian kelas dan ayat yang menjadi anggota kelas seperti yang ditunjukkan pada Tabel 3.2.

Tabel 3.2 Indeks klasifikasi Al-Qur’an

ID	Nama Kelas	Anggota
1	Iman	Qs.2:220; Qs.4:6; Qs.6:152; Qs.2:83; Qs.2:177; Qs.2:215; Qs.4:8; Qs.76:8; Qs.90:15
2	Ilmu	Qs.2:178; Qs.4:93; Qs.4:92
3	Bangsa Terdahulu	Qs.5:4; Qs.5:96; Qs.5:3; Qs.5:94
4	Sejarah	Qs.18:70; Qs.18:73; Qs.18:75; Qs.18:76; Qs.18:78; Qs.18:70; Qs.18:72; Qs.18:73;
5	Bangsa Terdahulu	Qs.46:13; Qs.9:40; Qs.9:61; Qs.10:65; Qs.11:54; Qs.20:34; Qs.24:37;
6	Akhlak dan Adab	Qs.31:15; Qs.34:9; Qs.38:24; Qs.38:34; Qs.39:8; Qs.39:17; Qs.39:54; Qs.40:13; Qs.42:10; Qs.42:13; Qs.50:8; Qs.50:33; Qs.60:4

7	Ibadah	Qs.8:30; Qs.9:40; Qs.9:61; Qs.10:65; Qs.10:71; Qs.10:76; Qs.10:78; Qs.11:12; Qs.11:27; Qs.11:32; Qs.11:53; Qs.11:54
8	Makanan dan Minuman	Qs.2:203; Qs.8:45; Qs.9:112; Qs.13:28; Qs.18:24; Qs.20:34; Qs.24:37
9	Pakaian dan Perhiasan	Qs.33:55; Qs.24:60; Qs.33:53; Qs.33:55; Qs.33:59
10	Hukum Privat	Qs.40:13; Qs.42:10; Qs.42:13; Qs.50:8; Qs.50:33; Qs.60:4
11	Muamalat	Qs.10:78; Qs.11:12; Qs.11:27; Qs.11:32; Qs.11:53; Qs.11:54
12	Peradilan dan Hakim	Qs.33:59; Qs.33:55; Qs.24:60; Qs.33:53
13	Hukum pidana	Qs.10:76; Qs.10:78; Qs.11:12; Qs.11:27; Qs.11:32; Qs.11:53; Qs.11:54
12	Jihad	Qs.4:95; Qs.4:100; Qs.5:54; Qs.8:60; Qs.8:72; Qs.33:59; Qs.33:55; Qs.24:60; Qs.33:53

3.2.2 Input

Input yang diguanakan yaitu *keyword* topik yang berupa tulisan Arab. *Keyword* yang di inputkan dapat berupa satu kata, kalimat ataupun satu bait ayat, seperti yang ditunjukkan pada tabel 3.3

Tabel 3.3 *Keyword* topik pembahasan

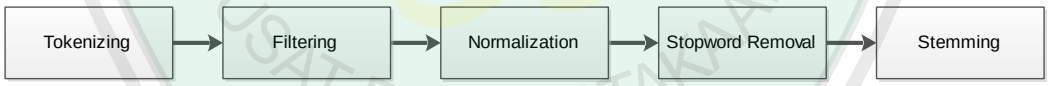
#	***Query***
Q1	**عبا دة الاصنام** Menyembah berhala
Q2	**طاعة الامير** Mematuhi pemimpin
Q3	**ستر العورة للمرأة** Munutup aurat bagi wanita
Q4	**القصاص فى القتلى** Qishaash berkenaan dengan orang-orang yang dibunuh
Q5	**الصديق** Jujur
Q6	**التوبة** Taubat

3.2.3 *Preprocessing*

Masing-masing dokumen melalui tahap *preprocessing*. Implementasi *preprocessing* terdiri dari beberapa tahapan, diantaranya adalah tokenisasi, *filtering*, normalisasi, *stopword removal* dan *stemming*. Tokenisasi dilakukan untuk memecah keseluruhan isi dokumen menjadi suku kata tunggal. Sedangkan pada tahapan *filtering* dilakukan pembuangan harokat-harokat bahasa Arab. Tahap

Penghilangan *stopword* dilakukan untuk menghilangkan kata-kata yang sering muncul dalam dokumen tetapi tidak mempunyai nilai yang berarti pada sebuah dokumen. Daftar kata *stopword* diambil dari website *http://Arabicstemmer.codeplex.com/*.

Tahap selanjutnya adalah *stemming* yang digunakan untuk memperoleh kata dasar dari masing-masing kata dengan cara mencari kata dasar (*root*) dan penghilangan *affix* serta *suffix*. Pada implementasi *stemming* ini digunakan *Light Stemmer* yang berada dalam library *lucene* (*http://lucene.apache.org/*).



Gambar 3.3 Diagram *Preprocessing*

3.2.4 *Term Weighting* (Pembobotan)

Tahap setelah *preprocessing* adalah *term weighting*. Pembobotan kata dilakukan dengan menghitung TF (*Term Frequency*) dan IDF (*Inverse Document Frequency*) dari masing-masing *term* pada seluruh dokumen. Kemudian perhitungan $ICS_{\delta}F$ (*Inverse Class Space Density Frequency*) juga dilakukan terhadap masing-masing *term* pada seluruh dokumen. Berikut ini adalah rumus perhitungan tahap pembobotan (Ren, 2013):

- ***Term Frequency***

  Term frequency merupakan metode yang paling sederhana dalam membobotkan setiap *term*. Setiap *term* diasumsikan memiliki kepentingan yang proporsional terhadap jumlah kemunculan *term* pada dokumen. Bobot dari *term* t pada dokumen *d* dengan *f(dj,ti)* adalah frekuensi kemunculan *term t* ke-*i* pada dokumen *d* ke-*j*.

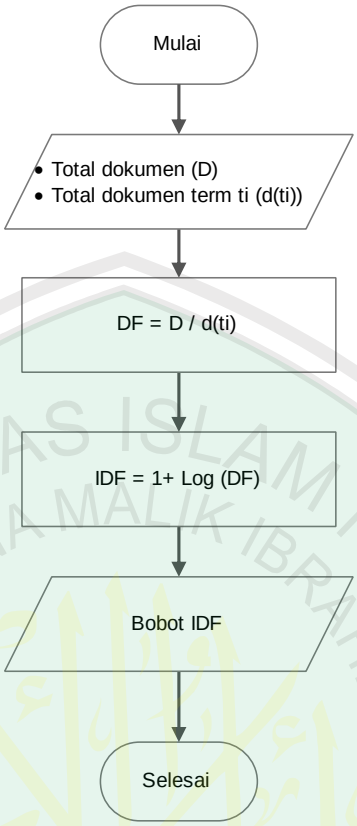
$$(t_i, d_j) = f(d_j, t_i) \quad (3.1)$$

- ***Inverse Document Frequency***

  Bila *term frequency* memperhatikan kemunculan *term* di dalam dokumen, maka IDF memperhatikan kemunculan *term* pada kumpulan dokumen. Latar belakang pembobotan ini adalah *term* yang jarang muncul pada kumpulan dokumen sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah dokumen yang mengandung *term*. Perhitungan bobot IDF dari *term* t.

$$W_{TF.IDF}(t_i, d_j) = 1 + log \frac{D}{d(ti)} \quad (3.2)$$

  Gambar 3.4 menunujukkan pembobotan IDF yang disajikan dalam bentuk *flowchart*.



Gambar 3.4 *Flowchart* IDF

Dimana D mengindikasikan kesuluruhan dokumen dalam koleksi, $d(t_i)$ adalah jumlah dokumen dalam koleksi yang didalamya terdapat *term* t_i, dan $\frac{D}{d(t_i)}$ sebagai DF dari t_i.

- ***Inverse Class Frequency***

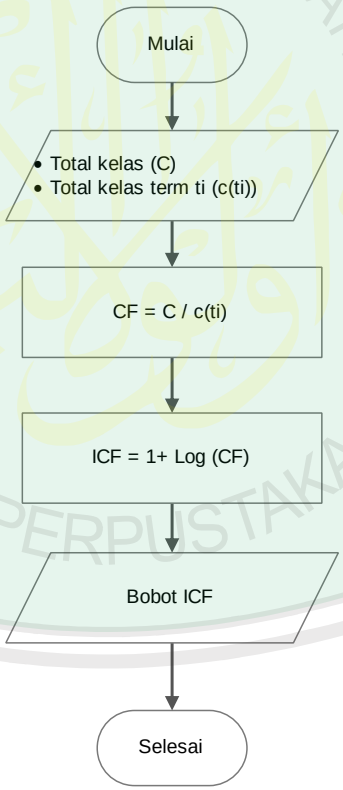
  Dalam pengindeksan dokumen klasik terdapat fungsi yang sering digunakan yaitu ICF. Pemetaan *category* bernilai 1 ketika *term* sesuai dengan *category* tertentu c_k, dan bernilai 0 jika term tidak sesuai. Dalam ICF term yang ditemukan pada banyak kelas tidak bisa memberikan nilai pembeda yang baik, yang menyebabkan fungsi tersebut memberikan nilai yang rendah pada *term* yang muncul pada banyak kelas.

$$W_{c_k}(t_i) = \begin{cases} 1, \text{jika term muncul dalam class} \rightarrow t_i \in |c_k| \\ 0, \text{selainnya} \end{cases} \quad (3.3)$$

Perhitungan bobot ICF dari *term* t adalah sebagai berikut:

$$W_{ICF}(t_i, d_j, c_k) = 1 + log \frac{C}{c(t_i)} \quad (3.4)$$

Dimana C mengindikasikan kesuluruhan kelas dalam koleksi, $c(t_i)$ adalah jumlah kelas dalam koleksi yang didalamya terdapat *term* t_i, $\frac{c(t_i)}{C}$ sebagai *CF*, dan $\frac{C}{c(t_i)}$ sebagai *ICF* dari t_i. Gambar 3.5 menunjukkan pembobotan ICF disajikan dalam bentuk *flowchart*.



Gambar 3.5 *Flowchart* ICF

- ***Inverse Class Space Density Frequency***

Dalam ICF term yang ditemukan pada banyak kelas tidak bisa memberi nilai pembeda yang baik, akibatnya fungsi ICF memberikan nilai yang rendah pada

term yang muncul pada banyak kelas. Bila ICF memperhatikan $ICS_{\delta}F$ memperhatikan kemunculan *term* pada kumpulan dokumen yang menjadi anggota *class/category*. *Term* yang jarang muncul pada banyak dokumen anggota *class* adalah *term* yang sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah dokumen pada class yang mengandung *term*. Perhitungan $ICS_{\delta}F$ ini dapat diadopsi langsung dari ICF, akan tetapi ICF hanya menghiraukan *term* yang ada pada *class* tanpa menghiraukan jumlah *term* yang terdapat dalam dokumen yang menjadi anggota class. Perhitungan bobot $ICS_{\delta}F$ dari *term t* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah *term* pada dokumen anggota kelas yang mengandung *term*. Dalam *class density* (C_{δ}) pengambilan nilai direpresentasikan seperti berikut:

$$C_{\delta}(t_i) = \frac{n_{c_k}(t_i)}{N_{c_k}} \tag{3.5}$$

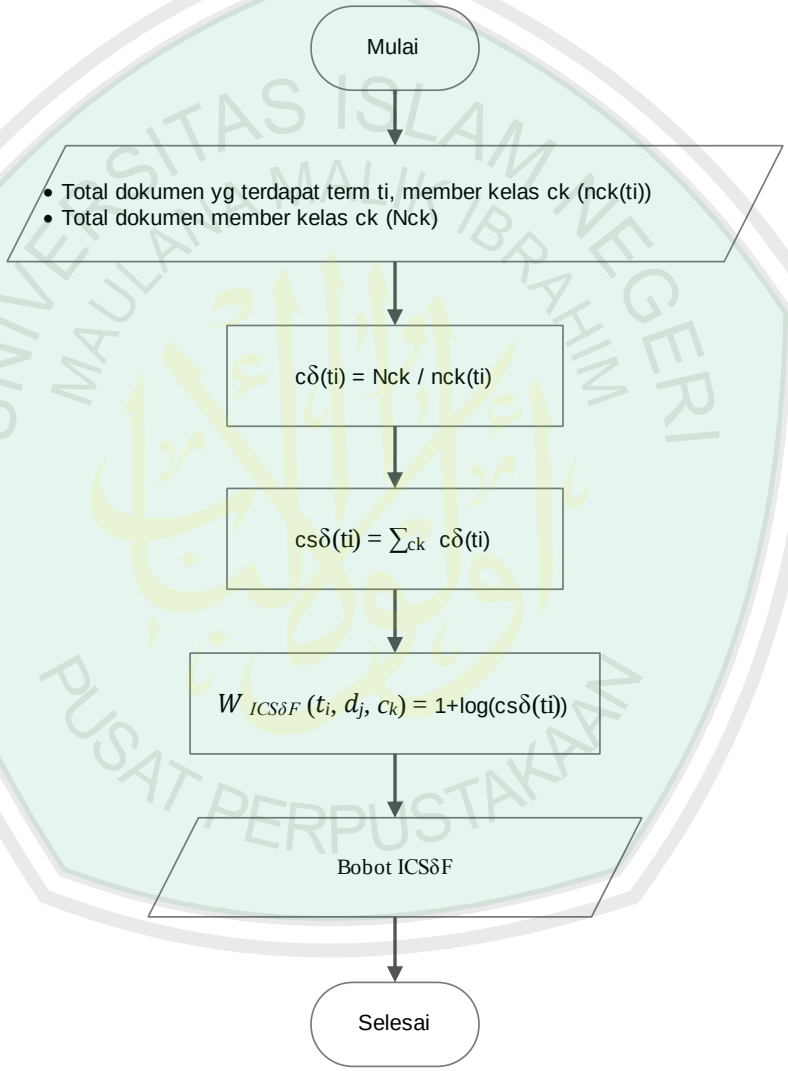
Dimana n_{c_k} (t_i) mengindikasikan jumlah keseluruhan dokumen yang di dalamnya terdapat *term* t_i dan merupakan anggota kelas c_k, dan N_{c_k} mengindikasikan jumlah keseluruhan dokumen yang menjadi anggota kelas c_k. Sedangkan nilai *space class density* merupakan nilai dari hasil penjumlahan kelas yang didapat dari *class density* (C_{δ}). δ

$$CS_{\delta}(t_i) = \sum_{c_k} C_{\delta}(t_i) \tag{3.6}$$

Perhitungan bobot $ICS_{\delta}F$ dari *term* t_i adalah sebagai berikut:

$$W_{ICS\delta F}(t_i, d_j, c_k) = 1 + log \frac{C}{cs\delta(ti)} \tag{3.7}$$

Dimana $\frac{CS_\delta(t_i)}{C}$ merujuk sebagai *class space density frequency* ($CS_\delta F$) dan $\frac{C}{CS_\delta(t_i)}$ merupakan inverse class space density frequency ($ICS_\delta F$) dari *term* t_i . Pembobotan *inverse class space density frequency* ($ICS_\delta F$) disajikan dalam *flowchart* seperti yang terdapat pada gambar 3.6.



Gambar 3.6 Pembobotan $ICS_\delta F$

Gabungan dari masing-masing pembobotan tersebut memberikan variasi pembobotan kata. Dalam penelitian ini menggunakan pembobotan TF.IDF.ICS$_\delta$F yang merupakan perkalian dari bobot TF, IDF dan ICS$_\delta$F dari suatu *term* terhadap

dokumen dan kelompok kepadatan kelas tertentu, sebagaimana yang didefinisikan pada persamaan berikut:

$$W_{TF.IDF.ICS\delta F}(t_i, d_j, c_k) = W_{TF}(t_i, d_j) \text{ x } W_{ICS\delta F}(t_i, d_j, c_k)$$

atau,

$$W_{TF.IDF.ICS\delta F}(t_i, d_j, c_k) = tf(ti, dj)\left(1 + log\frac{D}{d(ti)}\right) x \left(1 + log\frac{C}{cs\delta(ti)}\right) \quad (3.8)$$

3.2.5 *Vector Space Model (VSM)*

Pada *Vector space model* merepresentasikan dokumen atau *query* sebagai vektor dalam sebuah ruang *term* (Polettini, 2004; Cios, 2007). Ruang ini memiliki dimensi sebanyak jumlah *term* atau dengan kata lain untuk dokumen yang memiliki N kata maka dibutuhkan N dimensi. Setiap vektor direpresentasikan sesuai bobot dari *term-term* yang ada. *Term-term* yang ada menjadi sumbu-sumbu koordinat sebanyak jumlah *term*, sedangkan vektornya adalah sebuah titik yang posisinya berdasarkan nilai dari sumbu-sumbu tersebut. Misal untuk dokumen dengan dua dimensi, maka apabila direpresentasikan dalam bidang kartesian akan menjadi sebuah titik yang terdiri dari *x* dan *y*. Nilai *x* dan *y* ini tergantung pada bobot *term x* dan *y*. dalam implementasinya, *Vector Space Model* ini direpresentasikan sebagai matriks dua dimensi dengan kolom sebagai *term* dan dokumen sebagai baris, serta bobot *term* sebagai isi matriks.

Pembuatan *Vector Space Model* untuk dokemen adalah dengan cara membuat sebuah matriks dua dimensi dengan kolom sebagai *term* dan dokumen sebagai baris. Isi dari matriks tersebut merupakan nilai bobot *term* (*term weight*) dari masing-masing *term* terhadap masing-masing dokumen. *Term* adalah *Vector Space Model* yang diambil dari seluruh *term* unik pada keseluruhan dokumen.

3.2.6 *Cosine Similiarity*

Dari bobot *term* yang termuat dalam *vector space model* dilakukan perhitungan kemiripan menggunakan *cosine similiarity* sesuai dengan persamaan 3.5. Hasil dari perhitungan ini memberikan nilai kemiripan antara rentang 0 sampai 1. Nilai yang mendekati 1 menunjukkan tingkat kemiripan yang tinggi. Sehingga dari hasil perhitungan ini jika diurutkan akan menghasilkan dokumen yang berurutan (terangking).

$$\cos(q, dj) = \frac{\sum_{t_i}[w(t_i,q)].[w(t_i,d_j)]}{\sqrt{\sum |w(q)|^2}.\sqrt{\sum |w(d_j)|^2}} \quad (3.5)$$

Pada persamaan tersebut *cos (q, dj)* adalah nilai *cosinus* antara *query* dan dokumen *j*, $w(t_i,q)$ merupakan bobot TF.IDF.ICS$_\delta$F dari *term* t_i pada *query*, dan $w(t_i,d)$ merupakan bobot TF.IDF.ICS$_\delta$F untuk setiap *term* t_i pada dokumen j berdasarkan sebaran *term* pada kelas. Sedangkan $|w(q)|$ dan $|w(d_j)|$ masing-masing merupakan panjang dari vektor q dan panjang dari vektor dokumen *j*. sebagai contoh panjang vektor dokumen yaitu:

$||d_j||^2$=(TF.IDF.ICS$_\delta$F$_1^2$+TF.IDF.ICS$_\delta$F$_2^2$+TF.IDF.ICS$_\delta$F$_3^2$+…+ TF.IDF.ICS$_\delta$F$_i^2$)$^{1/2}$, dimana TF.IDF.ICS$_\delta$F adalah bobot kata ke-*ti* pada vektor dokumen d_j.

3.2.7 Output

Output dari proses pembobotan TF.IDF.ICS$_\delta$F adalah perangkingan ayat-ayat yang memiliki tingkat kemiripan paling tinggi. Tampilan output berupa surat, ayat, kelas dan *similiarity* (Tabel 3.4).

Tabel 3.4 Output program

No.	Surat	Ayat	Kelas	*Similiarity*

1	An-Nisa’	26	Iman	0.1666964599
2	Al-Baqarah	160	Iman	0.0903193511
3	An-Nisa’	17	Bangsa Terdahulu	0.0315471212
4	An-Naml	177	Jihad	0. 645047112
5	Al-Israa’	153	Al-Qur’an	0.0265294833

3.2.8 *Contoh Perhitungan Manual*

Contoh Kasus

Sebagai contoh, terdapat enam dokumen dengan kode mulai dari D1 hingga D6. Kenam dokumen tersebut adalah dokumen yang relevan. Dokumen tersebut tersebar pada kelas yang berbeda. Dokumen tersebut tersebar dalam 5 kelas dengan kode C1 hingga C6. Isi ke enam dokumen tersebut memiliki keterangan kelas dapat dilihat pada tabel 3.5.

Tabel 3.5 Contoh Dokumen untuk Perhitungan Manual

Dokumen	Kelas	Isi Dokumen
D1	Iman	إِنَّ اللَّهَ رَبِّي وَرَبُّكُمْ فَاعْبُدُوهُ هَٰذَا صِرَاطٌ مُسْتَقِيمٌ
D2	Iman	إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ
D3	Iman	وَاعْبُدْ رَبَّكَ حَتَّىٰ يَأْتِيَكَ الْيَقِينُ
D4	Ibadah	اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ
D5	Ibadah	الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ
D6	Bangsa Terdahulu	يَأْتُوكَ بِكُلِّ سَاحِرٍ عَلِيمٍ

Dilakukan pencarian terhadap keenam dokumen tersebut dengan keterangan :

- Query : وَاعْبُدْ رَبَّكَ حَتَّىٰ يَأْتِيَكَ الْيَقِينُ
- Kelas : Iman

Tahapan Perangkingan Dokumen

Tiap dokumen tersebut melaluai tahap *preprocessing*. Dari tahap preprocessing diketahui terdapat 14 *term* pada keenam dokumen tersebut sebagaimana berikut ini:

رب – نستعين – علم – سحر – بكل – أيك – يقن – أتي – عبد

حمد – قوم – صرط

Selanjutnya dilakukan perhitungan bobot TF, IDF, ICF dan $ICS_{\delta}F$ untuk setiap *term* dengan persamaan. Hasil perhitungan bobot TF, IDF, dan ICF tiap *term* tersebut dapat dilihat pada tabel 3.6, sedangkan perhitungan $ICS_{\delta}F$ tiap *term* dapat dilihat pada tabel 3.7. Pembobotan TF.IDF yang diperoleh dengan mengalikan bobot TF dengan bobot IDF di tunjukkan pada tabel 3.8. Pembobotan TF.IDF.ICF yang diperoleh dari perkalian bobot TF.IDF dengan bobot ICF ditunjukkan pada tabel 3.9. Sedangkan Tabel 3.10 menunujukkan pembobotan $TF.IDF.ICS_{\delta}F$ yang diperoleh dari perkalian bobot TF.IDF dengan $ICS_{\delta}F$.

Tabel 3.6 Perhitungan bobot TF, IDF, dan ICF

No	Term	TF							IDF	C				ICF
		D1	D2	D3	D4	D5	D6	$d(t_i)$	$1+\log(D/d(t_i))$	C1	C2	C3	$C(t_i)$	$1+\log(C/c(t_i))$
1	أتي	-	-	1	-	-	1	2	1.4771	1	-	1	2	1.301
2	أيك	-	2	-	-	-	-	2	1.4771	1	-	-	1	1.602
3	بكل	-	-	-	-	-	1	1	1.7781	-	-	1	1	1.602
4	حمد	-	-	-	-	1	-	1	1.7781	-	1	-	1	1.602
5	رب	2	-	1	-	1	-	4	1.1760	1	1	-	2	1.301
6	سحر	-	-	-	-	-	1	1	1.7781	-	-	1	1	1.602
7	صرط	1	-	-	1	-	-	2	1.4771	-	1	-	1	1.602
8	عبد	1	1	1	-	-	-	3	1.3010	1	-	-	1	1.602
9	علم	-	-	-	-	-	1	1	1.7781	-	-	1	1	1.602
10	نستعين	1	1	-	-	-	-	2	1.4771	1	-	-	1	1.602
11	يقن	-	-	1	-	-	-	1	1.7781	1	-	-	1	1.602
12	هدد	-	-	-	1	-	-	1	1.7781	-	1	-	1	1.602
13	قوم	-	-	-	1	-	-	1	1.7781	-	1	-	1	1.602

14	عالم	-	-	-	-	1	-	1	1.7781	-	-	1	1	1.602

Tabel 3.7 Perhitungan bobot $ICS_{\delta}F$

No	Term	Class	Nck_ti	nck	Class Density	Class Space D	c/csd	$ICS_{\delta}F$
1	أتي	C1 C3	1 1	13 5	0.0769 0.2	0.2769	10.834	2.034789
2	أيك	C1	2	13	0.1538	0.1538	19.505	2.290146
3	بكل	C3	1	5	0.2	0.2	15	2.176091
4	حمد	C2	1	4	0.25	0.25	12	2.079181
5	رب	C1	3	13	0.2307	0.2307	13.0039	2.114074
6	سحر	C3	1	5	0.5	0.5	6	1.778151
7	صرط صرط	C1 C2	1 1	13 4	0.0769 0.25	0.3269	9.1771	1.962705
8	عبد	C1	3	13	0.2307	0.2307	13.0039	2.114074
9	علم	C3	1	5	0.2	0.2	15	2.176091
10	نستعين	C1	2	13	0.1538	0.1538	19.5058	2.290164
11	يقن	C1	1	13	0.0769	0.0769	39.0117	2.591195
12	هدد	C2	1	4	0.25	0.25	12	2.079181
13	قوم	C2	1	4	0.25	0.25	12	2.079181
14	عالم	C3	1	5	0.2	0.2	15	2.176091

Tabel 3.8 Perhitungan bobot TF.IDF

No	Term	TF						IDF	TF.IDF					
		D1	D2	D3	D4	D5	D6		D1	D2	D3	D4	D5	D6
1	**أتي**	-	-	1	-	-	1	1.477	0.000	0.000	1.477	0.000	0.000	1.477
2	**أيك**	-	2	-	-	-	-	1.477	0.000	2.954	0.000	0.000	0.000	0.000
3	**بكل**	-	-	-	-	-	1	1.778	0.000	0.000	0.000	0.000	0.000	1.778
4	**حمد**	-	-	-	-	1	-	1.778	0.000	0.000	0.000	0.000	1.778	0.000
5	**رب**	2	-	1	-	1	-	1.176	2.352	0.000	1.176	0.000	1.176	0.000
6	**سحر**	-	-	-	-	-	1	1.778	0.000	0.000	0.000	0.000	0.000	1.778
7	**صرط**	1	-	-	1	-	-	1.477	1.477	0.000	0.000	1.477	0.000	0.000
8	**عبد**	1	1	1	-	-	-	1.301	1.301	1.301	1.301	0.000	0.000	0.000
9	**علم**	-	-	-	-	-	1	1.778	0.000	0.000	0.000	0.000	0.000	1.778
10	**نستعين**	1	1	-	-	-	-	1.477	1.477	1.477	0.000	0.000	0.000	0.000
11	**يقن**	-	-	1	-	-	-	1.778	0.000	0.000	1.778	0.000	0.000	0.000
12	**هدد**	-	-	-	1	-	-	1.778	0.000	0.000	0.000	1.778	0.000	0.000
13	**قوم**	-	-	-	1	-	-	1.778	0.000	0.000	0.000	1.778	0.000	0.000
14	**عالم**	-	-	-	-	1	-	1.778	0.000	0.000	0.000	0.000	1.778	0.000

Tabel 3.9 Perhitungan bobot TF.IDF.ICF

No	Term	TF	ICF	TF.IDF.ICF

		D1	D2	D3	D4	D5	D6	IDF		D1	D2	D3	D4	D5	D6
1	أتي	-	-	1	-	-	1	1.4771	1.301	0.000	0.000	1.921	0.000	0.000	1.921
2	أيك	-	2	-	-	-	-	1.4771	1.602	0.000	4.732	0.000	0.000	0.000	0.000
3	بكل	-	-	-	-	-	1	1.7781	1.602	0.000	0.000	0.000	0.000	0.000	2.848
4	حمد	-	-	-	-	1	-	1.7781	1.602	0.000	0.000	0.000	0.000	2.848	0.000
5	رب	2	-	1	-	1	-	1.1760	1.301	3.059	0.000	1.529	0.000	1.529	0.000
6	سحر	-	-	-	-	-	1	1.7781	1.602	0.000	0.000	0.000	0.000	0.000	2.848
7	صرط	1	-	-	1	-	-	1.4771	1.602	2.366	0.000	0.000	2.366	0.000	0.000
8	عبد	1	1	1	-	-	-	1.3010	1.602	2.084	2.084	2.084	0.000	0.000	0.000
9	علم	-	-	-	-	-	1	1.7781	1.602	0.000	0.000	0.000	0.000	0.000	2.848
10	نستعين	1	1	-	-	-	-	1.4771	1.602	2.366	2.366	0.000	0.000	0.000	0.000
11	يقن	-	-	1	-	-	-	1.7781	1.602	0.000	0.000	2.848	0.000	0.000	0.000
12	هدد	-	-	-	1	-	-	1.7781	1.602	0.000	0.000	0.000	2.848	0.000	0.000
13	قوم	-	-	-	1	-	-	1.7781	1.602	0.000	0.000	0.000	2.848	0.000	0.000
14	عالم	-	-	-	-	1	-	1.7781	1.602	0.000	0.000	0.000	0.000	2.848	0.000

Tabel 3.10 Perhitungan bobot TF.IDF. $ICS_{\delta}F$

No	Term	TF						IDF	$ICS_{\delta}F$	TF.IDF. $ICS_{\delta}F$					
		D1	D2	D3	D4	D5	D6			D1	D2	D3	D4	D5	D6
1	أتي	-	-	1	-	-	1	1.477	2.034	0.000	0.000	3.004	0.000	0.000	3.004
2	أيك	-	2	-	-	-	-	1.477	2.290	0.000	6.764	0.000	0.000	0.000	0.000
3	بكل	-	-	-	-	-	1	1.778	2.176	0.000	0.000	0.000	0.000	0.000	3.868
4	حمد	-	-	-	-	1	-	1.778	2.079	0.000	0.000	0.000	0.000	3.696	0.000
5	رب	2	-	1	-	1	-	1.176	2.114	4.972	0.000	2.486	0.000	2.486	0.000
6	سحر	-	-	-	-	-	1	1.778	1.778	0.000	0.000	0.000	0.000	0.000	3.161
7	صرط	1	-	-	1	-	-	1.477	1.962	2.897	0.000	0.000	2.897	0.000	0.000
8	عبد	1	1	1	-	-	-	1.301	2.114	2.750	2.750	2.750	0.000	0.000	0.000
9	علم	-	-	-	-	-	1	1.778	2.176	0.000	0.000	0.000	0.000	0.000	3.868
10	نستعين	1	1	-	-	-	-	1.477	2.290	3.382	3.382	0.000	0.000	0.000	0.000
11	يقن	-	-	1	-	-	-	1.778	2.591	0.000	0.000	4.606	0.000	0.000	0.000
12	هدد	-	-	-	1	-	-	1.778	2.079	0.000	0.000	0.000	3.696	0.000	0.000
13	قوم	-	-	-	1	-	-	1.778	2.079	0.000	0.000	0.000	3.696	0.000	0.000

14	عالم	-	-	-	-	1	-	1.778	2.176	0.000	0.000	0.000	0.000	3.868	0.000

Setelah memperoleh bobot TF, IDF, ICF dan ICS$_\delta$F dari setiap masing-masing *term*, maka pembobotan TF.IDF dilakukan terhadap setiap *term* sebagaimana yang ditunjukkan pada tabel 3.7. Pembobotan TF.IDF.ICF yang ditunjukkan pada tabel 3.9. Begitu juga TF.IDF.ICS$_\delta$F yang merupakan metode yang diajukan pada penelitian ini ditunjukkan pada tabel 3.10.

Sebelum melakukan perhitungan kemiripan antara *query* dengan dokumen, maka terhadap *query* yang dimasukkan oleh pengguna juga dilakukan tahap *preprocessing* dan pembobotan. Jumlah masing-masing *term query* dikalikan dengan bobot IDF dari *term* dokumen sehingga diperoleh TF.IDF (q_i, d_j). Masing-masing *term query* juga dikalikan dengan bobot IDF dan ICF dari *term* dokumen sehingga diperoleh TF.IDF.ICF(q_i, d_j, c_k). Begitu pula jumlah masing-masing *term query* dikalikan dengan bobot TF.IDF dan ICSdF dari *term* dokumen sehingga diperoleh TF.IDF.ICS$_\delta$F (q_i, d_j, c_k). Hasil pembobotan *query* ditunjukkan pada tabel 3.11.

Tabel 3.11 Pembobotan *Query*

No.	Term	TF	IDF	ICF	ICS$_\delta$F	TF.IDF	TF.IDF.ICF	TF.IDF.ICS$_\delta$F
1	عبد	1	1.301	1.602	2.114	1.301	2.084	2.750
2	رب	1	1.176	1.301	2.114	1.176	1.529	2.486
3	أتي	1	1.477	1.301	2.034	1.477	1.921	3.004
4	يقن	1	1.778	1.602	2.591	1.778	2.848	4.606

Tahap setelah pembobotan adalah *perhitungan cosine similarity* berdasarkan *query* pengguna dengan persamaan 3.5. Langkah untuk perhitungan *cosine*

similarity adalah dengan menghitung perkalian antara vektor *query* dan vektor setiap dokumen kemudian menghitung panjang vektor. Nilai perkalian antara vektor *query* dan dokumen dibagi dengan panjang vektor sehingga diperoleh nilai kosinus antara vektor *query* dan dokumen.contoh perhitungan *cosine similarity* untuk pembobotan TF.IDF.ICS$_\delta$F ditunjukkan pada tabel 3.12.

Tabel 3.12 Perhitungan *Cosine Similarity*

No.	Term	Q	TF.IDF.ICS$_\delta$F					
			D1	D2	D3	D4	D5	D6
1	أتي	3.004	0.000	0.000	3.004	0.000	0.000	3.004
2	أيك	0.000	0.000	6.764	0.000	0.000	0.000	0.000
3	بكل	0.000	0.000	0.000	0.000	0.000	0.000	3.868
4	حمد	0.000	0.000	0.000	0.000	0.000	3.696	0.000
5	رب	2.486	4.972	0.000	2.486	0.000	2.486	0.000
6	سحر	0.000	0.000	0.000	0.000	0.000	0.000	3.161
7	صرط	0.000	2.897	0.000	0.000	2.897	0.000	0.000
8	عبد	2.750	2.750	2.750	2.750	0.000	0.000	0.000
9	علم	0.000	0.000	0.000	0.000	0.000	0.000	3.868
10	نستعين	0.000	3.382	3.382	0.000	0.000	0.000	0.000
11	يقن	4.606	0.000	0.000	4.606	0.000	0.000	0.000
12	هدد	0.000	0.000	0.000	0.000	3.696	0.000	0.000
13	قوم	0.000	0.000	0.000	0.000	3.696	0.000	0.000
14	عالم	0.000	0.000	0.000	0.000	0.000	3.868	0.000
Jumlah kuadrat bobot		43.981	52.113	64.752	43.981	35.713	34.802	48.938
Panjang vektor		6.631	7.218	8.046	6.631	5.976	5.899	6.995

Jumlah perkalian bobot *query* dengan bobot dokumen	19.922	7.562	43.981	0.000	6.180	9.024
Cosine similarity	0.416	0.141	1.000	0	0.157	0.194

Panjang vektor merupakan nilai akar dari jumlah kuadrat bobot di masing-masing dokumen ataupun query. Sebagai contoh panjang vektor dokumen D1= $(TF.IDF.ICF_{\delta}{F_{t_1}}^2 + TF.IDF.ICF_{\delta}{F_{t_2}}^2 + TF.IDF.ICF_{\delta}{F_{t_3}}^2 + \cdots + TF.IDF.ICF_{\delta}{F_{t_i}}^2)^{1/2} = (4.972^2 + 2.897^2 + 2.750^2 + 3.382^2 = 52.113^{1/2} =$ 7.218. Begitu pula dengan perhitungan panjang vektor dokumen D2 hingga D6 dan panjang vektor *query*.

Setelah itu dilakukan perhitungan jumlah perkalian bobot *query* dengan setiap dokumen. Sebagai contoh jumlah perkalian bobot *query* dengan bobot dokumen D1 adalah (2.486 x 4972)+(2.750 x 2.750) = 19.922. begitu pula dilakukan perhitungan jumlah perkalian bobot query dengan bobot D2 hingga D6.

Nilai cosine similarity diperoleh dengan membagi jumlah perkalian bobot dengan perkalian panjang vektor query dan panjang vektor dokumen. Sebagaimana perhitungan berikut ini :

- $\cos(q, d_1) = \frac{19.922}{6.631\ x\ 7.218} = 0.416$
- $\cos(q, d_2) = \frac{7.562}{6.631\ x\ 8.046} = 0.141$
- $\cos(q, d_3) = \frac{43.981}{6.631\ x\ 6.631} = 1.000$
- $\cos(q, d_4) = \frac{0.000}{6.631\ x\ 5.976} = 0.000$
- $\cos(q, d_5) = \frac{6.180}{6.631\ x\ 5.899} = 0.157$
- $\cos(q, d_6) = \frac{9.024}{6.631\ x\ 6.995} = 0.194$

Tabel 3.13 menunjukkan hasil perhitungan *cosine similarity* yang telah diurutkan mulai dari nilai tertinggi hingga terendah dengan variasi pembobotan TF.IDF, TF.IDF.ICF dan TF.IDF.ICS$_\delta$F. Dari tabel tersebut diketahui ranking dokumen berdasarkan similaritas dokumen tersebut terhadap *query*. Hasil tersebut menunjukkan bahwa dengan metode TF.IDF.ICS$_\delta$F dokumen D3 dan D1 berada pada urutan ke-1 dan ke-2 dan memiliki nilai similaritas tertinggi dibandingkan dengan metode yang lain.

Tabel 3.13 Perhitungan *Query*

Ranking	TF.IDF		TF.IDF.ICF		TF.IDF.ICS$_\delta$F	
	Dokumen	Similaritas	Dokumen	Similaritas	Dokumen	similaritas
1.	D3	0.524	D3	0.551	D3	1.000
2.	D1	0.328	D1	0.297	D1	0.416
3.	D6	0.220	D2	0.177	D6	0.194
4.	D5	0.171	D6	0.149	D5	0.157
5.	D2	0.164	D5	0.116	D2	0.141

3.3 Implementasi Sistem

Pada tahapan ini dilakukan implementasi desain model sistem ke dalam kode program sehingga dapat dimengerti oleh komputer. Sistem yang di bangun adalah aplikasi berbasis desktop dengan menggunakan bahasa pemrograman Java dan *database* MySql. Setelah melalui proses selanjutnya data-data hasil ekstraksi diakses dari *database*.

Terdapat dua komponen utama dalam fase ini yaitu pengembangan sistem perangkingan dokumen bahasa Arab sesuai dengan metode yang diusulkan dan pengembangan *interface* sistem sebagai sara interaksi sistem dengan pengguna.

Lingkungan pengembangan penelitian yang digunakan dalam penelitian ini sebagai berikut:

1. Spesifikasi perangkat lunak yang digunakan:
   a. Sistem operasi : Windows 8.1 64-bit
   b. Netbeans IDE 7.2 dengan bahasa pemrograman Java
   c. Database sever: Mysql 5.5.24.
2. Spesifikasi perangkat keras yang digunakan:
   a. *Processor*: AMD A6-3420M APU with Radeon(tm) HD Graphic (4 CPUs), 1.5GHz
   b. Memory (RAM): 4 GB (3.47 *usable*)
   c. Database sever: Mysql 5.5.24.

3.4 Metode Pengujian

Uji coba dilakukan dengan melakukan pencarian dangan mengevaluasi hasil perangkingan dokumen. Pengujian dilakukan terhadap sejumlah *input user*, mulai input ke-1 hingga input ke-n. Dari masing-masing *input* tersebut dapat diperoleh nilai akurasi sebagai nilai evaluasi kemampuan sistem yang ditunjukkan pada tabel 3.5.

Tabel 3.5 Skenario Uji Coba tingkat akurasi

No	Input	Nilai similarity Terbesar			Output (surat : ayat)			Relevansi		
		A	B	C	A	B	C	A	B	C
1										
2										
3										
4										
5										
6										

Metode TF.IDF.ICS$_\delta$F akan dibandingkan dengan metode *term weighting* yang lainnya, yaitu TF.IDF dan TF.IDF.ICF. Pengujian ini ditunjukkan dilakukan dengan cara menghitung jumlah data yang dirtrieve pada urutan pertama yang relevan dengan input user kemudian. Kemudian dicari nilai rata-rata nilai akurasi.

$$\textbf{\textit{Akurasi}} = \frac{\textbf{\textit{hasil relevan}}}{\textbf{\textit{total pengujian}}} \times 100\% \quad (3.6)$$

BAB IV

UJI COBA DAN PEMBAHASAN

Pada bagian ini dijelaskan mengenai implementasi dari setiap langkah yang dipaparkan dari bab 3. Kemudian dilanjutkan dengan menunjukkan hasil dari uji coba sesuai dengan skenario pengujian yaitu menghitung tingkat akurasinya pada metode yang diusulkan dengan dibandingkan dengan beberapa metode sebelumnya. Setelah itu dipaparkan evaluasi dan pembahasan hasil diperoleh pada bagian akhir bab ini.

4.1 Implementasi

Metode yang diusulkan diimplementasikan dengan menggunakan bahasa *Java* pada *platform Developmen Kit* (JDK) 1.7.0 dan IDE Netbeans 8.0. *Database server* yang digunakan adalah MySQL, dengan desain antarmuka swing Java, dan *library* lucene sebagai *framework preprocessing*. Aplikasi ini dibangun diatas *platform* Microsoft Windows 8, dengan spesikasi *processor* AMD A6-3420M APU dan *memory* 4 GB.

Implementasi algoritma dilakukan dengan membuat fungsi-fungsi dari tahapan yang telah dipaparkan pada Bab 3. Pada bagian ini ditampilkan hasil implementasi di setiap langkahnya beserta potongan-potongan script yang penting dalam bagiannya.

4.1.1 Pembuatan Indeks Dokumen

Pembuatan indeks dokumen dilakukan melalui beberapa tahap. Tahap pertama adalah pengambilan data indeks al-Qur'an yang akan digunakan sebagai

data kelas dan pengambilan isi dokumen dari database. Tahap selanjutnya adalah *tokenizing*, *fitering*, dan *stemming* dan penghapusan *stopword*. Hasil dari *preprocessing* tersebut adalah kumpulan *term* dari seluruh dokumen yang menjadi set fitur asli. Setelah itu dilakukan *term weighting* dengan TF, IDF, dan $ICS_{\delta}F$ pada masing-masing *term* tersebut.

4.1.2 Pengambilan Data dari *Database*

Data yang digunakan pada penelitian ini disimpan pada MySQL yang terdiri dari tabel kelas dan tabel dokumen. Proses koneksi ke *database* dilakukan melalui perantara kelas Koneksi. *Instance* dari kelas Koneksi, digunakan dalam *form* utama dan dirujuk oleh beberapa kelas lain yang memerlukan fitur manipulasi data ke *database*, baik berupa fungsi *select*, *insert*, atau *update*. Proses koneksi ke MySQL dilakukan melalui fungsi-fungsi *library* API JDBC. Pada penelitian ini, *library* JDBC-MySQL yang digunakan adalah *library* mysql-connector-java-5.1.0-bin.jar.

Kelas koneksi memiliki atribut conn sebagai objek yang bertipe *Connection*. Terdapat beberapa *method* utama yang digunakan untuk melakukan operasi pada *database*. *Method destroyConnection()* digunakan untuk memutus koneksi ke *database*. *Method getConnection()* digunakan untuk membangun koneksi ke *database* MySQL dengan perantara *driver* JDBC. Untuk melakukan operasi pada koneksi, dibutuhkan empat parameter yakni nama server (URL), nama *database*, *username*, dan *password*. Pada Gambar 4.1 ditampilkan kode untuk membangun dan manajemen koneksi.

```
[1] String ipServer;//localhost
[2] String database;//nama database
[3] String username;//username database
[4] String password;// password database
[5] this.destroyCoonection(); //method putus koneksi
[6] String dbDriver; //driver JDBC
[7] String dbURL; //Driver URL JDBC
[8] Class.forName(dbDriver);
[9] //untuk koneksi
[10] conn = DriverManager.getConnection(dbURL, username, password);
[11] return true;//proses pembangunan koneksi berhasil
[12] Return false;//proses pembangunan koneksi gagal
```

Gambar 4.1 Method untuk koneksi *database*

4.1.3 Proses *Preprocessing* Dokumen

Pada masing-masing dokumen yang akan dibuat indeksnya terlebih dahulu dilakukan *preprocessing* untuk memudahkan proses selanjutnya. Implementasi *preprocessing* ini terdiri dari beberapa tahapan, yaitu *tokenizing*, *filtering*, *normalization*, *stopword removal* dan *stemming*.

a. Tokenizing

Tokenizing dilakukan untuk memecah keseluruhan isi dokumen menjadi suku kata tunggal. Proses *tokenizing* ditunjukkan pada gambar 4.2, *tokenizing* dilakukan dengan cara menghapus karakter tanda baca dan angka, dilanjutkan dengan pemotongan kata sebagaimana yang ditunjukkan pada baris ke-12.

```
[1] public static List<String> parseKeywords(Analyzer analyzer, String
    field, String keywords, List<String>stopword) throws IOException{
[2]   List<String> result = new ArrayList<>();
[3]   TokenStream stream = analyzer.tokenStream(field,
          newStringReader(keywords));
[4]   stream.reset();
[5]   try {
[6]     while(stream.incrementToken()) {
[7]        String temp =
              stream.getAttribute(CharTermAttribute.class).toString();
[8]        temp=temp.replace("'", "");
[9]        temp=temp.replace("\"", "");
[10]       temp=temp.replace(" ", "");
[11]       temp=temp.replaceAll("[0-9]","");
[12]       temp=temp.trim();
[13]    }
[14]  }
[15]}
```

Gambar 4.2 Method untuk *tokenizing* dokumen

b. *Normalization* dan *Filtering*

Sedangkan pada pada tahapan Normalization dan *filtering* dilakukan, menormalkan teks kedalam bentuk dasar tulisan Arab (*hijaiyah*), penghilangan karakter-karakter yang bukan yang tidak termasuk huruf *hijaiyah*, kemudian dilakukan penghapusan *diacritic* (harokat) dan tanda baca. Pada gambar 4.3 ditunjukkan satu contoh proses *filtering* yaitu method untuk menghilangkan harokat. Jika karakter bukan termasuk harokat maka karakter akan ditambahkan pada *modifiedWord* sebagaiman ditunjukkan pada kode baris ke-6. Setelah dilakukan proses *filtering* data yang dihasilkan berupa potongan kata tanpa harokat dan tanpa tanda baca.

```
[1] private boolean removeDiacritics (String currentWord, StringBuffer
    modifiedWord){
[2]    boolean diacriticFound = false;
[3]    modifiedWord.setLength (0);
[4]    Vector diacritics = (Vector) staticFiles.elementAt(17);
[5]    for (int i = 0; i < currentWord.length(); i++)
[6]       if (!(diacritics.contains(currentWord.substring(i, i+1))))
[7]           modifiedWord.append ( currentWord.substring ( i, i+1 ) );
[8]    else{
[9]       diacriticFound = true;
[10]   }
[11] return diacriticFound;
[12] }
```

Gambar 4.3 Method untuk menghapus harokat

c. *Stopword Removal*

Proses stopword removal adalah menghilangkan *term* yang sering muncul dalam dokumen tetapi tidak mempunyai nilai yang berarti pada sebuah dokumen. Mekanisme penghapusan yaitu dengan cara apakah *term* termasuk stopword, jika *term* tidak termasuk dalam *stopword* maka *term* akan disimpan, sebagaimana ditunjukkan pada gambar 4.4. Setelah dilakukan penghapusan *stopword*, tinggal menyisakan kata-kata yang memiliki pengaruh dalam suatu dokumen.

```
[1] if(!stopword.contains(StringUtils.lowerCase(temp)) &&
       temp.compareTo("")!=0){
[2]           result.add(temp);
[3] }
```

Gambar 4.4 Kode untuk *stopword removal*

d. *Stemming*

Stemming adalah proses konversi *term* ke dalam bentuk kata dasar. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi *terms* tertentu didalamnya. Dalam tahapan ini setiap kata yang telah dihasilkan dari serangkaian proses sebelumya kemudian dirubah menjadi kata dasar. Salah satu contoh proses stemming yaitu penghapusan *prefix*. Pada gambar 4.5 ditunjukkan method untuk

pengecekkan *prefix*. Kode baris ke-9 sampai ke-17 untuk mengecek apakah kata tersebut merupakan akar kata yang terdiri dari 3 atau 4 huruf, jika karakter hanya terdiri dari 2 huruf, uji untuk mengetahui salah satu huruf telah terhapus. Jika akar kata masih belum ditemukan akan dilanjutkan untuk pengecekkan pola kata, yang di kerjakan method *checkPatterns()* sebagai-mana ditunjukkan pada baris kode ke-18. Jika akar kata belum ditemukan maka menuju ke pengecekkan *suffixes* yang dikerjakan oleh method *checkForSuffixes()* yang ditunjukkan pada baris kode ke -21. Jika akar kata telah ditemukan dan bukan termasuk *stopword* maka nilai akan dikembalikan ke *modifiedWord* yang merujuk pada nilai pengembalian kata sebagaimana pada baris kode ke-24.

```
[1] private String checkForPrefixes(String word){
[2]    String prefix = "";
[3]    String modifiedWord = word;
[4]    Vector prefixes = (Vector) staticFiles.elementAt(10);
[5]    for (int i = 0; i < prefixes.size(); i++){
[6]       prefix = (String)prefixes.elementAt(i);
[7]       if (prefix.regionMatches(0,modifiedWord,0,prefix.length())){
[8]           modifiedWord = modifiedWord.substring(prefix.length());
[9]           if (modifiedWord.length() == 2){
[10]              modifiedWord = isTwoLetters(modifiedWord);
[11]          {
[12]          else if (modifiedWord.length() == 3 && !rootFound){
[13]              modifiedWord = isThreeLetters(modifiedWord);
[14]          }
[15]          else if (modifiedWord.length() == 4){
[16]              isFourLetters(modifiedWord);
[17]          }
[18]          if (!rootFound && modifiedWord.length() > 2){
[19]              modifiedWord = checkPatterns(modifiedWord);
[20]          }
[21]          if (!rootFound && !stopwordFound && !fromSuffixes){
[22]              modifiedWord = checkForSuffixes ( modifiedWord );
[23]          }
[24]         if (rootFound && !stopwordFound){
[25]              return modifiedWord;
[26]          }
[27]       }
[28]    }
[29]    fromSuffixes = false;
[29]    return word;
[30] }
```

Gambar 4.5 Method untuk menghapus *prefix*

Hasil dari seluruh proses *preprosesing* disimpan database. Daftar term hasil tahap preprocessing yang telah disimpan pada database ditunjukkan pada tabel 4.1. Dari 6236 dokumen yang digunakan pada penelitian ini, diperoleh 45.907 *term* hasil *preprocessing* dengan *distict term* sejumlah 3.013.

Tabel 4.1 Daftar *term* hasil *preprocessing*

id_doc	id_class	surat	ayat	term
1	**1**	**1**	**1**	**بسم**
1	**1**	**1**	**1**	**رحم**
1	**1**	**1**	**1**	**رحم**
2	**7**	**1**	**2**	**حمد**
2	**7**	**1**	**2**	**ربب**
2	**7**	**1**	**2**	**علم**

4.1.4 Pembobotan TF.IDF.ICS$_\delta$F

Tahap selanjutnya setelah *preproccessing* adalah *term weighting*. Pembobotan ini dilakukan dengan cara menghitung TF (*Term Frequency*), IDF (*Inverse Document Frequency*), kemudian ICS$_\delta$F (*Inverse Class Space Density Frequency*) dari masing-masing *term* pada seluruh dokumen. Setelah diperoleh nilai TF, IDF dan ICS$_\delta$F dari masing-masing *term*, maka dilakukan pembobotan TF.IDF. ICS$_\delta$F.

Perhitungan TF dilakukan dengan menghitung jumlah frekuensi kemunculan *term* pada masing-masing dokumen. Implementasi dari perhitungan TF dan *term* yang sudah tersimpan dalam *database* dilakukan dengan kode SQL *count(term)* sebagai TF yang di-*group* berdasarkan *id_doc* dan *term* ditunjukkan pada gambar 4.6. Darisana maka dapat diketahui jumlah setiap *term* pada setiap dokumen ditunjukkan pada tabel 4.2.

```
CREATE TABLE weight_tf (SELECT id_doc, term, count(term) as tf
        FROM term GROUP BY id_doc, term);
```

Gambar 4.6 Query sql bobot TF

Tabel 4.2 Daftar Nilai TF

id_doc	term	tf
1	بسم	1
1	رحم	2
2	حمد	1
2	ربب	1
2	علم	1
2	ليل	1
3	رحم	2
4	ملك	1

Pembobotan IDF dilakukan dengan menghitung persebaran *term* pada seluruh dokumen. Nilai IDF suatu *term* berbanding terbalik dengan jumlah dokumen yang mengandung term tersebut. Langkah pertama implementasi pembobotan IDF dilakukan dengan menghitung jumlah dokumen yang memuat suatu term. Hal ini dilakukan dengan kode SQL *count(id_doc)* sebagai *document frequency* (df) yang di-*group* berdasarkan *term* ditunjukkan pada gambar 4.7. Langkah selanjutnya, ditunjukkan pada gambar 4.8 perhitungan IDF dilakukan melalui program java dengan fungsi *log* pada java yaitu *Math.log10();*. Sehingga *idf = Math.log10(num_doc/df);*, sehingga tabel idf terupdate dengan diketahui nilai IDF yang ditunjukkan pada tabel 4.3.

```
CREATE TABLE weight_idf (id_idf INT NULL AUTO_INCREMENT KEY)
        SELECT term, count(id_doc) as df FROM term GROUP BY
        term);
```

Gambar 4.7 Query sql frekuensi *term* t_i pada dokumen d_j

```
[1] num_doc = 293;// jumlah doc
[2] for (int i = 1; i <= 750; i++) {//jumlah term
[3] sql = "SELECT DF FROM weight_IDF WHERE id_idf=" + i;
[4] ResultSet rs = theKoneksi.executeSelect(sql);
[5] while (rs.next()) {
[6]      df = Double.parseDouble(rs.getString("df"));
[7]      d_df = num_doc / df;
[8]      idf = Math.log10(d_df);
[9]      sql = "UPDATE weight_IDF SET d_df = " + d_df + ", idf=" + idf
              +" WHERE id_idf=" + i;
[10]     theKoneksi.executeUpdate(sql);
[11]     System.out.println("id_idf= "+ i+" d_df=" + d_df + ", idf =
              "+idf);
[12]  }
[13] }
```

Gambar 4.8 Kode pembobotan IDF

Tabel 4.3 Daftar Nilai IDF

id_idf	term	df	d_df	idf
1	آل	1	293	2.4668676203541096
2	أبن	3	97.66666666666667	1.989746365634447
3	أبي	2	146.5	2.1658376246901283
4	أتم	2	146.5	2.1658376246901283
5	أتن	3	97.66666666666667	1.989746365634447
6	أتي	3	97.66666666666667	1.989746365634447
7	أثم	3	97.66666666666667	1.989746365634447

Pembobotan ICF dilakukan dengan menghitung persebaran *term* pada seluruh kelas. Nilai ICF sebuah *term* masing-masing berbanding terbalik dengan jumlah kelas yang mengandung *term* tersebut. Sama dengan perhitungan IDF, langkah pertama pembobotan ICF dengan menggunakan kode SQL. Untuk mendapatkan bobot ICF, terlebih dahulu dilakukan perhitungan sebaran *term* pada kumpulan kelas yang digunakan, hal ini dilakukan dengan kode *count(term)* yang di-*group* berdasarkan *id_class* dan *term* yang ditunjukkan gambar 4.9. Setelah proses *select* sebaran *term* tersebut maka dapat dilakukan perhitungan jumlah kelas

yang memuat *term* dengan kode *count(id_class)* dari sebaran *term* yang ditunjukkan gambar 4.10. Sehingga dapat diperoleh nilai *class frequency* (cf). Pada gambar 4.11 ditunjukkan perhitungan bobot ICF yang juga dilakukan dengan kode java, yaitu dengan pemanggilan fungsi *Math.log10();*. Sehingga *icf = Math.log10(num_class/cf);*. Sehingga didapatkan nilai ICF dari masing-masing term yang ditunjukkan pada tabel 4.4.

```
CREATE TABLE sebaran_term (SELECT id_class, term, count(term)
     as df FROM term GROUP BY id_class, term);
```

Gambar 4.9 Query sql frekuensi *term* t_i pada kelas c_k

```
CREATE TABLE weight_icf (select id_class, term, count(id_class)
    as cf from sebaran_term group by term)
```

Gambar 4.10 Query sql frekuensi kelas c_k yang memuat *term* t_i

```
[1] num_class=11;//jumlah kelas
[2] for (int i = 1; i <= 768; i++) {
[3]     sql = "SELECT CF FROM weight_ICF WHERE id_icf=" + i;
[4]     ResultSet rs = theKoneksi.executeSelect(sql);
[5]     while (rs.next()) {
[6]         cf = Double.parseDouble(rs.getString("cf"));
[7]         c_cf = num_class / cf;
[8]         icf = i + (Math.log10(c_cf));
[9]         sql = "UPDATE weight_ICF SET c_cf = " + c_cf + ",
                  icf = " + icf + " WHERE id_icf=" + i;
[10]        theKoneksi.executeUpdate(sql);
[11]        System.out.println("id_icf = " + i + "c_cf =" +
                  c_cf + ",icf = " + icf);
[12]    }
[13] }
```

Gambar 4.11 Kode pembobotan ICF

Tabel 4.4 Daftar Nilai ICF

id_icf	id_class	term	cf	c_cf	icf
1	1	آل	1	11	1.0413926851582251
2	1	أبن	1	11	1.0413926851582251
3	1	أبي	2	5.5	0.7403626894942439
4	1	أتم	1	11	1.0413926851582251
5	1	أتن	1	11	1.0413926851582251
6	1	أتي	1	11	1.0413926851582251
7	1	أثم	1	11	1.0413926851582251

Pembobotan $ICS_{\delta}F$ terlebih dahulu dilakukan perhitungan sebaran *term* pada kelas c_k, hal ini dilakukan dengan kode *count(term)* yang di-*group* berdasarkan *id_class* dan *term* yang ditunjukkan pada gambar 4.12. Setelah proses *select* sebaran *term* terhadap suatu kelas, maka dilakukan perhitungan jumlah dokumen yang memuat *term* yang menjadi anggota kelas c_k atau *class density*, dengan cara membagi jumlah dokumen yang memuat *term* t_i yang menjadi anggota kelas c_k ($n_{c_k}(t_i)$) dengan jumlah dokumen anggota kelas c_k (N_{c_k}) yang ditunjukkan pada gambar 4.13. Setelah didapatkan nilai *class density*, kemudian mencari nilai *class space density* dengan kode *sum(class_density)* dari tabel class_density yang di *group* berdasarkan *term* sebagaimana ditunjukkan pada gambar 4.14. Sehingga dapat diperoleh nilai *class space density* (csd) seperti yang ditunjukkan pada tabel 4.5.

```
CREATE TABLE class_density11 (select id_class, term,
      count(term) as nck_ti from term group by id_class, term)
```

Gambar 4.12 *Query* sql frekuensi *term* t_i pada kelas c_k

```
CREATE TABLE class_density (id_cd INT NULL AUTO_INCREMENT)
      (SELECT id_class, term, nck_ti, nck,(nck_ti/nck) as
      class_density FROM class_density11);
```

Gambar 4.13 *Query* sql *Class Density* (kepadatan kelas c_k)

```
CREATE TABLE class_space_density (id_csd INT NULL
      AUTO_INCREMENT (select term, sum(class_density ) as
      class_density from class_density group by term)
```

Gambar 4.14 *Query* sql *Class Space Density*

Perhitungan bobot $ICS_\delta F$ juga dilakukan dengan kode java, yaitu dengan pemanggilan fungsi *Math.log10();*. Sehingga *icf = Math.log10(num_class/cf);* yang ditunjukkan pada gambar 4.15. dari proses perhitungan tersebut dihasilkan bobot $ICS_\delta F$ yang ditunjukkan pada tabel 4.5.

```
[1] num_class=11;//jumlah kelas
[2] for (int i = 1; i <= 768; i++) {
[3]   sql = "SELECT csd FROM class_space_density WHERE id_csd="
        + i;
[4]   ResultSet rs = theKoneksi.executeSelect(sql);
[5]   while (rs.next()) {
[6]    csd = Double.parseDouble(rs.getString("csd"));
[7]    c_csd = num_class / csd;
[8]    icsdf = i + (Math.log10(c_csd));
[9]    sql = "UPDATE class_space_density SET c_csd = " + c_csd
           + ", icsdf = " + icsdf + " WHERE id_csd=" + i;
[10]   theKoneksi.executeUpdate(sql);
[11]   System.out.println("id_csd = " + i + " c_csd =" + c_csd
           + ", icsdf = " + icsdf);
[12]   }
[13] }
```

Gambar 4.15 Kode pembobotan $ICS_\delta F$

Tabel 4.5 Daftar Nilai ICS$_\delta$F

id_csd	term	csd	c_csd	icsd
1	آل	0.0004	27500	4.439332693830263
2	أبن	0.0012	9166.666666666668	3.9622114391106003
3	أبي	0.0106	1037.7358490566037	3.016086819893455
4	أتم	0.0008	13750	4.138302698166282
5	أتن	0.0012	9166.666666666668	3.9622114391106003
6	أتي	0.0012	9166.666666666668	3.9622114391106003
7	أثم	0.0012	9166.666666666668	3.9622114391106003

Setelah mendapatkan nilai TF, IDF, ICS, dan ICS$_\delta$F maka dilakukan perkalian bobot TF.IDF, TF.IDF.ICF, dan TF.IDF. ICS$_\delta$F menggunakan *query* sebagaimana ditunjukkan pada gambar 4.16. Tabel 4.6 menunjukkan daftar nilai seluruh bobot yang telah tersimpan dalam *database*.

```
CREATE TABLE weight (SELECT i.id_doc, i.term, (t.tf*d.idf) as
    tf_idf,(t.tf*d.idf*c.icf) as tf_idf_icf,
    (t.tf*d.idf*s.icsdf) as tf_idf_icsdf FROM term i,
    weight_idf d, weight_tf t, weight_icf c, weight_icsdf s
    WHERE i.term = d.term AND i.term = t.term AND i.term =
    c.term AND i.term = s.term GROUP BY i.id_doc, i.term)
```

Gambar 4.16 *Query* sql perhitungan keseluruhan bobot

Tabel 4.6 Daftar bobot TF.IDF, TF.IDF.ICF, dan TF.IDF.ICS$_\delta$F

id_idoc	term	tf_idf	tf_idf_icf	tf_idf_icsdf
255	آل	2.4668676203541096	2.568950904474	10.951246078389259
94	أبن	1.989746365634447	2.072107310491876	7.883795810845549
41	أبي	2.1658376246901283	1.6035053688234082	6.532354313857243
131	أتم	2.1658376246901283	2.2554874595927648	8.962891686045209
128	أتن	1.989746365634447	2.072107310491876	7.883795810845549
125	أتي	1.989746365634447	2.072107310491876	7.883795810845549
188	أثم	1.989746365634447	2.072107310491876	7.883795810845549

4.1.5 Ukuran Kemiripan dengan *Cosine Similarity*

Pencarian *Query* dilakukan dengan menghitung *cosine similarity* antara *query* dan masing-masing dokumen. Perhitungan *cosine similarity* ini didasarkan pada pembotan TF.IDF.ICS$_\delta$F. Pada implementasinya, perhitungan *cosine similarity* ini dilakukan dengan cara membandingkan kedekatan antara matriks *vector space model query* dan matriks *vector space model* masing-masing dokemen.

Sebelum dilakukan perhitungan ukuran kemiripan antara *query* dengan dokumen, terlebih dahaulu dilakukan *preprocessing* dan pembobotan terhadap *query* pengguna. Sebagaimana *preprocessing* dokumen, terhadap *preprocessing query* juga dilakukan dengan *library lucene*. Hasil *preprocessing query* disimpan dalam tabel sebagai penyimpanan sementara pada *database*. Contoh *query* pengguna adalah:

"تِلْكَ آيَاتُ اللَّهِ نَتْلُوهَا عَلَيْكَ بِالْحَقِّ وَإِنَّكَ لَمِنَ الْمُرْسَلِينَ"

Setelah kata yang di input melalui tahap *preprocessing*, *term* yang dihasilkan berupa kata berikut ini : تلا - حقق - علك - رسـل - ان - اي. Kemudian dilakukan perhitungan bobot *query* menggunakan kode seperti yang ditunjukkan pada gambar 4.17.

```
CREATE TABLE proses_bobot_input (SELECT i.input_term,
        count(i.input_term)as tf, d.idf, s.icsdf,
        (count(i.input_term)*d.idf*s.icsdf) as bobot FROM
        Q_term i, weight_idf d, weight_icsdf s WHERE
        i.input_term = d.term AND i.input_term = s.term GROUP
        BY i.input_term);
```

Gambar 4.17 *Query* sql perhitungan bobot input

Tabel 4.7 Daftar bobot input

input_term	tf	idf	icsdf	bobot
ان	1	0.864807629	1.7431948180	1.5075281775
اي	1	0.935388703	2.0099842209	1.8801165340
تلا	1	1.767897616	2.2532243140	3.9834698931
حقق	1	1.466867620	2.3372421683	3.4284248576
رسل	1	1.262747637	1.9788107009	2.4987385380
علك	1	2.165837624	3.0160868198	6.5323543138

Langkah untuk perhitungan *cosine similarity* adalah dengan menghitung perkalian antara vektor *query* vektor setiap dokumen kemudian menghitung panjang vektor. Panjang vektor merupakan nilai akar dari jumlah kuadrat bobot pada masing-masing dokumen atau *query*. Nilai *cosine similarity* diperoleh dengan membagi jumlah perkalian bobot dengan perkalian panjang vektor *query* dan panjang vektor dokumen.

Implementasi perhitungan perkalian antara vektor *query* dengan vektor setiap dokumen dilakukan dengan kode *sum(i.bobot*b.tf_idf_icsδf)* dengan kondisi term dokumen sama dengan *term query* dan di *groub* berdasarkan *id_doc*. Sedangkan panjang vektor dokumen dihitung dengan kode *sqrt(sum(tf_idf_ icsδf* tf_idf_ icsδf))* yang di groub berdasarkan *id_doc*. Begitu pula panjang vektor *query* dihitung dengan kode *sqrt(sum(bobot*bobot)).*

```
[1] sql = "INSERT INTO proses_dot_product (SELECT b.id_doc, "
                + "sum(i.bobot*b." + tipe_weighting + ") "
                + "FROM proses_bobot_input i, weight b "
                + "WHERE b.term = i.input_term GROUP BY
                  b.id_doc)";
[2] theKoneksi.executeUpdate(sql);
[3] sql = "DROP TABLE vector length";
[4] theKoneksi.executeUpdate(sql);
[5] sql = "CREATE TABLE vector_length (SELECT id_doc,
      sqrt(sum("+ tipe_weighting + "*" + tipe_weighting + "))
      as " + " vector_length FROM weight GROUP BY id_doc)";
[6] theKoneksi.executeUpdate(sql);
[7] sql = "SELECT sqrt(sum(bobot*bobot)) as vectorQ FROM "
              proses_bobot_input";
[8] ResultSet rs = theKoneksi.executeSelect(sql);
[9]   while (rs.next()) {
[10]    vectorQ = Double.parseDouble(rs.getString("vectorQ"));
[11]  }
[12] sql = "INSERT INTO result_similarity (SELECT p.id_doc, "
           + " c.id_class, (p.dot_product/(" + vectorQ
           + "*v.vector_length)) FROM proses_dot_product p, "
           + "vector_length v, document d, class c "
           + "WHERE p.id_doc = v.id_doc AND p.id_doc =
           d.id_doc
           AND d.id_class = c.id_class)";
[13] theKoneksi.executeUpdate(sql);
```

Gambar 4.18 Kode perhitungan *cosine similarity*

Pada gambar 4.18 ditunjukkan kode perhitungan *cosine similarity*. Baris ke-1 menunujukkan perkalian antara vektor *query* dengan vektor dokumen. Hasil perkalian antara antara vektor *query* dengan vektor dokumen ditunjukkan pada tabel 3.13. baris ke-5 menujukkan perhitungan panjang vektor dokumen, sedangkan baris ke-7 menunjukkan perhitungan panjang vektor *query*. Tabel 4.9 menunjukkan hasil perhitungan panjang vektor *query* sedangkan 4.10 menunjukkan hasil perhitungan panjang vektor dokumen. Perhitungan jarak kosinus antara vektor

dokumen dan vektor *query* ditunjukkan pada baris ke-12, yang dihasilkan dari pembagian antara hasil perkalian vektor dengan hasil panjang vektor dokumen dikali hasil panjang vektor *query*.

4.8 Hasil perkalian vektor *query* dengan vektor dokumen

id_doc	dot_product
5	7.069676363375149
29	2.272641206038213
78	11.754097004432628
94	12.487388563085743
109	15.86803238927199
259	42.67165288176933

Tabel 4.9 Hasil perhitungan panjang vektor *query*

vectorQ
9.07441215422479

Tabel 4.10 Hasil perhitungan panjang vektor dokumen

id_doc	vector_length
1	9.762447653416993
2	7.006161896799026
4	5.271634926712369
5	12.075128680177244
6	10.827632468673231
7	4.76966024182396

Perhitungan *cosine similarity* ini akan menghasilkan nilai kemiripan antara matriks *vector space model query* dan matriks *vector space model* masing-masing dokumen. Semakin besar nilai *cosine similarity* antara *query* dan sebuah dokumen, maka semakin besar pula tingkat kemiripannya. Berdasarkan nilai *cosine similarity* tersebut akan didapatkan hasil perangkingan dokumen sesuai dengan tingkat kemiripan dokumen terhadap *query* diurutkan berdasakan dokumen memiliki nilai

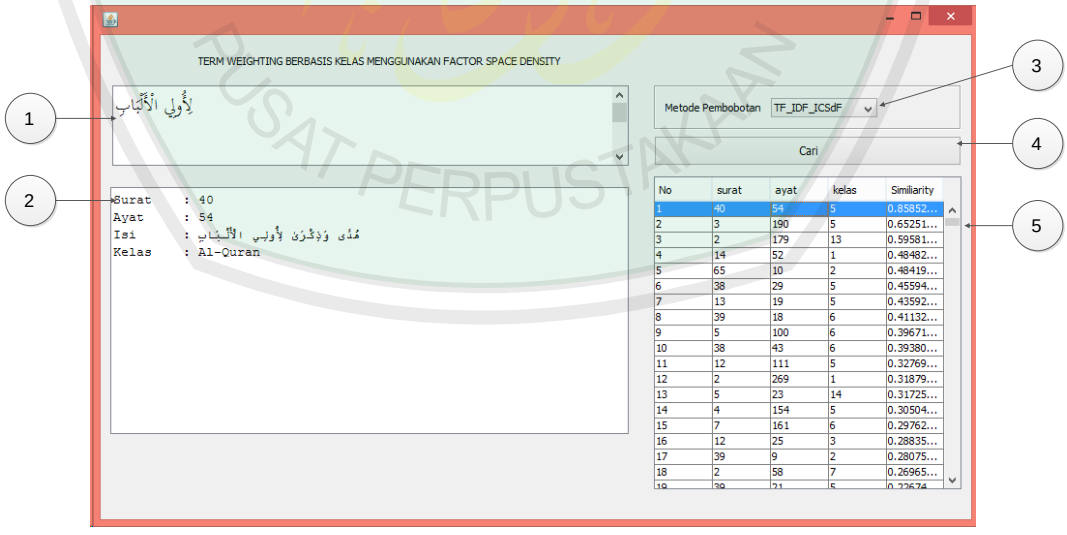
kemiripan yang paling tinggi. Hasil perhitungan *cosine similarity* sebagaimana pada tabel 4.11.

Tabel 4.11 Hasil perhitungan *cosine similarity*

id_doc	id_class	similarity
5	1	0.06451924322732819
29	1	0.013828050085401358
78	3	0.06747797394747801
94	1	0.062462237763400125
109	1	0.10934923886614745
259	4	0.7198652874518118

4.1.6 Desain dan Implementasi GUI

Di dalam desain GUI, dijelaskan kegunaan dari komponen yang ada pada aplikasi perangkingan dokemen Al-Qur'an. Tampilan aplikas seperti yang ditunjukkan pada gambar 4.19.



Gambar 4.19 Tampilan GUI Perangkingan Dokumen Al-Qur'an

Penjelasan dari tampilan aplikasi diatas antara lain:

1. TextArea input ayat

   TextArea ini berfungsi sebagai tempat untuk menuliskan kalimat yang akan di ranking.

2. TextArea detail ayat

   Berfungsi menampilkan detail penjelasan ayat yang telah dirangking, cara penggunaannya yaitu memilih ayat dari tabel hasil ranaking

3. Pilihan metode pembobotan

   Berfungsi memilih metode pembobotan yang akan digunakan yaitu metode TF.IDF, TF.IDF.ICF dan TF.IDF.ICS$_{\delta}$F.

4. Tombol cari

   Berfungsi untuk memerintahkan eksekusi pembobotan akan dimulai.

5. Tabel output ranking ayat

   Berfungsi menampilkan daftar ayat yang telah diranking berisi penjelasan surat, ayat, kelas, dan nilai similarity.

4.2 Hasil dan Uji Coba

Subbab ini menampilkan hasil pengujian metode yang dikembangkan pada penelitian ini. Pengujuian ini yang pertama yaitu uji coba *query* dari pengguna untuk mengetahui pengaruh nilai *space density* terhadap perangkingan dokumen. Metode TF.IDF.ICS$_{\delta}$F berdasarkan uji coba akan dibandingkan dengan metode pembobotan yang lainnya, yaitu TF.IDF dan TF.IDF.ICF. Pada tiap pengujian dilakukan pengukuran relevansi.

4.2.1 Lingkungan Uji Coba

Proses uji coba aplikasi dilakukan pada komputer dengan spesifikasi *processor* AMD A6-3420M APU dengan memori (RAM) 4 GB dan sistem operasi Windows 8.1. Aplikasi yang dibangun berjalan diatas *Java Runtime Standard Edition 7* dengan *database* dataset menggunakan MySQL dan library lucene.

4.2.2 Karakteristik Data Uji Coba

Data yang digunakan dalam uji coba ini merupakan *corpus* atau kumpulan dokumen teks berbahasa Arab, dimana total dokumen berjumlah 6236 ayat. Isi dari datasete Al-Qur'an terdiri dari id_doc, id_class, surat, ayat dan teks seperti yang ditunjukkan pada tabel 4.12.

Tabel 4.12 Tabel isi dokumen al-Qur'an

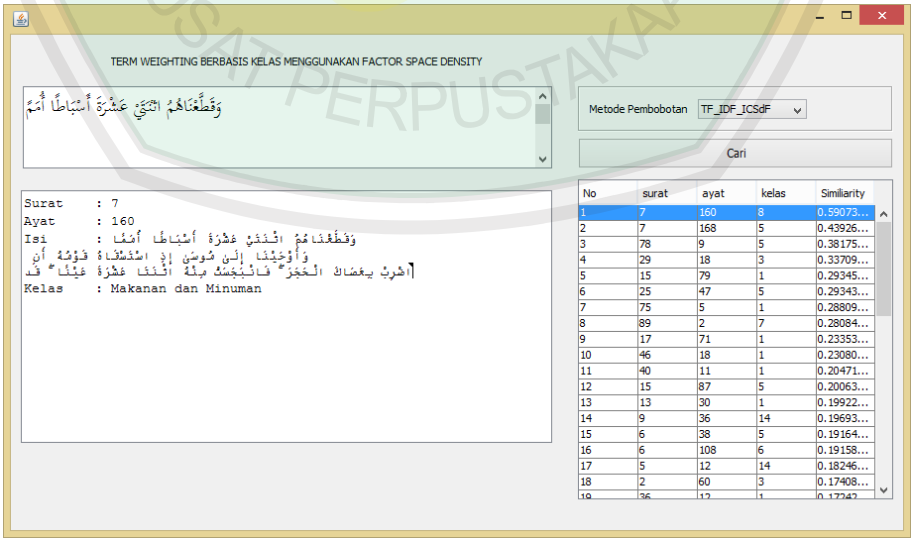
id_doc	id_class	Surat	Ayat	Text
1	1	1	1	بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ
2	7	1	2	الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ
3	1	1	3	الرَّحْمَٰنِ الرَّحِيمِ
4	1	1	4	مَالِكِ يَوْمِ الدِّينِ
5	1	1	5	إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ
6	7	1	6	اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ
7	5	1	7	صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ
8	1	2	1	بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ الم
9	1	2	2	ذَٰلِكَ الْكِتَابُ لَا رَيْبَ ۛ فِيهِ ۛ هُدًى لِلْمُتَّقِينَ
10	1	2	3	الَّذِينَ يُؤْمِنُونَ بِالْغَيْبِ وَيُقِيمُونَ الصَّلَاةَ وَمِمَّا رَزَقْنَاهُمْ يُنْفِقُونَ

Salah satu data uji perangkingan dokumen pada penelitian ini terlihat pada gambar 4.28 dimana dokumen berisi teks Bahasa Arab. Dokumen-dokumen inilah yang diproses dari tahap *preprocessing*, penghapusan *stopword*, pembentukan kata dasar, dan perangkingan. Peringkat Dokumen Hasil Pencarian

Evaluasi kemempuan sistem dalam perangkingn dokumen dapat dilakukan dengan melihat posisi dokumen yang relevan yang terdapat pada urutan pertama terhadap ke-100 input yang diujikan. Posisi dokumen yang relevan sesuai dengan input pengguna secara keseluruhan dapat dilihat pada lampiran 1.

Input tersebut akan diujikan ke dalam aplikasi perangkingan dokumen yang telah dibangun. Salah satu contoh hasil pencarian dapat dilihat pada gambar 4.20. Pada gambar tersebut terlihat bahwa input yang diujikan adalah input ke-1. Pada bagian bawahannya terdapat tabel yang berisi daftar dokumen-dokumen hasil pencarian.

Menurut hasil perangkingan data yang di-ritrieve adalah dokumen yang berasal dari surat 23 ayat 67. Pada gambar 4.20 dapat kita lihat bahwa dokumen yang dimaksud berada pada urutan nomor 1 dari daftar dokumen hasil pencarian. Berdasarkan hasil pencarian tersebut, dapat dikatakan bahwa posisi dokumen relevan pada hasil pencarian adalah 1.



Gambar 4.20 Posisi dokumen relevan pada hasil pencarian

Semua input pada lampiran 1 akan di uji hasilnya seperti yang dilakukan pada input 1 kemudian dihitung jumlah keseluruhan dokumen relevannya. Dari hasil pencarian semua input tersebut akan dihitung rata-rata posisi dokumen relevan. Pengujian ini dilakukan dengan menggunakan metode pembobotan yang diajukan dan dibandingkan dengan beberapa metode pembobotan yang ada sebelumnya. Metode pembobotan yang ada sebelumnya.

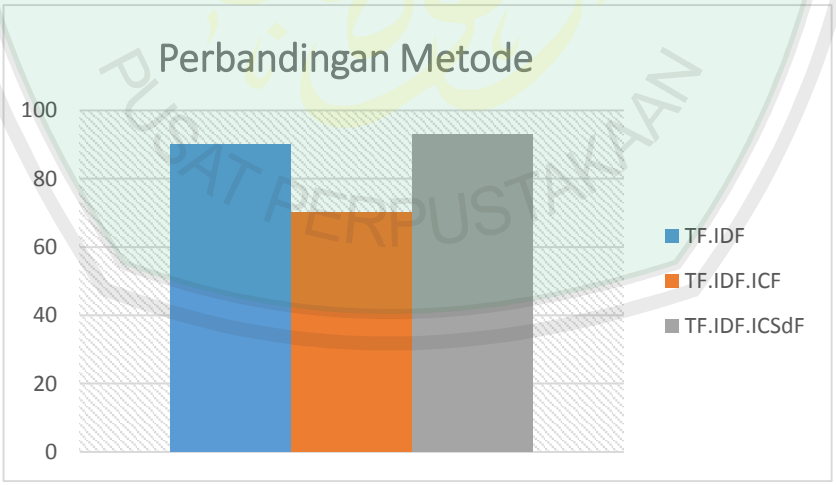
Nilai akurasi diperoleh dari jumlah *term* yang sama dalam ayat dari hasil perangkinan, semakin banyak *term* yang sama dengan *input user* maka semakin tinggi nilai *similarity* dan menjadikan ayat tersebut relevan. Namun, tidak semua ayat yang ter-*retrieve* dan menjadi urutan pertama relevan dengan input user, ada beberapa dari hasil pengujian ayat yang menjadi urutan pertama tapi tidak relevan. Seperti pada percobaan berikut ini :

Tabel 4.13 Tabel Percobaan Pengujian Metode

Percobaan		
Input user	Ayat سَرِيعُ الْحِسَابِ	Arti Maha cepat hisab-Nya
	Hasil	
TF.IDF	نُسَارِعُ لَهُمْ فِي الْخَيْرَاتِ ۚ بَلْ لَا يَشْعُرُونَ	Kami bersegera memberikan kebaikan-kebaikan kepada mereka? Tidak, sebenarnya mereka tidak sadar
TF.IDF.ICF	لِيَجْزِيَ اللَّهُ كُلَّ نَفْسٍ مَا كَسَبَتْ ۚ إِنَّ اللَّهَ سَرِيعُ الْحِسَابِ	agar Allah memberi pembalasan kepada tiap-tiap orang terhadap apa yang ia usahakan. Sesungguhnya Allah Maha cepat hisab-Nya.
TF.IDF.ICS$_{\delta}$F	نُسَارِعُ لَهُمْ فِي الْخَيْرَاتِ ۚ بَلْ لَا يَشْعُرُونَ	Kami bersegera memberikan kebaikan-kebaikan kepada mereka? Tidak, sebenarnya mereka tidak sadar

Percobaan yang ditunjukkan pada tabel 4.13 menunjukkan metode pembobotan TF.IDF.ICF berhasil me-*retrive* ayat yang relevan, yaitu ayat yang didalamnya terdapat *term* سَرِيعُ الْحِسَابِ dan mengandung arti Maha cepat hisab-Nya. Sementara metode pembobotan TF.IDF dan TF.IDF.ICS$_{\delta}$F tidak berhasil me-*retrive* ayat dan arti yang relevan.

Pada gambar 4.21 menunjukkan perbandingan nilai akurasi pada hasil pencarian masing-masing metode dengan beberapa variasi query menunjukkan bahwa metode TF.IDF.ICS$_{\delta}$F memiliki memiliki nilai akurasi lebih tinggi dari dua metode lainnya yaitu nilai akurasi sebesar 93 %. Sedangkan metode TF.IDF menempati posisi kedua dengan nilai akurasi sebesar 90 %. Sedangkan metode pembobotan TF.IDF.ICF mengalami penurunan performa yang signifikan yaitu dengan nilai akurasi sebesar 70 %.



Gambar 4.21 Grafik Akurasi Perbandingan Metode

Dari keseluruhan hasil pengujian, dapat dilihat bahwa metode pembobotan yang diajukan yaitu metode TF.IDF.ICS$_{\delta}$F terbukti baerhasil diimplementasikan dalam perangkingan dokumen Al-Qur'an berbahasa Arab dengan tingkat relevansi

yang tinggi. Metode ini mampu mencari dokumen yang relevan terhadap *query* yang dimasukkan dengan tidak hanya memperhatikan indeks dokumen, tetapi juga memperhatikan kelas, bahkan memperhatikan juga jumlah kepadatan anggota dalam suatu kelas. Hal ini memungkinkan metode ini untuk mendapatkan dokumen yang relevan dari kategori yang tepat sesuai dengan karakteristik *query* yang dimasukkan.

Berdasarkan hasil pengujian juga dapat dilihat bahwa metode TF.IDF memiliki akurasi yang lebih tinggi dibandingkan dengan metode TF.IDF.ICF. hal ini menunjukkan bahwa metode TF.IDF lebih *stabil* dalam me-*retrieve* dokumen dari pada metode TF.IDF.ICF.

4.3 Integrasi Islam

Perangkingan dokumen Al-Qur'an adalah proses pencarian ayat yang sesuai dilakukan dengan mengukur kemiripan ayat yang akan dicari dengan dokumen terkait (document similarity). Semakin banyak ciri yang sesuai maka semakin tinggi nilai similarity yang dihasilkan. Hal ini sesuai dengan firman Allah:

يَا أَيُّهَا النَّاسُ إِنَّا خَلَقْنَاكُمْ مِنْ ذَكَرٍ وَأُنْثَىٰ وَجَعَلْنَاكُمْ شُعُوبًا وَقَبَائِلَ لِتَعَارَفُوا ۚ إِنَّ أَكْرَمَكُمْ عِنْدَ اللَّهِ أَتْقَاكُمْ ۚ إِنَّ اللَّهَ عَلِيمٌ خَبِيرٌ

"Hai manusia, sesungguhnya Kami menciptakan kamu dari seorang laki-laki dan seorang perempuan dan menjadikan kamu berbangsa-bangsa dan bersuku-suku supaya kamu saling kenal-mengenal. Sesungguhnya orang yang paling mulia diantara kamu disisi Allah ialah orang yang paling taqwa diantara kamu. Sesungguhnya Allah Maha Mengetahui lagi Maha Mengenal."

Allah menciptakan manusia berbangsa-bangsa dan bersuku-suku yang berbeda untuk saling mengenal. Dari perbedaan tersebut agar mereka mengetahui masing-masing ciri dari manusia. Hal ini diperkuat dengan ayat Al-Qur'an surat Ar Rum : 13.

وَمِنْ آيَاتِهِ خَلْقُ السَّمَاوَاتِ وَالْأَرْضِ وَاخْتِلَافُ أَلْسِنَتِكُمْ وَأَلْوَانِكُمْ ۚ إِنَّ فِي ذَٰلِكَ لَآيَاتٍ لِلْعَالِمِينَ

"Dan di antara tanda-tanda kekuasaan-Nya ialah menciptakan langit dan bumi dan berlain-lainan bahasamu dan warna kulitmu. Sesungguhnya pada yang demikan itu benar-benar terdapat tanda-tanda bagi orang-orang yang mengetahui."

Ayat di atas menjelaskan bahwa sebagian dari tanda-tanda kekuasaan-Nya adalah keragaman bahasa dan warna kulit manusia. Dengan keragaman tersebut kita mengetahui ciri khas dari masing-masing orang. Setiap orang memiliki dua mata, dua alis, satu hidung, dua buah pelipis, satu mulut, dan dua pipi. Meskipun demikian, antara satu dengan yang lainnya tidak memiliki kesamaan. Bahkan dibedakan satu sama lain antara jalannya, sikapnya atau pembicaraannya. Ciri khas yang dimiliki oleh setiap orang dapat dijadikan sebagai pembeda antara manusia satu dengan manusia yang lain. Dengan begitu apabila ada seseorang yang hilang kita dapat menemukan orang tersebut berdasarkan ciri khas yang dimilikinya. Metode yang digunakan dalam penelitian ini juga mencari persamaan ciri dokumen yang dicari terhaap dokumen kunci. Kemiripan dokumen dinilai dari frekuensi kata yang terdapat pada suatu dokumen, semakin banyak frekuensi kata yang dikandung maka semakin tinggi nilai *similarity*-nya dan dokumen dinyatakan sebagai dokumen yang mirip.

BAB V

KESIMPULAN DAN SARAN

Pada bab terakhir ini, ditarik beberapa kesimpulan yang didapat dari hasil penelitian ini, juga saran-saran yang dapat digunakan sebagai bahan pertimbangan untuk pengembangan atau riset selanjutnya.

5.1 Kesimpulan

Berdasarkan aplikasi yang telah dibuat dan hasil yang didapat dari serangkaian uji coba yang telah dilakukan, maka dapat ditarik beberapa kesimpulan atas penelitian ini sebagai berikut :

1. TF.IDF.ICS$_\delta$F dapat diaplikasikan pada perangkingan dokumen berbahasa Al-Qur'an yang berbahasa Arab.
2. Pada penelitian ini terbukti bahwa metode TF.IDF.ICS$_\delta$F menghasilkan pencarian yang lebih baik dari pada menggunakan ICF saja, dengan nilai akurasi 93 %.

5.2 Saran

Beberapa saran setelah dilakukan penelitian ini adalah sebagai berikut :

1. Indeks kelas yang digunakan pada penelitian ini berdasarkan kelas tunggal. Oleh karena itu dapat dilakukan penelitian lebih lanjut untuk memenuhi kebutuhan multi-kelas.
2. Kesesuaian hasil pencarian ditentukan oleh *input* pengguna berdasarkan kesamaan *term*. Perlu adanya *expansion* untuk meningkatkan kemampuan pencarian berdasarkan makna.

DAFTAR PUSTAKA

Al-Taani, A. T. a. A. M. A.-G., 2010. *Searching Concepts and Keywords in the Holy Quran.* Jordan, Department of Computer Science, Yarmouk University.

Cios, K., 2007. *Data Mining A Knowledge Discovery Approach.* New : Springer..

El Emary, I. a. J. A., 2005. Designing and Building an Automatic Information Retrieval System for Handling the Arabic Data. *American Journal of Applied Sciences,* II(11), pp. 1520-1525.

Esraa, E. B. N. a. M. T., 2010. An Efficient Ranking Module for an Arabic Search Engine. *IJCSNS International Journal of Computer Science and Network Security,* 10(2), pp. 218-225.

Harrag, F. A. H.-C. a. E. E.-Q., 2008. *Vector space model for Arabic information retrieval—application to "Hadith" indexing.* Ostrava, Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008.

Hersh, W., 2003. *Information Retrieval: A Health and Biomedical Perspective: A Health and Biomedical Perspective.* New York: Springer.

Hmeidi, I. G. K. a. M. E., 1997. Design and Implementation of Automatic Indexing for. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE.* , IV(10), p. 867 – 881.

Ibrahim, A. E.-K., 2007. Arabic Information Retrieval. *Annual Review of Information Science and Technology,* 41(1), pp. 505-533.

Jabal, 2010. *Mushaf Al Azhar.* Bandung: Jabal Raudatul Jannah .

Larkey, L. S. B. L. a. C. M. E., 2007. Light Stemming. *Springer Link: Text, Speech and Language* , Volume XXXVIII, pp. 221-243 .

Machnik, Ł., 2004. Documents Clustering Techniques. *IBIZA 2004, Annales UMCS Informatica Poland,* II(1), pp. 401-411.

Mahmoud, R. S. M. a. Z. K., 2009. Improving Arabic information retrieval system using n-gram method. *WSEAS Transactions on Computers,* X(2011), pp. 125-133.

Manning, C. D. P. R. a. H. S., 2008. *Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Mesleh, A. M., 2008. Support Vector Machines based Arabic Language Text Classification System: Feature Selection Comparative Study. In: T.

Sobh, ed. *Advances in Computer and Information Sciences and Engineering.* Springer Netherlands: Springer Netherlands, pp. 11-16.

Murugesan, A. K. a. B. J. Z., 2011. *Clustering, A New Term Weighting Scheme for Document.* Las Vegas, Nevada, 7th Int. Conf. Data Min.(DMIN 2011-WORLDCOMP 2011).

Quthb, S., 2004. *Tafsir Fi Zhilalil Qur`an Jld 10.* Jakarta: Gema Insani Press.

Ren, F. a. M. G. S., 2013. Class-indexing-based term weighting for automatic text classification. *Information Sciences,* pp. 109-125.

Salton, G., 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of.* s.l.:Reading: Addison-Wesley.

Tata, S. a. J. M. P., 2007. Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record,* 36(2), pp. 7-12.

No	Input	Nilai similarity Terbesar			Output (surat : ayat)			Relevansi		
		TF.IDF	TF.IDF.ICF	TF.IDF.ICSdF	TF.IDF	TF.IDF.ICF	TF.IDF.ICSdF	TF.IDF	TF.IDF.ICF	TF.IDF.ICSdF
1	وَقَطَّعْنَاهُمُ اثْنَتَيْ عَشْرَةَ أَسْبَاطًا أُمَمًا	0.639	0.639	0.655	7:160	7:160	7:160	Ya	Ya	Ya
2	نَطْبَعُ عَلَىٰ قُلُوبِهِمْ	0.959	1.000	0.988	30:59	30:59	30:59	Ya	Ya	Ya
3	حَيَّةٌ تَسْعَىٰ	0.833	0.811	0.842	20:20	20:20	20:20	Ya	Ya	Ya
4	لِأُولِي الْأَلْبَابِ	0.789	0.958	0.859	40:54	40:54	40:54	Ya	Ya	Ya
5	الطَّلَاقُ مَرَّتَانِ فَإِمْسَاكٌ بِمَعْرُوفٍ	0.534	0.755	0.526	2:229	77:29	2:241	Ya	Tidak	Ya
6	الصيام شهر رمضان	0.508	0.579	0.556	2:185	2:185	2:185	Ya	Ya	Ya
7	اعْتَزِلُوا النِّسَاءَ فِي الْمَحِيضِ	0.640	0.707	0.653	2:222	69:39	26:212	Ya	Tidak	Tidak
8	لِشَهْرِ الْحَرَامِ	0.732	0.952	0.791	2:194	97:3	2:194	Ya	Tidak	Ya
9	قطع يد السارق	0.754	0.947	0.756	5:38	12:81	5:38	Ya	Ya	Ya
10	يَرْمُونَ الْمُحْصَنَاتِ	0.722	0.898	0.735	24:23	24:23	24:23	Ya	Ya	Ya
11	وَجَاهِدُوا فِي اللَّهِ	0.596	0.917	0.642	25:52	25:52	25:52	Ya	Ya	Ya
12	اجْلِدُوا الزِّنَا	0.578	0.721	0.572	24:2	41:21	24:2	Ya	Tidak	Ya
13	أُجِيبَتْ دَعْوَتُكُمَا	0.606	0.651	0.644	10:89	35:14	10:89	Ya	Ya	Ya
14	الطَّلَاقُ مَرَّتَانِ فَإِمْسَاكٌ بِمَعْرُوفٍ أَوْ تَسْرِيحٌ بِإِحْسَانٍ	0.674	0.729	0.659	2:229	16:6	2:229	Ya	Tidak	Ya
15	فَتَيَمَّمُوا صَعِيدًا طَيِّبًا فَامْسَحُوا بِوُجُوهِكُمْ وَأَيْدِيكُمْ	0.371	0.518	0.380	88:8	5:6	5:6	Tidak	Ya	Ya
16	آتَى الزَّكَاةَ	0.696	0.913	0.740	79:18	23:4	79:18	Tidak	Ya	Tidak
17	الْمُوفُونَ بِعَهْدِهِمْ إِذَا عَاهَدُوا	0.597	0.795	0.550	13:20	9:1	3:76	Ya	Ya	Ya
18	الصَّابِرِينَ فِي الْبَأْسَاءِ	0.478	0.491	0.423	74:7	27:33	74:7	Ya	Tidak	Ya
19	زُيِّنَ لِلَّذِينَ كَفَرُوا الْحَيَاةُ	0.592	0.499	0.581	2:212	52:47	2:212	Ya	Tidak	Ya
20	شَيْطَانٍ رَجِيمٍ	0.847	0.990	0.896	15:17	15:17	15:17	Ya	Ya	Ya
21	أُوتِيَ كِتَابَهُ بِشِمَالِهِ	0.688	0.976	0.745	69:25	56:41	56:41	Ya	Tidak	Tidak
22	أَنْفَقُوا مِمَّا رَزَقْنَاهُمْ سِرًّا وَعَلَانِيَةً	0.693	0.781	0.716	14:31	14:31	14:31	Ya	Ya	Ya
23	الصَّابِرِينَ فِي الضَّرَّاءِ	0.418	0.596	0.507	74:7	7:94	7:94	Tidak	Ya	Ya
24	طَعَامِ الْمِسْكِينِ	0.679	0.641	0.649	74:44	68:24	76:8	Ya	Tidak	Ya
25	نُفِخَ فِي الصُّورِ	0.874	0.985	0.909	50:20	69:13	50:20	Ya	Ya	Ya
26	فَاعْتَرَفُوا بِذَنْبِهِمْ	0.604	0.790	0.595	67:11	81:9	40:11	Ya	Tidak	Ya
27	أَجْرٌ غَيْرُ مَمْنُونٍ	0.876	0.992	0.943	41:8	41:8	41:8	Ya	Ya	Ya
28	لُجُّوا فِي عُتُوٍّ وَنُفُورٍ	0.854	0.948	0.897	67:21	67:21	67:21	Ya	Ya	Ya
29	قُوا أَنْفُسَكُمْ وَأَهْلِيكُمْ نَارًا	0.704	0.771	0.630	38:64	20:23	38:64	Ya	Ya	Ya
30	الزَّانِي يَنْكِحُ زَانِيَةً	0.681	0.835	0.702	24:3	17:32	24:3	Ya	Tidak	Ya
31	قُتِلَ مَظْلُومًا	0.624	0.969	0.706	17:33	17:33	17:33	Ya	Ya	Ya
32	فَاسْتَقِمْ كَمَا أُمِرْتَ وَمَنْ تَابَ مَعَكَ وَلَا تَطْغَوْا	0.917	0.994	0.958	11:112	11:112	11:112	Ya	Ya	Ya
33	اذْهَبُوا بِقَمِيصِي	0.582	0.976	0.720	12:27	12:26	12:27	Ya	Ya	Ya
34	وَيَحْمِلُ الْمَلَكُ عَرْشَ	0.639	0.926	0.573	69:17	20:5	69:17	Ya	Tidak	Ya
35	دَعْوَةُ الْحَقِّ	0.495	0.547	0.495	22:62	26:72	22:62	Ya	Tidak	Ya
36	عُقْبَى الدَّارِ	0.637	0.817	0.709	13:24	13:24	13:24	Ya	Ya	Ya
37	سِلْسِلَةٍ ذَرْعُهَا	0.739	0.733	0.750	69:32	69:32	69:32	Ya	Ya	Ya
38	احْتَمَلَ السَّيْلُ زَبَدًا رَابِيًا	0.653	0.897	0.749	13:17	13:17	13:17	Ya	Ya	Ya
39	كَبَاسِطِ كَفَّيْهِ إِلَى الْمَاءِ	0.654	0.924	0.781	13:14	13:14	13:14	Ya	Ya	Ya
40	دُعَاءُ الْكَافِرِينَ فِي ضَلَالٍ	0.568	0.816	0.562	22:12	7:61	22:12	Ya	Tidak	Ya
41	وَالْبَلَدُ الطَّيِّبُ يَخْرُجُ نَبَاتُهُ بِإِذْنِ رَبِّهِ	0.640	0.515	0.536	7:58	89:8	7:58	Ya	Tidak	Ya
42	قَوْمًا عَمِينَ	0.585	0.925	0.761	27:66	27:66	27:66	Ya	Ya	Ya
43	نَذَرَ مَا يَعْبُدُ آبَاؤُنَا	0.616	0.894	0.583	7:70	16:35	7:70	Ya	Ya	Ya
44	نَضْرِبُهَا لِلنَّاسِ	0.714	0.802	0.699	39:27	16:74	29:43	Ya	Tidak	Ya
45	تَرَى الشَّمْسَ إِذَا طَلَعَتْ تَزَاوَرُ عَنْ كَهْفِهِمْ ذَاتَ الْيَمِينِ	0.411	0.506	0.404	18:17	20:29	18:17	Ya	Tidak	Ya
46	تَحْسَبُهُمْ أَيْقَاظًا	0.352	0.595	0.429	18:18	18:18	18:18	Ya	Ya	Ya
47	فَانْبَجَسَتْ مِنْهُ اثْنَتَا عَشْرَةَ عَيْنًا	0.635	0.766	0.651	7:160	7:160	7:160	Ya	Ya	Ya
48	اضْرِبْ بِعَصَاكَ الْحَجَرَ	0.472	0.472	0.506	79:21	15:80	15:80	Tidak	Ya	Ya
49	ضَلُّوا السَّبِيلَ	0.674	0.764	0.651	4:167	40:74	4:167	Ya	Ya	Ya
50	عبا دة الاصنام	0.408	0.507	0.462	25:77	14:35	25:77	Tidak	Ya	Tidak
51	القصاص فى القتلى	0.462	0.770	0.557	3:62	2:179	3:62	Tidak	Ya	Tidak
52	التوبة	0.566	0.537	0.623	25:71	24:5	24:5	Ya	Ya	Ya
53	يُسْقَىٰ بِمَاءٍ وَاحِدٍ	0.376	0.593	0.509	26:79	67:30	67:30	Tidak	Ya	Ya
54	الطَّعَامُ حَلَالًا طَيِّبًا	0.550	0.719	0.598	5:88	5:88	5:88	Ya	Ya	Ya
55	جِذْعِ النَّخْلَةِ	0.568	0.570	0.587	19:25	19:25	19:25	Ya	Ya	Ya
56	يُؤْلُونَ مِنْ نِسَائِهِمْ تَرَبُّصُ أَرْبَعَةِ	0.615	0.678	0.651	2:226	2:226	2:226	Ya	Ya	Ya
57	أَوْفُوا بِالْعَهْدِ	0.629	0.680	0.579	13:20	9:1	3:76	Ya	Tidak	Ya
58	أَنْفَقُوا مِمَّا رَزَقْنَاهُمْ	0.841	0.989	0.888	8:3	8:3	8:3	Ya	Ya	Ya
59	جَاءُوا بِالْإِفْكِ عُصْبَةٌ	0.673	0.749	0.726	24:11	24:11	24:11	Ya	Ya	Ya
60	عَاقِبَةُ الدَّارِ	0.637	0.817	0.709	13:24	13:24	13:24	Ya	Ya	Ya
61	الصِّرَاطَ الْمُسْتَقِيمَ	1.000	1.000	1.000	36:4	15:41	36:4	Ya	Ya	Ya
62	فِي الْمَهْدِ صَبِيًّا	0.749	0.809	0.786	74:14	74:14	74:14	Ya	Ya	Ya

63	أَضَلُّ مِمَّنْ يَدْعُو	0.711	0.934	0.761	46:5	46:5	46:5	Ya	Ya	Ya
64	يَهْدِي الْقَوْمَ الظَّالِمِينَ	0.553	0.729	0.541	23:94	92:12	23:94	Ya	Tidak	Ya
65	كَرَّةً فَنَتَبَرَّأَ	0.535	0.916	0.700	2:167	2:167	2:167	Ya	Ya	Ya
66	يَأْمُرُكُمْ بِالْفَحْشَاءِ	0.738	0.919	0.796	2:169	2:169	2:169	Ya	Ya	Ya
67	تَنْكِحَ زَوْجًا غَيْرَهُ	0.600	0.832	0.549	2:230	24:3	24:3	Ya	Ya	Ya
68	مَنْ يَعْمَلْ مِنَ الصَّالِحَاتِ وَهُوَ مُؤْمِنٌ	0.759	0.672	0.658	31:8	29:9	31:8	Ya	Ya	Ya
69	نَكَحَ آبَاؤُكُمْ مِنَ النِّسَاءِ	0.674	0.773	0.603	4:22	24:3	4:22	Ya	Ya	Ya
70	فَوَسْوَسَ إِلَيْهِ الشَّيْطَانُ	0.608	0.608	0.801	114:5	114:5	114:5	Ya	Ya	Ya
71	طَفِقَا يَخْصِفَانِ	0.402	0.552	0.402	20:121	20:121	20:121	Ya	Ya	Ya
72	شَجَرَةِ الْخُلْدِ وَمُلْكٍ لَا يَبْلَىٰ	0.570	0.587	0.501	20:120	84:15	20:120	Ya	Tidak	Ya
73	اهْبِطَا مِنْهَا جَمِيعًا	0.556	0.796	0.671	7:24	7:24	7:24	Ya	Ya	Ya
74	أَعْرَضَ عَنْ ذِكْرِي	1.000	1.000	1.000	74:49	74:49	74:49	Ya	Ya	Ya
75	حَشَرْتَنِي أَعْمَىٰ	0.677	0.957	0.779	20:125	20:125	20:125	Ya	Ya	Ya
76	فِي الدِّينِ وَأَخْرَجُوكُمْ مِنْ دِيَارِكُمْ	0.581	0.705	0.640	60:69	60:69	60:69	Ya	Ya	Ya
77	الْحَيَاةَ الدُّنْيَا مَتَاعٌ	0.591	0.542	0.552	43:35	87:13	40:39	Ya	Tidak	Ya
78	الْآخِرَةَ هِيَ دَارُ الْقَرَارِ	0.685	0.941	0.762	40:39	40:39	40:39	Ya	Ya	Ya
79	يُحِبُّ الْمُقْسِطِينَ	0.440	0.506	0.381	60:8	10:47	10:47	Ya	Tidak	Tidak
80	فَاعْتَرَفُوا بِذَنْبِهِمْ	0.604	0.790	0.595	67:1	81:9	40:11	Ya	Tidak	Ya
81	بِأَيِّ ذَنْبٍ قُتِلَتْ	1.000	1.000	1.000	81:9	81:9	81:9	Ya	Ya	Ya
82	يَخْشَوْنَ رَبَّهُمْ بِالْغَيْبِ	0.772	0.956	0.827	67:12	67:12	67:12	Ya	Ya	Ya
83	الْمَاءُ حَمَلْنَاكُمْ فِي الْجَارِيَةِ	0.756	0.546	0.728	69:11	69:11	69:11	Ya	Ya	Ya
84	يَبْغُونَهَا عِوَجًا	0.671	0.928	0.761	7:45	18:1	7:45	Ya	Ya	Ya
85	غَرَّتْهُمُ الْحَيَاةُ الدُّنْيَا	0.516	0.758	0.569	7:51	25:65	7:51	Ya	Ya	Ya
86	أَصْحَابُ النَّارِ	0.683	0.886	0.679	40:6	80:36	2:39	Ya	Tidak	Ya
87	أَتَأْتُونَ الْفَاحِشَةَ	0.659	0.877	0.661	27:54	37:28	27:54	Ya	Tidak	Ya
88	أَصْبَحُوا فِي دَارِهِمْ جَاثِمِينَ	0.847	0.935	0.859	7:78	11:67	11:67	Ya	Ya	Ya
89	جِنَّتُكُمْ بِبَيِّنَةٍ	0.559	0.931	0.699	43:63	7:105	7:105	Ya	Ya	Ya
90	وَنَزَعَ يَدَهُ	0.688	0.544	0.661	26:33	20:62	26:33	Ya	Tidak	Ya
91	تَنَازَعُوا أَمْرَهُمْ	0.662	0.556	0.626	20:62	20:62	20:62	Ya	Ya	Ya
92	قُلِ الرُّوحُ مِنْ أَمْرِ رَبِّي	0.664	0.574	0.606	17:85	56:89	17:85	Ya	Tidak	Ya
93	لَأُقَطِّعَنَّ أَيْدِيَكُمْ وَأَرْجُلَكُمْ	0.704	0.415	0.627	7:124	27:32	7:124	Ya	Ya	Ya
94	يُسَبِّحُ الرَّعْدُ	0.470	0.692	0.541	79:3	13:13	13:13	Tidak	Ya	Ya
95	يَسْتَوِي الْأَعْمَىٰ	0.796	0.934	0.822	35:19	15:93	35:19	Ya	Tidak	Ya
96	مَتَاعٍ زَبَدٌ مِثْلُهُ	0.618	0.897	0.738	13:17	13:17	13:17	Ya	Ya	Ya
97	سَرِيعُ الْحِسَابِ	0.679	0.893	0.752	23:56	14:51	23:56	Tidak	Ya	Tidak
98	يُسَارِعُونَ فِي الْخَيْرَاتِ	0.802	0.889	0.816	23:56	14:51	23:56	Ya	Tidak	Ya
99	وَيَنْهَىٰ عَنِ الْفَحْشَاءِ	0.523	0.760	0.568	53:42	2:169	2:169	Tidak	Ya	Ya
100	يُخْرِبُونَ بُيُوتَهُمْ بِأَيْدِيهِمْ	0.406	0.592	0.466	59:2	59:2	59:2	Ya	Ya	Ya
	Total Relevan							90	70	93