TESIS – KI142502

PEMBOBOTAN KATA BERBASIS *PREFERENCE* UNTUK PERANGKINGAN DOKUMEN FIQIH BERBAHASA ARAB

KHADIJAH FAHMI HAYATI HOLLE
NRP 5113 201 036

DOSEN PEMBIMBING
Dr. Agus Zainal Arifin, S.Kom., M.Kom.
Diana Purwitasari, S.Kom, M.Sc.

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI CERDAS DAN VISUALISASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS - KI142502

PREFERENCE BASED TERM WEIGHTING FOR ARABIC FIQH DOCUMENT RANKING

KHADIJAH FAHMI HAYATI HOLLE
NRP 5113 201 036

SUPERVISOR
Dr. Agus Zainal Arifin, S.Kom., M.Kom.
Diana Purwitasari, S.Kom, M.Sc.

MAGISTER PROGRAM
INTELLIGENCE COMPUTATIONAL AND VISUALISATION
DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2015

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Teknik Informatika (M.Kom)
di
Institut Teknologi Sepuluh Nopember

oleh :
Khadijah Fahmi Hayati Holle
Nrp. 5113201036

Tanggal Ujian : 13 Januari 2015
Periode Wisuda : Maret 2015

Disetujui oleh :

1. Dr. Agus Zainal Arifin, S.Kom., M.Kom.
   NIP. 19720809 199512 1 001 (Pembimbing 1)

2. Diana Purwitasari, S.Kom., M.Sc.
   NIP. 19780410 200312 2 001 (Pembimbing 2)

3. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
   NIP. 19751220 200112 2 002 (Penguji 1)

4. Isye Arieshanti, S.Kom., M.Phil.
   NIP. 19780412 200604 2 001 (Penguji 2)

5. Wijayanti Nurul Khotimah, S.Kom., M.Sc.
   NIP. 19860312 201212 2 004 (Penguji 3)

Direktur Program Pasca Sarjana

Prof.Dr.Ir. Adi Soeprijanto, MT
NIP. 19640405 199002 1 001

PEMBOBOTAN KATA BERBASIS *PREFERENCE* UNTUK PERANGKINGAN DOKUMEN FIQIH BERBAHASA ARAB

Nama Mahasiswa	:	Khadijah Fahmi Hayati Holle
NRP	:	5113 201 036
Pembimbing	:	Dr. Agus Zainal Arifin, S.Kom., M.Kom. Diana Purwitasari, S.Kom., M.Sc.

ABSTRAK

Dalam pencarian, selain kesesuaian *query* dengan hasil pencarian, terdapat penilaian subjektif pengguna yang diharapkan turut menjadi faktor penentu dalam perangkingan dokumen. Aspek preferensi tersebut tampak pada pencarian dokumen fiqih. Seseorang cenderung mengutamakan metodologi fiqih tertentu meskipun tidak mengabaikan pendapat metodologi fiqih lain. Faktor preferensi menjadi hal yang diperlukan selain relevansi dalam perangkingan dokumen. Oleh karena itu, pada penelitian ini diajukan metode pembobotan kata berbasis preferensi untuk merangkingkan dokumen sesuai dengan preferensi pengguna. Metode yang diajukan digabungkan dengan pembobotan kata berbasis indeks dokumen dan buku sehingga mampu memperhatikan aspek kesesuaian (*relevance*) dan keutamaan (*preference*). Metode pembobotan yang diusulkan disebut dengan *Invers Preference Frequency with* α *value* (IPF_α). Langkah pembobotan yang diusulkan yaitu dengan perhitungan nilai preferensi *term* dengan pembobotan IPF. Kemudian nilai preferensi dari *term* dokumen yang sama dengan *term query* dikalikan dengan $\boldsymbol{\alpha}$ sebagai penguat. IPF_α digabungkan dengan metode pembobotan yang telah ada menjadi $TF.IDF.IBF.IPF_\alpha$. Pengujian metode yang diusulkan menggunakan dataset dari beberapa dokumen fiqih berbahasa Arab. Evaluasi menggunakan perhitungan *recall*, *precision*, dan *f-measure*. Hasil uji coba menunjukkan bahwa dengan pembobotan $TF.IDF.IBF.IPF_\alpha$ diperoleh perangkingan dokumen dengan urutan yang tepat dan sesuai dengan preferensi pengguna. Hal ini ditunjukkan dengan nilai maksimal *recall* mencapai 75%, *precision* 100%, dan *f-measure* 85.7%.

Kata kunci: Perangkingan Dokumen, Temu Kembali Dokumen, Pembobotan Kata, Preferensi, $IPF_\propto$.

PREFERENCE BASED TERM WEIGHTING FOR ARABIC FIQH DOCUMENT RANKING

Student Name	: Khadijah Fahmi Hayati Holle
NRP	: 5113201036
Supervisor	: Dr. Agus Zainal Arifin, S.Kom, M.Kom
	Diana Purwitasari, S.Kom, M.Sc.

ABSTRACT

.

In document retrieval, besides suitability of query with search results, there is a subjective assessment that user are expected to be deciding factor in document ranking. This preference aspect referred at the *fiqh* document searching. People tend to prefer on certain *fiqh* methodology without reject other *fiqh* methodologies. So, it is necessary to investigate preference factor beside relevance factor in the document ranking. Therefore, this research proposed a method of term weighting based on preference to rank documents according to user preference. Proposed method also combined with term weighting based on documents index and books index so it seeing relevance and preference aspect. The proposed method is Inverse Preference Frequency with α value (IPF_α). In this method, we calculate preference value by IPF term weighting. Then, preference values of terms that equal with query are multiplied by α. IPF_α combined with existing weighting methods become $TF.IDF.IBF.IPF_\alpha$. Experiment of proposed method is using dataset of several Arabic *fiqh* documents. Evaluation uses recall, precision, and f-measure calculations. Proposed term weighting method is obtained to rank document in the right order according to user preference. It is shown from the result with recall value reach 75%, precision 100%, and f-measure 85.7%.

***Keywords**:* Document Ranking, Document Retrieval, Term weighting, Preference, $IPF_\propto$.

KATA PENGANTAR

Segala puji bagi Allah SWT sehingga buku tesis ini dapat diselesaikan dengan baik. Meski dalam menyelesaikan buku ini banyak ditemui kesulitan, namun berkat bantuan dan bimbingan berbagai pihak, akhirnya Penulis berhasil menyelesaikan buku ini. Sholawat dan salam semoga selalu tercurahkan kepada Nabi Muhammad SAW. Pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang membantu penulis dalam menyelesaikan tesis ini baik secara langsung maupun tidak langsung.

1. Kepada Almarhummah Muslimatul Umroh yang telah mendidik dan membesarkan Penulis hingga 20 tahun. Keluarga Penulis, Bapak Ali AR Holle, Ibu Khisniati, Mas Husni Wardana Holle, mbak Amalia Firdausi Holle, dan Adik Mukhlish Jamal Musa Holle yang telah memberikan doa, pengertian, dukungan, dan pengorbangan yang besar selama Penulis menyelesaikan studi ini.
2. Kepada Dosen Pembimbing Dr. Agus Zainal Arifin, S.Kom, M.Kom dan Diana Purwitasari, S.Kom, M.Sc. yang dengan sabar membimbing penulis, sehingga Penulis dapat menyelesaikan tesis ini.
3. Kepada Dr. Tohari Ahmad, S.Kom, MIT selaku Dosen wali.
4. Kepada para Dosen Penguji, Dr. Eng. Chastine Fatichah, S.Kom., M.Kom., Isye Arieshanti, S.Kom., M.Phil., dan Wijayanti Nurul Khotimah, S.Kom, M.Sc. yang telah memberikan masukan berharga.
5. Kepada teman-teman S2 seangkatan, Kartika, Suastika, Laili, Resti, Arini, Lely, Dian, Annisa, Jayanti, Amalia, Mutmainnah, Yuwanda, dan yang lainnya yang telah berusaha bersama dan saling mendukung dalam menjalani studi.
6. Kepada teman-teman seperjuangan di Surabaya, mbak Ida, mbak Ana, mbak Tawang, mbak Henik.

7. Kepada pembimbing spiritual di Malang, Ustadzah Salma, Ustadzah Zahro, Ustdzah Yulia, Ustadzah Nadia, Ustadzah Diana, Ustadzah Ivana, dan yang lainnya yang telah banyak mengajarkan Islam dan arti kehidupan yang haqiqi.
8. Kepada teman-teman dan adik seperjuangan di Malang, Atik, Hikmah, Zidna, Fitri, Roudhloh, Zia, Umi, Putri, Mega, Lilik, Ainul, Sevi, Hanifah, Ida, Nita, Ria, dan yang lainnya.
9. Kepada staf administrasi Program Pascasarjana Teknik Informatika, Bu Rini dan Bu Feni, atas pengertian dan kebijaksanaannya dalam proses pengurusan administrasi.
10. Kepada teman-teman lain yang tidak bisa Penulis sebutkan satu-persatu, terima kasih atas segala bantuan, baik berupa ide, gagasan, pemikiran, atau bahkan sekedar kesediaan mendengarkan keluh kesah Penulis.

Akhirnya, Penulis berharap, buku laporan tesis ini dapat memberikan kontribusi ilmiah bagi khasanah pengembangan riset di bidang Komputasi Cerdas dan Visualisasi.

Surabaya, Robi'uts-tsani 1436 H
Januari 2015 M

Khadijah F. H. Holle

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini semakin banyak data teks yang terdigitasi. Seiring dengan hal tersebut, kebutuhan untuk memperoleh suatu informasi dengan cepat dan tepat dari kumpulan data juga semakin meningkat. Cara sederhana untuk memperoleh informasi dari data teks dapat dilakukan dengan mencari padanan kata dari pengguna dengan kata yang dimuat dalam dokumen. Namun cara tersebut dinilai kurang efektif karena memerlukan proses yang lama, adanya redudansi hasil, serta ketidaksesuaian hasil informasi yang ditemukan (Manning dkk, 2008).

Metode-metode melakukan pencarian yang efektif dipelajari dalam bidang temu kembali informasi (Manning dkk, 2008; Ibrahim, 2007). Objek pembahasan temu kembali informasi semakin luas. Tidak hanya pada dokumen berbahasa Inggris, tapi juga dokumen berbahasa Arab meskipun masih jauh dari bahasa Inggris yang telah mendominasi temu kembali informasi selama lebih dari 50 tahun (Salton, 1989; Haidar, 2001). 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; Leah dkk, 2002).

Perangkingan dokumen merupakan salah satu pembahasan temu kembali informasi yang biasa diteliti (Manning dkk, 2008; Esraa dkk, 2010). Harrag dkk (2008) menggunakan *vector space model* untuk melakukan perangkingan pada dokumen berbahasa Arab. Pada metode ini dokumen direpresentasikan sebagai sebuah vektor yang dibentuk dari nilai-nilai *term* yang menjadi indeksnya. Nilai-nilai *term* tersebut dihitung dengan menggunakan *term weighting*.

Diantara metode pembobotan yang dapat digunakan adalah pembobotan TF.IDF (Salton, 1989); pembobotan *inverse class frequency* (ICF) dan variasinya, *inverse class space density frequency* (ICSdF) (Fuji dkk, 2013); serta pembobotan *inverse book frequency* (IBF) (Fauzi, 2013).

Pembobotan TF.IDF mengkombinasikan *term frequency* (TF) yang mengukur kepadatan *term* dalam sebuah dokumen dikalikan dengan *inverse document frequency* (IDF) yang mengukur signifikasi sebuah *term* (kelangkaannya

pada keseluruhan korpus) (Salton, 1989). Sedangkan dengan ICF dan ICSdF, *term* yang sering muncul pada banyak kelas akan memiliki nilai yang kecil. Metode ini terbukti memiliki *precision* dan *recall* yang lebih tinggi daripada TF.IDF (Fuji dkk, 2013). Pembobotan yang diajukan oleh Fauzi (2013) adalah pembobotan *term* berbasis buku untuk perangkingan dokumen bahasa Arab yang dinamakan *inverse book frequency* (IBF) untuk meningkatkan performa perangkingan dokumen yang memiliki hierarki berupa buku-buku yang memiliki banyak halaman. Perhitungan IBF tersebut dikalikan dengan metode sebelumnya sehingga menjadi TF.IDF.ICF.IBF. Metode TF.IDF.ICF.IBF terbukti dapat diaplikasikan pada perangkingan dokumen berbahasa Arab dan memiliki performa yang lebih bagus dibanding metode sebelumnya (Fauzi, 2013).

Hanya saja, selain kesesuaian *query* dengan hasil pencarian, terdapat penilaian subjektif pengguna yang diharapkan turut menjadi faktor penentu dalam perangkingan dokumen. Aspek preferensi ini pun tampak pada pencarian dokumen fiqih. Dokumen fiqih dapat dikelompokkan dalam kelompok mazhab tertentu. Mazhab tersebut sebagai paradikma atau metodologi fiqih yang menjadi salah satu faktor dalam pengambilan keputusan (Sulaiman, 2009). Diantara metodologi fiqih yang terkenal adalah Syafi'iyah, Hanabilah, Hanafiyah, Malikiyah. Nama metodologi fiqih tersebut sesuai dengan nama tokohnya yaitu Imam Syafi'i, Hambali, Hanafi, dan Maliki.

Di masyarakat terdapat fenomena adanya kecenderungan bagi seseorang untuk mengikuti pendapat dari metodologi fiqih tertentu atau mengutamakannya dengan tidak menolak/menyalahkan pendapat metodologi fiqih lain. Oleh sebab itu, maka perlu adanya faktor preferensi disamping relevansi dalam melakukan pencarian. Dalam pencarian dokumen fiqih mazhab sebagai faktor preferensi dalam merankingkan dokumen.

Maka pada penelitian ini diajukan suatu metode pembobotan kata dengan menambahkan faktor preferensi yang digunakan untuk perangkingan dokumen fiqih berbahasa Arab. Untuk meningkatkan bobot preferensi dimodelkan dengan adanya konstanta $\propto$ sebagai nilai pengali yang akan menguatkan atau meningkatkan bobot sesuai preferensi pengguna. Dimana $\propto$ bernilai antara 0 sampai 1. Nilai 0 menghasilkan bobot yang berimbang atau tanpa preferensi yang ditingkatkan.

Sedangkan nilai 1 akan meningkatkan bobot dari preferensi dan menghilangkan bobot yang tidak termasuk preferensi, sehingga dokumen yang termasuk dalam preferensi saja yang akan tampil.

Pembobotan berbasis preferensi ini dinamakan dengan *inverse preference frequency with* ∝ *value* (IPF∝). Metode IPF∝ ini memberikan nilai yang lebih tinggi pada *term* yang menjadi indeks *preference* (yang dipilih atau lebih diutamakan). Pembobotan ini digabungkan dengan pembobotan yang telah ada sebelumnya dengan cara mengkalikannya, sehingga diperoleh metode *term* weighting TF.IDF.IBF.IPF∝. Gabungan metode ini dengan metode pembobotan yang ada sebelumnya menghasilkan pembobotan yang memperhatikan aspek kesesuaian (*relevance*) dan preferensi (*preference*). Harapannya, dengan pembobotan ini akan diperoleh perangkingan dokumen dengan urutan yang tepat dan sesuai dengan kecenderungan yang diharapkan pengguna.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan maka rumusan masalah yang diangkat dalam penelitian ini antara lain:

1. Bagaimana memperoleh nilai preferensi dengan pembobotan *Invers Preference Frequency with* ∝ *value* (IPF∝)?
2. Bagaimana melakukan perangkingan dokumen fiqih berbahasa Arab menggunakan pembobotan TF.IDF.IBF.IPF∝?
3. Bagaimana pengaruh nilai ∝ pada pembobotan TF.IDF.IBF.IPF∝ terhadap perangkingan dokumen fiqih berbahasa Arab?

1.3 Tujuan Penelitian

Tujuan yang akan dicapai dalam pembuatan tesis ini adalah sebagai berikut:

1. Mendapatkan hasil perangkingan dokumen dengan menggunakan metode TF.IDF.IBF.IPF∝ pada dokumen fiqih berbahasa Arab.
2. Memperoleh nilai pengukuran performa dari metode TF.IDF.IBF.IPF∝.

1.4 Manfaat Penelitian

Penelitian ini menghasilkan metode TF.IDF.IBF.IPF$_{\propto}$ yang menggabungkan aspek relevansi dan preferensi untuk melakukan perangkingan pada dokumen fiqih berbahasa Arab.

1.5 Kontribusi Penelitian

Kontribusi penelitian ini yaitu mengajukan metode pembobotan kata berbasis preferensi yang memperhatikan aspek kesesuaian dan pilihan pengguna untuk perangkingan dokumen fiqih berbahasa Arab.

1.6 Batasan Masalah

Pembahasan dalam penelitian ini dibatasi pada beberapa hal berikut ini:

1. Metode pembobotan yang diusulkan diterapkan pada dokumen fiqih berbahasa Arab.
2. Dokumen yang digunakan diambil dari sejumlah halaman di beberapa buku fiqih berbahasa Arab.

BAB II
KAJIAN PUSTAKA

Pada bagian ini akan dipaparkan konsep dasar tentang teori-teori yang dipakai sebagai pegangan dalam melakukan penelitian.

2.1 Mazhab Fiqih

Mazhab (bahasa Arab: مذهب, madzhab) menurut bahasa Arab adalah *isim makan* (kata benda keterangan tempat) dari akar kata *dzahab* (pergi). Sehingga mazhab secara bahasa artinya, "tempat pergi", yaitu jalan (*ath-tharîq*) (Abdullah, 1995; Nahrawi, 1994). Menurut para ulama dan ahli agama Islam, yang dinamakan mazhab adalah metode (manhaj) yang dibentuk setelah melalui pemikiran dan penelitian, kemudian orang yang menjalaninya menjadikannya sebagai pedoman yang jelas batasan-batasannya, bagian-bagiannya, dibangun di atas prinsip-prinsip dan kaidah-kaidah. Sedangkan menurut istilah ushul fiqih, mazhab adalah kumpulan pendapat mujtahid yang berupa hukum-hukum Islam, yang digali dari dalil-dalil syariat yang rinci serta berbagai kaidah (*qawâ'id*) dan landasan (*ushûl*) yang mendasari pendapat tersebut, yang saling terkait satu sama lain sehingga menjadi satu kesatuan yang utuh (Abdullah, 1995; Nahrawi, 1994).

Menurut Muhammad Husain Abdullah (1995), istilah *mazhab* mencakup dua hal: (1) sekumpulan hukum-hukum Islam yang digali seorang imam mujtahid; (2) ushul fiqih yang menjadi jalan (*tharîq*) yang ditempuh mujtahid itu untuk menggali hukum-hukum Islam dari dalil-dalilnya yang rinci. Berbagai mazhab fiqih lahir pada masa keemasan fiqih, yaitu dari abad ke-2 H hingga pertengahan abad ke-4 H dalam rentang waktu 250 tahun di bawah Khilafah Abbasiyah yang berkuasa sejak tahun 132 H. Mazhab yang digunakan secara luas saat ini antara lain mazhab Hanafi, mazhab Maliki, mazhab Syafi'i dan mazhab Hambali.

2.2 *Prepocessing* Teks Berbahasa 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 IR. Elemen dari *indexing* adalah fitur (*keyword*, *term*, atau 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 atau topik yang dikandung oleh dokumen. Indeks akan membedakan antara suatu dokumen dengan dokumen lain yang berada di dalam 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 dapat menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pemrosesan (Machnik, 2004).

Untuk dokumen teks, setiap kata di dalam dokumen dapat dianggap sebagai *term*, setelah kata tersebut mengalami *preprocessing*. *Preprocessing* berkaitan erat dengan penerapan proses *indexing* pada suatu sistem IR. Suatu skema *preprocessing* biasanya tidak hanya berurusan dengan bagaimana merepresentasikan suatu dokumen teks, tapi juga bagaimana menghasilkan solusi yang efektif dan efisien. Pembuatan indeks melibatkan konsep *linguistic processing* yaitu pemrosesan *term* sesuai dengan struktur bahasa yang bertujuan mengekstrak *term-term* penting dari dokumen yang direpresentasikan sebagai *bag of words*. Urutan langkah pembangunan indeks dengan implementasi *linguistic processing* dengan menggunakan sebuah dokumen teks Arab adalah sebagai berikut (Mesleh dan Kanaan, 2008; Mesleh, 2007):

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

"Katakan, 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. Pemisahan rangkaian kata *(tokenization).*

   Tahap ini berfungsi memisahkan deretan kata di dalam kalimat, paragraf, maupun halaman menjadi *token* atau potongan kata tunggal atau *termmed*

word. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti digit, angka, tanda hubung, dan tanda baca. Maka dari dokumen tersebut akan menghasilkan:

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

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

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

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

2. *Normalization and filtration*.

Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. Selain itu, juga dilakukan penghilangan karakter-karakter yang tidak termasuk dalam huruf *hijaiyah* **(…** ,ٌت ,ٌب ,ا**(,** kemudian dilakukan penghapusan *diacritic (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 -

احد - اوكفوا -له نكي ملد ولي ملد لي

Setelah dilakukan proses *normalization and 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 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 ,بدًا ,آخر), kata ganti, artikel dan preposisi dihapus. 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** -

دلي –**لم** -ولد -**يكن** -**له** -**احد** - كفوا

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

قل الصمد دلي ولد كفوا

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

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 tahap ini setiap kata yang telah dihasilkan dari serangkaian proses sebelumnya kemudian dirubah menjadi kata dasar, dan akan menghasilkan *output* sebagai berikut :

قال -صمد -ولد -ولد كفا

Setelah proses *stemming* selesai dilakukan, maka didapat kata-kata dalam bentuk dasar seperti di atas.

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 yang sering digunakan dalam *information retrieval* teks Arab (Larkey, 2002). *Stemmer* inilah yang digunakan pada *library Lucene* apabila menggunakan *class ArabicAnalyzer*.

2.3 Perangkingan Dokumen

Perangkingan dokumen menggunakan representasi *vector space model* dari kumpulan dataset. Dokumen dalam *vector space model* di representasikan dalam matriks yang berisi bobot kata pada dokumen. Bobot tersebut menyatakan kepentingan/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 di peroleh bobot kata pada dokumen yang digunakan sebagai representasi vektor. Dari representasi bobot tersebut dapat dihitung nilai kemiripan suatu dokumen dengan *query*. Nilai kemiripan ini biasa dihitung dengan rumusan *cosine similarity*, perhitungan tingkat kemiripan ini dibuat dengan berdasar pada besar sudut kosinus antara dua vektor, dalam hal ini adalah vektor dokumen. Hasil dari perhitungan jarak cosinus antara tian dokumen terhadap *query* ini lah yang digunakan untuk merangkingkan dokumen.

2.3.1 Pembobotan kata

Dibawah ini terdapat beberapa metode pembobotan :

- *Term Frequency (TF)*

  Term frequency merupakan metode yang paling sederhana dalam membobotkan kata. Setiap kata diasumsikan memiliki kepentingan yang proporsional terhadap jumlah kemunculan kata pada dokumen. Bobot dari kata *t* pada dokumen *d* didefinisikan sebagaimana **Persamaan 2.1** dimana $f(t_i, d_j)$ adalah frekuensi kemunculan *term* t_i pada dokumen d_j.

$$TF(t_i, d_j) = f(t_i, d_j), \tag{2.1}$$

- *Inverse Document Frequency (IDF)*

  Bila *term frequency* memperhatiakan 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 memilki proporsi yang berkebalikan dengan jumlah dokumen yang mengandung *term*. Faktor *IDF* dari *term t* yaitu sebagaimana **Persamaan**

2.2. Pada persamaan tersebut *N* adalah jumlah seluruh dokumen dan *df(t)* adalah jumlah dokumen yang mengandung *term t*.

$$IDF(t_i) = 1 + log(N/df(t_i)), \tag{2.2}$$

- *TFIDF*

  Perkalian antara *term frequency* dan *IDF* dapat menghasilkan performansi yang lebih baik. Kombinasi bobot dari *term t* pada dokumen *d* sebagaimana **Persamaan 2.3** dengan $TF(t_i, d_j)$ adalah frekuensi *term ke-I* dan $IDF(t_i)$ adalah *inverse* kemunculan *term ke-i* pada dokumen ke-j.

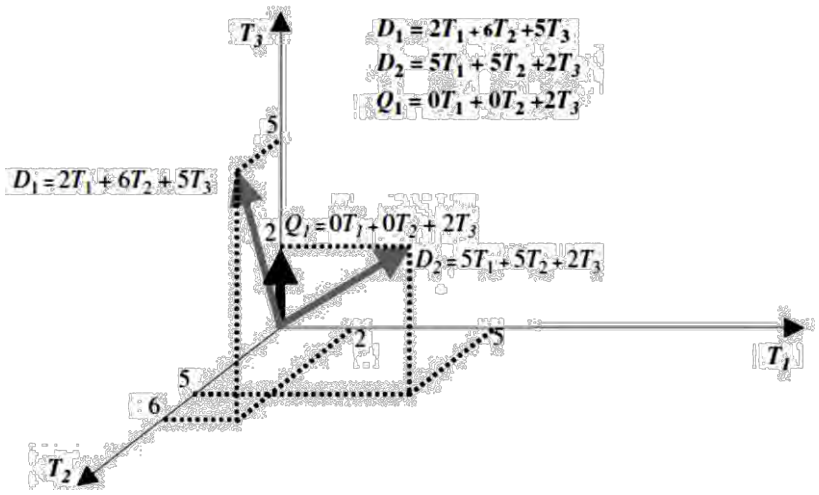
$$TFIDF(t_i, d_j) = TF(t_i, d_j) \times IDF(t_i), \tag{2.3}$$

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

2.3.2 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* index. *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 $d_j = (w_1, w_2, ..., w_{tj})$ dan terdapat n dokumen di dalam koleksi, yaitu j = 1, 2, ..., n.



Gambar 2.1 Model ruang vektor

Dalam model ruang vektor, koleksi dokumen direpresentasikan 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*.

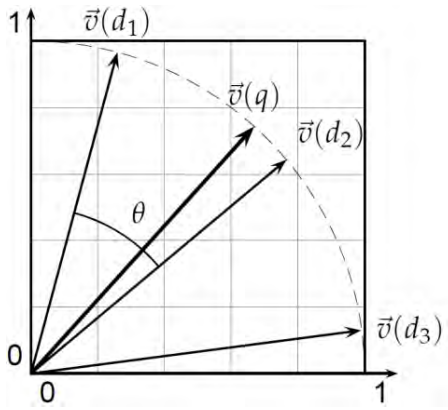
$$\begin{bmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \dots & \dots & \dots & \dots & \dots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{bmatrix}$$

Gambar 2.2 Matriks *term-document*

Keberhasilan dari model VSM ini ditentukan oleh skema pembobotan terhadap suatu *term* baik untuk cakupan lokal maupun global, dan faktor normalisasi (Polettini, dkk, 2004). Pembobotan lokal hanya berpedoman pada frekuensi munculnya *term* dalam suatu dokumen dan tidak melihat frekuensi kemunculan *term* tersebut di dalam dokumen lainnya.

2.3.3 Cosine similarity

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



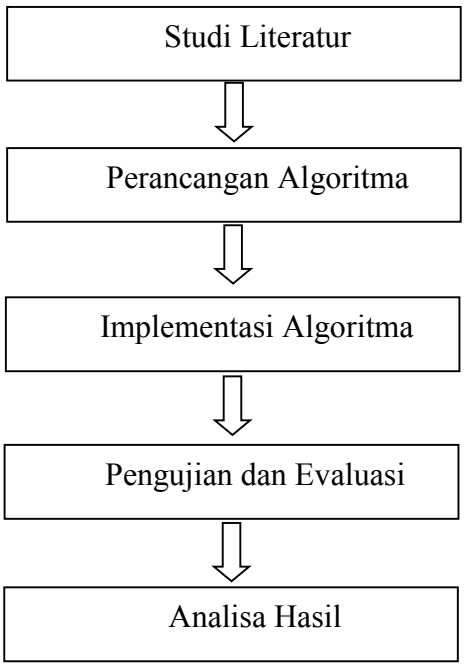
Gambar 2.3 Representasi Perumusan *Cosine Similarity*

Karena berdasarkan kosinus sudut antara 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 2.4** berikut (Garcia, 2006). Pada persamaan tersebut $cos(q,d_j)$ adalah notasi dari nilai kosinus antara *query* dan dokumen *j*, $TFIDF(t_k,q)$ dan $TFIDF(t_k,d_j)$ masing-masing menunjukkan pembobotan *TFIDF* kata t_k pada *query* dan pembobotan *TFIDF* kata t_k pada dokumen *j*, sedangkan $|TFIDFq|$ dan $|TFIDFd_j|$ maisng-masing merupakan panjang dari vektor *query* q dan panjang dari vektor dokumen *j*.

$$\cos(q,d_j) = \frac{\sum_{t_k}\left[TFIDF(t_k,q)\right]\bullet\left[TFIDF(t_k,d_j)\right]}{\sqrt{\sum|TFIDFq|^2}\bullet\sqrt{\sum|TFIDFd_j|^2}}, \tag{2.4}$$

BAB III
METODOLOGI PENELITIAN

Tahapan-tahapan yang dilalui pada penelitian ini meliputi (1) Studi Literatur, (2) Perancangan algoritma, (3) Implementasi algoritma, (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

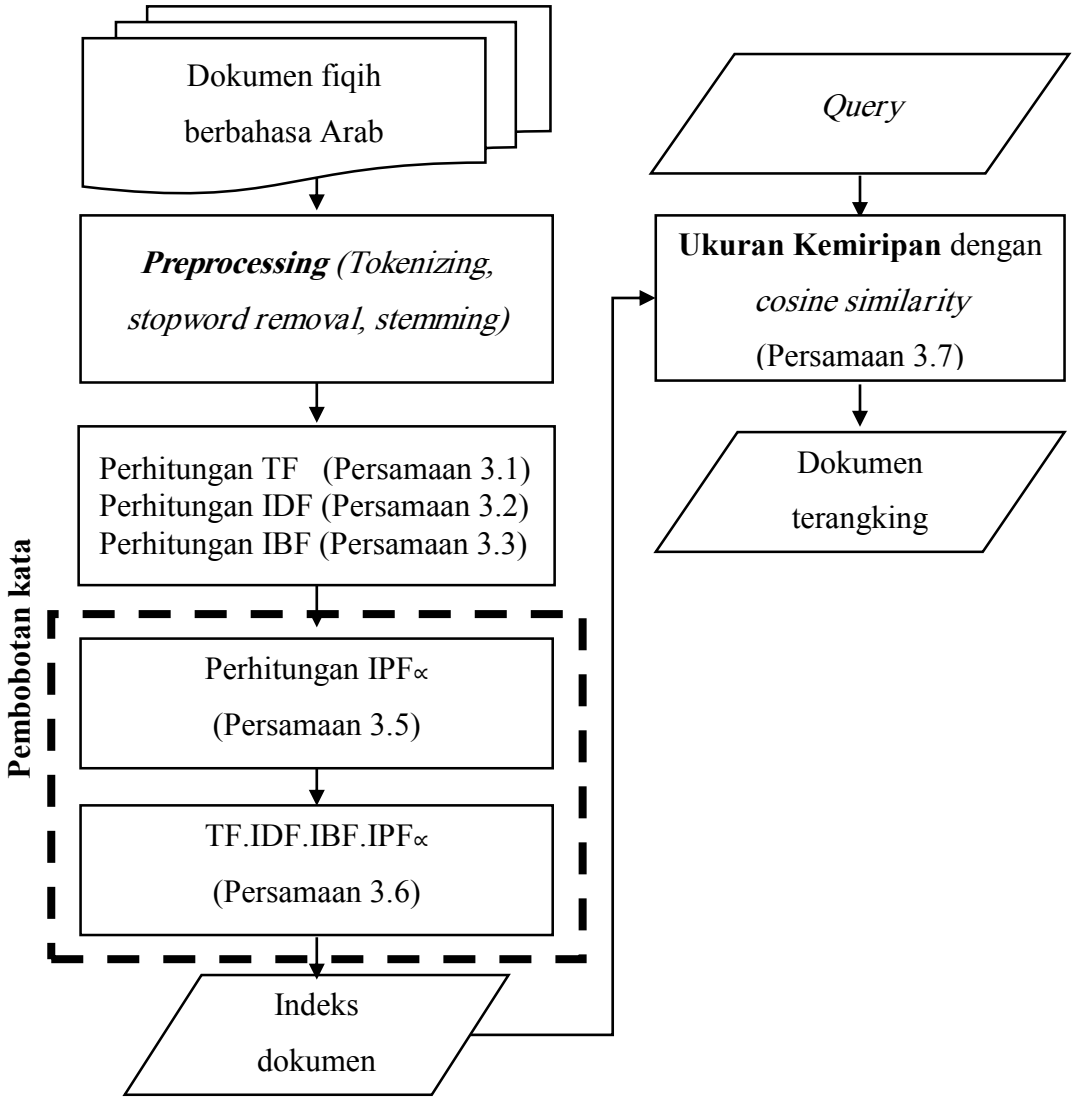
Studi literatur dilakukan untuk mendapatkan informasi yang berkaitan dengan lingkup pembahasan dalam penelitian, perkembangan keilmuan terkait, serta metode yang telah ada sebelumnya. Studi literatur yang dilakukan diharapkan dapat memberikan data, informasi, dan fakta mengenai perangkingan dokumen berbahasa 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.IBF
3. Metode pengukuran kemiripan dokumen menggunakan *cosine similarity*.
4. Preferensi pencarian
5. Evaluasi hasil perangkingan dokumen dengan perhitungan *recall*, *precision*, dan *f-measure*.

3.2 Perancangan Algoritma

Alur proses dalam penelitian yang dilakukan terdiri dari beberapa tahapan sebagaimana yang tertera pada **Gambar** 3.2. Dokumen fiqih berbahasa Arab sebagai dataset melalui tahap awal yaitu *preprocessing*. Pada tahap *preprocessing* dokumen dipotong menjadi potongan-potongan kata (*tokenizing*), dihilangkan kata yang termasuk daftar *stoplist* (*stopword removal*), dan dirubah menjadi akar kata (*stemming*). Setelah itu, dilakukan pembobotan kata terhadap *term* hasil tahap *preprocessing*, mulai dari perhitungan nilai TF (*Term Frequency*) menggunakan **Persamaan 3.1**, IDF (*Invers Document Frequency*) menggunakan **Persamaan 3.2**, IBF (*Invers Book Frequency*) menggunakan **Persamaan 3.3**, kemudian perhitungan IPF$_{\propto}$ (*Invers Preference Frequency with* $\propto$ *value*) menggunakan **Persamaan 3.5**. Variabel $\propto$ sebagai penguat yang menunjukkan tingkat preferensi dari pengguna. Nilai $\propto$ antara 0 hingga 1, dimana 0 menunjukkan keberimbangan preferensi sedangkan 1 menunjukkan preferensi penuh.

Masing-masing bobot TF, IDF, IBF, dan IPF$_{\propto}$ dikalikan sehingga menjadi pembobotan TF.IDF.IBF.IPF$_{\propto}$ (**Persamaan 3.6**). Bobot tiap *term* dimodelkan dalam *vector space model* hingga dilakukan perhitungan jarak kosinus antar sumbu bobot *term* menggunakan ukuran kemiripan *cosine similarity* dengan **Persamaan 3.7**. Perhitungan *cosine similarity* memberikan hasil akhir nilai 0 hingga 1. Nilai 0 menunjukkan bahwa dokumen tersebut tidak sesuai dengan *query* dan semakin mendekati 1 menunjukkan bahwa dokumen tersebut sesuai dengan *query*. Dari nilai kemiripan tersebut dokumen dapat dirangkingkan mulai dari nilai tertinggi, yang paling sesuai hingga terendah, yang paling tidak sesuai.



Gambar 3.2 Diagram Rancangan Sistem

3.2.1 Dataset

Data yang digunakan pada penelitian ini adalah dokumen fiqih berbahasa Arab, dimana dokumen merupakan tiap halaman dari buku. Buku fiqih tersebut adalah buku yang telah terkategorikan dalam empat metodologi fiqih. Data diambil dari *e-book* Bahasa Arab (*http://shamela.ws/*) dengan rincian (**Tabel** 3.1):

- 4 metodologi fiqih: Hanafiyah, Malikiyah, Syafi'iyah, Hanabilah
- Tiap metodologi fiqih diambil 3 buku sehingga total buku yang digunakan adalah 12 buku fiqih
- Dokumen yang digunakan adalah 10 dokumen dari tiap buku sehingga total dokumen adalah 120 dokumen

Tabel 3.1 Dataset dokumen fiqih

No	Metodologi Fiqih	Judul Buku	
1	Hanafiyah	المبسوط	*al-Mabsuth*
		الجامع الصغير	*al-jami' ash-shoghir*
		مراقي الفلاح شرح متن نور الإيضاح	*maroqiy al-falaah syarah matan nuur al-iydhooh*
2	Malikiyah	المدونة	*al-mudwinah*
		متن الرسالة	*matan ar-risalah*
		التلقين في الفقه المالكي	*at-talqiin fiy al-fiqhi al-malikiy*
3	Syafi'iyah	الأم	*al-Umm*
		مختصر المزني	*Mukhtashar al-muzani*
		الإقناع في الفقه الشافعي	*al-iqnaa'u fiy al-fiqhi asy-syafi'I*
4	Hanabilah	المغني لابن قدامة	*al-Mughni li-ibni Qudamah*
		الإقناع في فقه الإمام أحمد بن حنبل	*al-iqnaa'u fiy al-fiqhi al-imam ahmad bin hambal*
		الشرح الكبير على متن المقنع	*asy-syarah al-kabir 'ala matan al-maqna'i*
Total	**4 metodologi fiqih**	**12 Buku**	

3.2.2 *Preprocessing*

Masing-masing dokumen melalui tahap *preprocessing*. Implementasi *preprocessing* ini terdiri dari beberapa tahapan, diantaranya adalah tokenisasi, *filtering, stemming*, dan *stopword removal*. Tokenisasi dilakukan untuk memecah keseluruhan isi dokumen menjadi suku kata tunggal. Sedangkan pada tahapan *filtering* dilakukan pembuangan harokat-harokat bahasa Arab. Penghilangan *stopword* dilakukan untuk menghilangkan kata-kata yang sering muncul dalam dokumen tetapi tidak mempunyai nilai yang berarti pada sebuah dokumen. Daftar 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/*).

3.2.3 Pembobotan

Tahap setelah *preprocessing* adalah *term weighting*. Pembobotan kata dilakukan dengan menghitung TF (*Term Frequency*), IDF (*Inverse Document Frequency*), dan IBF (*Inverse Book Frequency*) dari masing-masing *term* pada seluruh dokumen. Kemudian perhitungan IPF (*Inverse Preference Frequency*) juga dilakukan terhadap masing-masing *term* pada seluruh dokumen. Bobot IPF dikalikan dengan nilai $\propto$ untuk *term* yang termasuk *query* pengguna sebagaimana **Persamaan 3.5** sehingga diperoleh bobot IPF$_\propto$ (*Invers Preference Frequency with* $\propto$*value*). Rumusan TF.IDF.IBF.IPF$_\propto$ menggunakan **Persamaan 3.6**.

Berikut ini adalah rumus perhitungan tahap pembobotan:

- *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 **Persamaan 3.1** dengan $f(d_j,t_i)$ adalah frekuensi kemunculan *term t* ke-*i* pada dokumen *d* ke-*j*.

$$W_{TF}(t_i, d_j) = f(d_j, t_i), \tag{3.1}$$

- *Inverse Document Frequency (IDF)*

  Bila *term frequency* memperhatiakan 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* menggunakan **Persamaan 3.2.** Pada persamaan tersebut, *D* merupakan jumlah seluruh dokumen dan $d(t_i)$ adalah jumlah dokumen yang mengandung *term t* ke-*i*.

$$W_{IDF}(t_i, d_j) = 1 + log\left(\frac{D}{d_{(t_i)}}\right) \quad (3.2)$$

- *Inverse Book Frequency (IBF)*

  IBF memperhatikan kemunculan *term* pada kumpulan kitab/buku. *Term* yang jarang muncul pada banyak buku adalah *term* yang sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah buku yang mengandung *term*. Perhitungan IBF ini dapat diadopsi langsung dari IDF. Perhitungan bobot *IBF* dari *term t* berdasarkan **Persamaan 3.3** dengan *B* adalah jumlah seluruh buku dan $b(t_i)$ adalah jumlah buku yang memuat minimal satu *term* t_i.

$$W_{IBF}(t_i, b_k) = 1 + log\left(\frac{B}{b_{(t_i)}}\right) \quad (3.3)$$

- *Inverse Preference Frequency with* $\propto$ *value* ($IPF_{\propto}$)

  Sebelum perhitungan bobot $IPF_{\propto}$, terlebih dahulu dilakukan pembobotan IPF (*Invers Preference Frequency*) dari masing-masing *term*. Jika IBF memperhatikan kemunculan *term* pada kumpulan buku, maka IPF memperhatikan kemunculan *term* pada tiap kelompok yang menjadi pilihan preferensi pengguna. Dalam penelitian ini kelompok pilihan preferensi pengguna adalah empat metodologi fiqih. *Term* yang jarang muncul pada keempat metodologi fiqih adalah *term* yang sangat bernilai. Kepentingan tiap *term* diasumsikan memiliki proporsi yang berkebalikan dengan jumlah metodologi fiqih yang mengandung *term*. Seperti halnya IBF, perhitungan IPF ini dapat diadopsi langsung dari IDF. Perhitungan bobot *IPF* dari *term t* dengan **Persamaan 3.4.** Sedangkan bobot $IPF_{\propto}$ diperoleh dari perkalian bobot IPF dengan nilai $\propto$ sesuai **Persamaan 3.5.** Pada kedua persamaan tersebut *P* adalah jumlah seluruh metodologi fiqih, $p(t_i)$ adalah jumlah metodologi fiqih yang memuat minimal satu *term* t_i, $\propto$ adalah variabel bernilai antara 0 hingga 1, UP merupakan preferensi pengguna, dan Q adalah *query*.

$$W_{IPF}(t_i, p_l) = 1 + log\left(\frac{P}{p_{(t_i)}}\right) \tag{3.4}$$

$$W_{IPF_\propto}(t_i, d_j, p_l) = \begin{cases} \left(1 + log\left(\frac{P}{p_{(t_i)}}\right)\right) \times \left(\frac{\propto}{2} + 0.5\right), & d_j \in UP, t_i \in Q \\ \left(1 + log\left(\frac{P}{p_{(t_i)}}\right)\right) \times \left(1 - \left(\frac{\propto}{2} + 0.5\right)\right), & d_j \notin UP, t_i \in Q \\ \left(1 + log\left(\frac{P}{p_{(t_i)}}\right)\right), & t_i \notin Q \end{cases} \tag{3.5}$$

Ide dari pembobotan IPF∝ adalah meningkatkan bobot *term* yang berada pada dokumen yang termasuk preferensi pengguna dan menurunkan bobot *term* yang tidak berada pada dokumen yang termasuk preferensi pengguna. Sehingga *term* yang berada pada dokumen yang termasuk preferensi pengguna memiliki nilai signifikan lebih tinggi dibandingkan yang lain. Selain itu, peningkatan bobot *term* yang berada pada dokumen sesuai dengan preferensi pengguna dan merupakan term yang sama dengan *query* mampu mendekatkan posisi vektor dokumen yang memuat *term* tersebut dengan vektor *query* pada representasi *Vector Space Model*. Sebaliknya, vektor dokumen yang tidak sesuai dengan preferensi pengguna akan semakin jauh dengan vektor *query*. Dengan demikian, maka sistem mampu menempatkan dokumen yang termasuk preferensi pengguna pada urutan yang lebih atas pada perangkingan dokumen daripada dokumen selain preferensi pengguna. Penguat bobot tersebut dimodelkan dengan adanya variabel $\propto$ yang bernilai antara 0 sampai dengan 1. Nilai 0 menunjukkan tidak ada yang di utamakan, sedangkan 1 menunjukkan hanya mengutamakan preferensi pengguna. Ketika dokumen termasuk dalam kategori preferensi pengguna (UP) dan term termasuk *query* (Q) maka bobot IPF dikalikan dengan $\frac{\propto}{2} + 0.5$. Sehingga semakin besar nilai $\propto$ maka term akan diberi bobot yang lebih tinggi. Sebaliknya, ketika dokumen tidak termasuk dalam kategori preferensi pengguna dan term termasuk *query* maka bobot IPF dikalikan dengan $1 - \left(\frac{\propto}{2} + 0.5\right)$. Sehingga semakin besar nilai $\propto$ maka term akan

diberi bobot lebih rendah. Sedangkan jika term tidak termasuk dalam *query* maka term tersebut hanya diberi bobot IPF tanpa nilai $\propto$ sebagai pengali.

Gabungan dari masing-masing pembobotan tersebut memberikan variasi pembobotan kata. Dalam penelitian ini, menggunakan pembobotan TF.IDF.IBF.IPF$_\propto$ yang merupakan perkalian dari bobot TF, IDF, IBF, dan IPF$_\propto$ dari suatu *term* terhadap dokumen, buku, dan kelompok preferensi tertentu. Hal ini sebagaimana yang didefinisikan pada **Persamaan 3.6**.

$$W_{TF.IDF.IBF.IPF_\propto}(t_i, d_j, b_k, p_l) = W_{TF.IDF}(t_i, d_j) \times W_{IBF}(t_i, b_k) \times W_{IPF_\propto}(t_i, d_j, p_l) \quad (3.6)$$

3.2.4 Pembentukan Vector Space model

Vector space model merepresentasikan dokumen atau *query* sebagai sebuah vektor dalam sebuah ruang *term*. 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 vectornya 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 dokumen 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* dalam *Vector Space Model* ini diambil dari seluruh *term* unik pada keseluruhan dokumen.

3.2.5 Ukuran kemiripan dengan *cosine similarity*

Dari bobot *term* yang *term*uat dalam vector space model dilakukan perhitungan kemiripan menggunakan cosine similarity sesuai dengan **Persamaan 3.7**. Hasil dari perhitungan ini memberikan nilai kemiripan antara rentang 0 sampai 1. Nilai yang semakin mendekati 1 menunjukkan tingkat kemiripan yang tinggi. Sehingga dari hasil perhitungan ini jika diurutkan akan menghasilkan dokumen yang berurutan (terangking).

$$\cos(q, d_j) = \frac{\sum_{t_i}[w(t_i, q)] \cdot [w(t_i, d_j]}{\sqrt{\sum |w(q)|^2} \cdot \sqrt{\sum \left|w(d_j)\right|^2}} \tag{3.7}$$

Pada Persamaan 3.7 tersebut *cos(q,dj)* adalah nilai kosinus antara *query* dan dokumen *j*, $w(t_i, q)$ merupakan bobot TF.IDF.IBF.IPF dari term t_i pada *query*, dan $w\left(t_i, d_j\right)$ merupakan bobot TF.IDF.IBF.IPF$\propto$ berdasarkan **Persamaan 3.6** untuk setiap term t_i pada dokumen *j* berdasarkan sebaran term pada koleksi buku dan kelompok preferensi. Sedangkan $|w(q)|$ dan $|w(d_j)|$ masing-masing merupakan panjang dari vektor *query* q dan panjang dari vektor dokumen j. Sebagai contoh panjang vektor dokumen yaitu $||d_j||^2$ = (TF.IDF.IBF.IPF$\propto t1^2$ + TF.IDF.IBF.IPF$\propto t_2^2$ + TF.IDF.IBF.IPF$\propto t_3^2$ +...+ TF.IDF.IBF.IPF$\propto t_i^2$)$^{1/2}$, dimana TF.IDF.IBF.IPF$\propto t_i$ adalah bobot kata ke-t_i pada vektor dokumen d_j.

3.2.6 Contoh Perhitungan Manual

Contoh Kasus

Sebagai contoh, terdapat enam dokumen dengan kode mulai dari D1 hingga D6. Keenam dokumen tersebut adalah dokumen yang relevan, yaitu dengan topik tentang hukum membaca *al-fatihah* bagi *ma'mum*. Namun keenam dokumen tersebut tersebar pada buku dan kelompok preferensi yang berbeda. Yaitu, dokumen tersebut tersebar dari lima buku dengan kode buku mulai dari B1 hingga B5. Dan tiap buku tersebut dari empat metodologi fiqih yang menjadi pilihan preferensi pengguna. Pilihan preferensi pengguna ini ditulis dengan kode mulai dari P1 hingga P4. Isi keenam dokumen tersebut dengan keterangan buku dan preferensi

dapat dilihat pada **Tabel 3.2**. Untuk dokumen D1 dan D2 sama-sama dimuat pada buku B1 yang termasuk pada pilihan preferensi P1. Dokumen D4 dimuat pada buku B3 dan dokumen D5 dimuat dalam buku B4, kedua buku tersebut masuk dalam kategori preferensi P3. Sedangkan dokumen yang lain masing-masing dari satu buku dan satu preferensi yang berbeda.

Tabel 3.2 Contoh Dokumen untuk Perhitungan Manual

Dokumen	Buku	Preferensi	Isi Dokumen
D1	B1	P1	وجب على المأموم أن يقرأ الفاتحة خلف الإمام سواء كان الصلاة سرية أم جهرية wajib bagi makmum untuk membaca al fatihah setelah imam baik untuk sholat siriyyah maupun jahriyah
D2	B1	P1	وجب قراءة الفاتحة على المأموم في كل الركعات من الصلاة سرية والجهرية wajib membaca al fatihah bagi makmum di setiap rokaat pada sholat siriyah dan jahriyah
D3	B2	P2	وجب قراءة الفاتحة خلف الإمام wajib membaca al fatihah setelah imam
D4	B3	P3	الصلاة إذا كانت سرية يقرأ الفاتحة خلف الإمام, ولا يقرأ في الجهرية Sholat ketika siriyyah maka membaca al fatihah setelah imam, dan tidak membaca dalam jahriyah
D5	B4	P3	قراءة الفاتحة قبل المأموم هي مندوب في السرية وكروه في الصلاة الجهرية membaca al fatihah bagi makmum adalah mandub untuk sholat siriyah dan makruh untuk sholat jahriah
D6	B5	P4	ان المأموم لا يقرأ الفاتحة خلف الإمام , لا في السرية ولا في الجهرية sesungguhnya makmum tidak membaca al fatihah setelah imam, tidak di siriyyah dan tidak di jahriyah

Dilakukan pencarian terhadap keenam dokumen tersebut dengan keterangan:

Query : المأموم يقرأ الفاتحة في السرية وفي الجهرية

Preferensi pengguna : P1

Nilai ∝ : 0.6

Tahapan Perangkingan Dokumen

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

مامإ	هيوج	فخ	هيس	ءاوس	هال	هحنن
1	2	3	4	5	6	7

ارق	مومأم	ةبسن	بودنم	هوركم	ءاعكر	بجو
8	9	10	11	12	13	14

Selanjutnya dilakukan perhitungan bobot TF, IDF, IBF, dan IPF untuk setiap *term* dengan **Persamaan 3.1**, **Persamaan 3.2**, **Persamaan 3.3** dan **Persamaan 3.4.** Hasil perhitungan bobot TF, IDF, dan IBF tiap *term* tersebut dapat dilihat pada **Tabel 3.3.** Pembobotan IPF dan IPF$_{\propto}$ yang diajukan pada penelitian ini dapat dilihat pada **Tabel 3.4**. Variasi dari pembobotan kata ditunjukkan pada **Tabel 3.5**, **Tabel 3.6**, dan **Tabel 3.7**. **Tabel 3.5** menunjukkan pembobotan TF.IDF yang diperoleh dengan mengalikan bobot TF dengan IDF. **Tabel 3.6** menunjukkan pembobotan TF.IDF.IBF yang diperoleh dari perkalian bobot TF.IDF dengan IBF. **Tabel 3.7** menunjukkan pembobotan yang diusulkan yaitu dengan metode TF.IDF.IBF.IPF$_{\propto}$ dengan kondisi dimisalkan yang menjadi preferensi pengguna adalah P1 dengan nilai $\propto = 0.6$.

Pada contoh kasus ini, dokumen yang termasuk dalam kategori preferensi pengguna (UP) adalah D1 dan D2. Oleh karena itu, berdasarkan rumusan yang diajukan (**Persamaan 3.6**), untuk *term* pada D1 dan D2 dan *term* tersebut termasuk dalam query (Q) maka bobot IPF dikalikan dengan $\frac{\propto}{2} + 0.5$. Sebaliknya, untuk dokumen D3 hingga D6 yang tidak termasuk dalam kategori preferensi pengguna maka bobot IPF dari *term* pada dokumen tersebut yang termasuk *query* dikalikan dengan $1 - \left(\frac{\propto}{2} + 0.5\right)$. Sedangkan jika term tidak termasuk dalam *query* maka term tersebut hanya diberi bobot IPF tanpa nilai $\propto$ sebagai pengali.

Tabel 3.3 Perhitungan bobot TF, IDF, dan IBF

No	Term	TF						$d(t_i)$	IDF $1+\log(D/d(t_i))$						$b(t_i)$	IBF $1+\log(B/b(t_i))$
		D1	D2	D3	D4	D5	D6			B1	B2	B3	B4	B5		
1	امام	1		1	1		1	4	1.176	1	1	1		1	4	1.097
2	ج ه ي ه	1	1		1	1	1	5	1.079	2		1	1	1	4	1.097
3	خ ل ف	1		1	1		1	4	1.176	1	1	1		1	4	1.097
4	س ي ه	1	1		1	1	1	5	1.079	2		1	1	1	4	1.097
5	س و ا ء	1						1	1.778	1					1	1.699
6	ا ل ه	1	1		1	2		4	1.176	2		1	1		3	1.222
7	ف ل ت ح ه	1	1	1	1	1	1	6	1.000	2	1	1	1	1	5	1.000
8	ق ر ا	1	1	1	2	1	1	6	1.000	2	1	1	1	1	5	1.000
9	م أ م و م	1	1			1	1	4	1.176	2			1	1	3	1.222
10	ن س ب ة					1		1	1.778				1		1	1.699
11	م ن د و ب					1		1	1.778				1		1	1.699
12	ه ك ر و					1		1	1.778				1		1	1.699
13	ك ر ع ا ت		1					1	1.778	1					1	1.699
14	و ج ب	1	1	1				3	1.301	2	1				2	1.398

Tabel 3.4 Perhitungan bobot IPF dan $IPF_{\propto}$ dengan UP=P1 dan $\propto = 0.6$

No	Term					$p(t_i)$	IPF $1+\log(P/p(t_i))$	$IPF_{\propto}$					
		P1	P2	P3	P4			**D1**	**D2**	D3	D4	D5	D6
1	امام	1	1	1	1	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	**ج ه ي ه**	2		2	1	3	1.125	**0.900**	**0.900**	0.225	0.225	0.225	0.225
3	خ ل ف	1	1	1	1	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000
4	**س ي ه**	2		2	1	3	1.125	**0.900**	**0.900**	0.225	0.225	0.225	0.225
5	س و ا ء	1				1	1.602	1.602	1.602	1.602	1.602	1.602	1.602
6	ا ل ه	2		2		2	1.301	1.301	1.301	1.301	1.301	1.301	1.301
7	**ف ل ت ح ه**	2	1	2	1	4	1.000	**0.800**	**0.800**	0.200	0.200	0.200	0.200
8	**ق ر ا**	2	1	2	1	4	1.000	**0.800**	**1.000**	0.200	0.200	0.200	0.200
9	**م أ م و م**	2		1	1	3	1.125	**0.900**	**0.900**	0.225	0.225	0.225	0.225
10	ن س ب ة			1		1	1.602	1.602	1.602	1.602	1.602	1.602	1.602
11	م ن د و ب			1		1	1.602	1.602	1.602	1.602	1.602	1.602	1.602
12	ه ك ر و			1		1	1.602	1.602	1.602	1.602	1.602	1.602	1.602
13	ك ر ع ا ت	1				1	1.602	1.602	1.602	1.602	1.602	1.602	1.602
14	و ج ب	2	1			2	1.301	1.301	1.301	1.301	1.301	1.301	1.301

Pada **Tabel 3.4** dapat dilihat yang termasuk term query adalah term nomor 2, 4, 7, 8, dan 9, yaitu ج ه ي ه, س ي ه ف ل ت ح ه ق ر ا dan م أ م و م. Bobot IPF$\propto$ untuk kelima term tersebut pada dokumen D1 dan D2 yaitu $IPF \times \left(\frac{0.6}{2} + 0.5\right) = IPF \times 0.8$, sehingga bobot term nomor 2 pada dokumen D1 dan D2 adalah $1.125 \times 0.8 = 0.9$, begitupun seterusnya. Sebaliknya, bobot IPF$\propto$ untuk kelima term tersebut pada selain dokumen D1 dan D2 yaitu $IPF \times \left(1 - \left(\frac{0.6}{2} + 0.5\right)\right) = IPF \times (1 - 0.8) = IPF \times 0.2$, sehingga bobot term nomor 2 pada dokumen D3 hingga D6 adalah

$1.125 \times 0.2 = 0.225$, begitupun seterusnya. Sedangkan untuk sembilan term yang lainnya yang tidak termasuk term query hanya diberi bobot IPF dikalikan dengan nilai $\propto$ baik ketika dokumen tersebut termasuk preferensi pengguna ataupun tidak, sehingga bobot term nomor 1 pada dokumen D1 hingga D6 bernilai sama dengan bobot IPF term tersebut, yaitu 1, begitupun seterusnya. Dari sini tampak bahwa term pada dokumen yang termasuk pada preferensi pengguna memiliki bobot yang lebih tinggi dibandingkan dengan term pada dokumen yang bukan termasuk preferensi pengguna.

Tabel 3.5 Pembobotan TF.IDF

No	Term	TF						IDF	TF.IDF					
		D1	D2	D3	D4	D5	D6		D1	D2	D3	D4	D5	D6
1	امام	1		1	1		1	1.176	1.176	0.000	1.176	1.176	0.000	1.176
2	هيرج	1	1		1	1	1	1.079	1.079	1.079	0.000	1.079	1.079	1.079
3	خلف	1		1	1		1	1.176	1.176	0.000	1.176	1.176	0.000	1.176
4	هيس	1	1		1	1	1	1.079	1.079	1.079	0.000	1.079	1.079	1.079
5	سواء	1						1.778	1.778	0.000	0.000	0.000	0.000	0.000
6	اله	1	1		1	2		1.176	1.176	1.176	0.000	1.176	2.352	0.000
7	فلتحه	1	1	1	1	1	1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
8	قرا	1	1	1	2	1	1	1.000	1.000	1.000	1.000	2.000	1.000	1.000
9	مأموم	1	1			1	1	1.176	1.176	1.176	0.000	0.000	1.176	1.176
10	نسبة					1		1.778	0.000	0.000	0.000	0.000	1.778	0.000
11	منذوب					1		1.778	0.000	0.000	0.000	0.000	1.778	0.000
12	هوكم					1		1.778	0.000	0.000	0.000	0.000	1.778	0.000
13	تاعكر		1					1.778	0.000	1.778	0.000	0.000	0.000	0.000
14	وجب	1	1	1				1.301	1.301	1.301	1.301	0.000	0.000	0.000

Tabel 3.6 Pembobotan TF.IDF.IBF

No	Term	TF						IDF	IBF	TF.IDF.IBF					
		D1	D2	D3	D4	D5	D6			D1	D2	D3	D4	D5	D6
1	امام	1		1	1		1	1.176	1.097	1.290	0.000	1.290	1.290	0.000	1.290
2	هيرج	1	1		1	1	1	1.079	1.097	1.184	1.184	0.000	1.184	1.184	1.184
3	خلف	1		1	1		1	1.176	1.097	1.290	0.000	1.290	1.290	0.000	1.290
4	هيس	1	1		1	1	1	1.079	1.097	1.184	1.184	0.000	1.184	1.184	1.184
5	سواء	1						1.778	1.699	3.021	0.000	0.000	0.000	0.000	0.000
6	اله	1	1		1	2		1.176	1.222	1.437	1.437	0.000	1.437	2.874	0.000
7	فلتحه	1	1	1	1	1	1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
8	قرا	1	1	1	2	1	1	1.000	1.000	1.000	1.000	1.000	2.000	1.000	1.000
9	مأموم	1	1			1	1	1.176	1.222	1.437	1.437	0.000	0.000	1.437	1.437
10	نسبة					1		1.778	1.699	0.000	0.000	0.000	0.000	3.021	0.000
11	منذوب					1		1.778	1.699	0.000	0.000	0.000	0.000	3.021	0.000
12	هوكم					1		1.778	1.699	0.000	0.000	0.000	0.000	3.021	0.000
13	تاعكر		1					1.778	1.699	0.000	3.021	0.000	0.000	0.000	0.000
14	وجب	1	1	1				1.301	1.398	1.819	1.819	1.819	0.000	0.000	0.000

Setelah memperoleh bobot TF, IDF, IBF, dan IPF$_\propto$ dari dari setiap *term,* maka pembobotan TF.IDF dilakukan terhadap setiap *term* sebagaimana yang ditunjukkan pada **Tabel 3.5**. Begitu pula pembobotan TF.IDF.IBF dilakukan terhadap setiap *term* sebagaimana yang ditunjukkan pada **Tabel 3.6** dan juga pembobotan TF.IDF.IBF.IPF$_\propto$ yang diajukan pada penelitian ini ditunjukkan pada **Tabel 3.7.**

Tabel 3.7 Pembobotan TF.IDF.IBF.IPF$_\propto$ dengan UP=P1 dan $\propto$ = 0.6

No	Term	TF.IDF.IBF.IPF$_\propto$ (dengan persamaan 3.6)					
		D1	**D2**	D3	D4	D5	D6
1	امام	1.290	0.000	1.290	1.290	0.000	1.290
2	**جهيه**	**1.065**	**1.065**	0.000	0.266	0.266	0.266
3	خلف	1.290	0.000	1.290	1.290	0.000	1.290
4	**سهيه**	**1.065**	**1.065**	0.000	0.266	0.266	0.266
5	سواء	4.840	0.000	0.000	0.000	0.000	0.000
6	اله	1.870	1.870	0.000	1.870	3.739	0.000
7	**فلتحه**	**0.800**	**0.800**	0.200	0.200	0.200	0.200
8	**قرا**	**0.800**	**1.000**	0.200	0.400	0.200	0.200
9	**مأموم**	**1.293**	**1.293**	0.000	0.000	0.323	0.323
10	نسبة	0.000	0.000	0.000	0.000	4.840	0.000
11	منذوب	0.000	0.000	0.000	0.000	4.840	0.000
12	مكروه	0.000	0.000	0.000	0.000	4.840	0.000
13	ركعات	0.000	4.840	0.000	0.000	0.000	0.000
14	وجب	2.366	2.366	2.366	0.000	0.000	0.000

Sebelum melakukan perhitungan ukuran 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). Begitu pula jumlah masing-masing *term query* dikalikan dengan bobot IDF dan IBF dari term dokumen sehingga diperoleh TF.IDF.IBF (t_i,q,d_j,b_k). Hasil pembobotan *query* ditunjukkan pada **Tabel** 3.8.

Tabel 3.8 Pembobotan *Query*

No	Term	TF (t_i, q)	IDF (t_i, d_j)	IBF (t_i, b_k)	IPF (t_i, p_l)	TF.IDF (t_i, q, d_j)	TF.IDF.IBF (t_i,q,d_j,b_k)	TF.IDF.IBF.IPF (t_i,q,d_j,b_k,p_l)
1	جهيه	1	1.079	1.097	1.125	1.079	1.184	1.332
2	سهيه	1	1.079	1.097	1.125	1.079	1.184	1.332
3	فلتحه	1	1.000	1.000	1.000	1.000	1.000	1.000
4	قرا	1	1.000	1.000	1.000	1.000	1.000	1.000
5	مأموم	1	1.176	1.222	1.125	1.176	1.437	1.617

Tahap setelah pembobotan adalah perhitungan *cosine similarity* berdasarkan *query* pengguna dengan **Persamaan 3.7**. 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.IBF.IPF$\propto$ ditunjukkan pada **Tabel** 3.9.

Tabel 3.9 Perhitungan *Cosine Similarity*

No	Term	Q	TF.IDF.IBF.IPF$\propto$					
			D1	D2	D3	D4	D5	D6
1	امام	0.000	1.290	0.000	1.290	1.290	0.000	1.290
2	جهيه	1.332	1.065	1.065	0.000	0.266	0.266	0.266
3	خلف	0.000	1.290	0.000	1.290	1.290	0.000	1.290
4	سهي	1.332	1.065	1.065	0.000	0.266	0.266	0.266
5	سواء	0.000	4.840	0.000	0.000	0.000	0.000	0.000
6	اله	0.000	1.870	1.870	0.000	1.870	3.739	0.000
7	فلتحه	1.000	0.800	0.800	0.200	0.200	0.200	0.200
8	قرا	1.000	0.800	1.000	0.200	0.400	0.200	0.200
9	مأموم	1.617	1.293	1.293	0.000	0.000	0.323	0.323
10	نسبة	0.000	0.000	0.000	0.000	0.000	4.840	0.000
11	منذوب	0.000	0.000	0.000	0.000	0.000	4.840	0.000
12	هكاروه	0.000	0.000	0.000	0.000	0.000	4.840	0.000
13	ركعات	0.000	0.000	4.840	0.000	0.000	0.000	0.000
14	وجب	0.000	2.366	2.366	2.366	0.000	0.000	0.000
Jumlah kuarat bobot		8.160	41.070	38.101	9.008	7.166	84.581	3.655
Panjang vector		2.857	6.409	6.173	3.001	2.677	9.197	1.912
Jumlah perkalian bobot *query* dengan bobot dokumen			6.528	6.728	0.400	1.309	1.632	1.632
Nilai *cosine similarity*			0.357	0.382	0.047	0.171	0.062	0.299

Panjang vektor merupakan nilai akar dari jumlah kuadrat bobot di masing-masing dokumen ataupun *query*. Sebagai contoh panjang vektor dokumen D1 = (TF.IDF.IBF.IPF$\propto t1^2$ + TF.IDF.IBF.IPF$\propto t2^2$ + TF.IDF.IBF.IPF$\propto t3^2$ + ... + TF.IDF.IBF.IPF$\propto ti^2$)$^{1/2}$ = (1.290^2 + 1.065^2 + 1.290^2 + 1.065^2 + 4.840^2 + 1.870^2 + 0.800^2 + 0.800^2 + 1.293^2 + 2.366^2)$^{1/2}$ = $(41.070)^{1/2}$ = 6.409. Begitu pula dengan perhitungan panjang vektor dokumen D2 hingga D6 dan panjang vektor *query*.

Setelah itu dilakukan perhitungan jumlah perkalian bobot *query* dengan bobot setiap dokumen. Sebagai contoh jumlah perkalian bobot *query* dengan bobot

dokumen D1 adalah $(1.332 \times 1.065) + (1.332 \times 1.065) + (1 \times 0.800) + (1 \times 0.800) + (1.617 \times 1.293) = 6.528$. 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:

- $\cos(q, d_1) = \frac{6.528}{2.857 \times 6.409} = 0.357$
- $\cos(q, d_2) = \frac{6.728}{2.857 \times 6.173} = 0.382$
- $\cos(q, d_3) = \frac{0.4}{2.857 \times 3.001} = 0.047$
- $\cos(q, d_4) = \frac{1.309}{2.857 \times 2.677} = 0.171$
- $\cos(q, d_5) = \frac{1.632}{2.857 \times 9.197} = 0.062$
- $\cos(q, d_6) = \frac{1.632}{2.857 \times 1.912} = 0.299$

Tabel 3.10 menunjukkan hasil perhitungan *cosine similarity* yang telah diurutkan mulai dari nilai tertinggi hingga terendah dengan variasi pembobotan TF.IDF, TF.IDF.IBF dan TF.IDF.IBF.IPF$_\propto$. Dari tabel tersebut diketahui rangking dokumen berdasarkan nilai similaritas dokumen tersebut terhadap *query*. Hasil tersebut menunjukkan bahwa dengan metode TF.IDF, dokumen D1 dan D2 yang termasuk dalam preferensi pengguna berada di urutan ke-2 dan ke-4. Hasil yang tidak jauh beda juga ditunjukkan dengan metode TF.IDF.IBF, dokumen D2 dan D1 berada pada urutan ke-3 dan ke-4. Sedangkan dengan metode TF.IDF.IBF.IPF$_\propto$, dokumen D1 dan D2 yang merupakan anggota dari P1 yang menjadi preferensi pengguna ditempatkan pada urutan yang lebih tinggi dibandingkan dengan dokumen lain, yaitu pada urutan ke-1 dan ke-2.

Tabel 3.10 Hasil Perangkingan Dokumen

Ranking	TF.IDF		TF.IDF.IBF		TF.IDF.IBF.IPF$_\propto$	
	Dokumen	Similaritas	Dokumen	Similaritas	Dokumen	Similaritas
1	D6	0.821	D6	0.821	**D2**	**0.372**
2	**D2**	**0.691**	D4	0.609	**D1**	**0.357**
3	D4	0.658	**D2**	**0.567**	D6	0.299
4	**D1**	**0.623**	**D1**	**0.527**	D4	0.171
5	D5	0.525	D5	0.402	D5	0.062
6	D3	0.329	D3	0.259	D3	0.047

3.3 Implementasi algoritma

Pada tahapan ini dilakukan implementasi desain model sistem ke dalam kode program sehingga dapat dimengerti oleh komputer. Sistem yang dibangun adalah *dekstop based application* dengan bahasa pemrograman Java dan *database* MySql. Setelah melalui proses *text preprocessing* data disimpan kedalam *database.* Sehingga untuk proses selanjutnya data-data hasil ekstraksi diakses dari *database.*

Terdapat dua komponen utama dalam fase ini yaitu pengembangan sistem perangkingan dokumen berbahasa Arab sesuai dengan metode yang diusulkan dan pengembangan *interface* sistem sebagai sarana 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, 64 bit
   b. Netbeans 8 dengan bahasa pemrograman Java.
   c. *Database server*: Mysql 5.5.24.
2. Spesifikasi perangkat keras yang digunakan:
   a. *Processor*: Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz.
   b. *Memory* (RAM): 4 GB (3.88 *usable*).

Penelitian ini dilaksanakan di laboratorium Pascasarjana Fakultas Teknologi Informasi, Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember. Penelitian ini dimulai dari bulan Oktober 2014 s/d Januari 2015.

3.4 Pengujian dan Evaluasi

3.4.1 Skenario Uji Coba

Uji coba dilakukan dengan melakukan pencarian dan mengevaluasi hasil perangkingan dokumen. Uji coba yang dilakukan adalah uji coba dengan variasi nilai $\propto$ sebagaimana **Tabel** 3.11. Uji coba variasi nilai $\propto$ dilakukan untuk mengetahui nilai $\propto$ maksimal serta pengaruh ukuran nilai $\propto$ terhadap perangkingan dokumen. Pengujian dilakukan terhadap sejumlah *query*, mulai Q ke-1 hingga Q ke n. Untuk masing-masing *query* dilakukan pengujian dengan kondisi preferensi pengguna yang beragam, mulai dari P-1 hingga P-4. Tiap ragam *query* dan preferensi pengguna tersebut dilakukan pengujian dengan nilai $\propto = 0$ hingga $\propto =$

1. Dari masing-masing variasi tersebut dapat diperoleh nilai *recall* (R), *precision* (P), dan *f-measure* (F) sebagai nilai evaluasi kemampuan system. Pembahasan tentang evaluasi kemampuan sistem dalam melakukan pencarian dokumen sesuai preferensi pengguna menggunakan *recall*, *precision*, dan *f-measure* akan dijelaskan pada **subbab 3.4.2**.

Tabel 3.11 Skenario Uji Coba variasi nilai ∝

Query	Preferensi Pengguna	0			0.1			0.2			….			0.8			0.9			1		
		R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Q-1	P-1																					
	P-2																					
	P-3																					
	P-4																					
Q-…	P-1																					
	P-2																					
	P-3																					
	P-4																					
Q-n	P-1																					
	P-2																					
	P-3																					
	P-4																					

Metode TF.IDF.IBF.IPF∝ dengan nilai ∝ maksimal berdasarkan uji coba pertama akan dibandingkan dengan metode pembobotan yang lainnya, yaitu TF.IDF dan TF.IDF.IBF. Skenario pengujian kedua ini ditunjukkan pada **Tabel** 3.12. Pengujian kedua ini juga dilakukan terhadap sejumlah *query* mulai dari Q ke-1 hingga Q ke-n dengan kondisi preferensi pengguna yang beragam mulai dari P-1 hingga P-4. Dari masing-masing variasi tersebut dapat diperoleh nilai *recall*, *precision*, dan *f-measure* menggunakan metode TF.IDF.IBF.IPF∝ serta dibandingkan dengan nilai *recall*, *precision*, dan *f-measure* menggunakan metode TF.IDF dan TF.IDF.IBF.

Tabel 3.12 Skenario Uji Coba Perbandingan Metode

Query	Preferensi Pengguna	TF.IDF			TF.IDF.IBF			TF.IDF.IBF.IPF$_{\propto}$		
		R	P	F	R	P	F	R	P	F
Q-1	P-1									
	P-2									
	P-3									
	P-4									
Q-…	P-1									
	P-2									
	P-3									
	P-4									
Q-n	P-1									
	P-2									
	P-3									
	P-4									

3.4.2 Evaluasi dengan *Recall*, *Precision*, dan *F-measure*

Dalam penelitian ini, metode evaluasi perangkingan yang digunakan adalah *precision, recall,* dan *F-measure*. *F-measure* digunakan pada temu kembali informasi dengan mengkombinasikan konsep *recall* dan *precision*.

Metode sistem temu kembali informasi diaplikasikan untuk mendapatkan informasi yang relevan dengan keinginan pengguna. Dari kumpulan dokumen yang ada penelitian ini melakukan perangkingan berdasarkan *query* yang diberikan pengguna. Dari *query* ini di cari dokumen yang relevan dan sesuai dengan preferensi pengguna.

Evaluasi dilakukan untuk menganalisa performa dari metode perangkingan yang digunakan. Analisa performa ini menggambarkan efektifitas metode yang direlasikan dengan kemampuan metode untuk mengembalikan dokumen yang relevan dengan *query* dan sesuai dengan preferensi pengguna. Disini metode temu kembali informasi berusaha untuk mengembalikan dokumen yang relevan sesuai dengan preferensi dan tidak mengembalikan dokumen yang tidak relevan.

Nilai *recall* dan *precision* sistem temu kembali informasi tersebut dapat dinyatakan sebagai **Persamaan 3.8** dan **Persamaan 3.9**. Sebagaimana yang ditunjukkan pada

Tabel 3.13, TP (*true positive*) merupakan jumlah dokumen relevan sesuai dengan preferensi pengguna yang ter-*retrieve* secara tepat oleh sistem. FP (*false positive*) merupakan jumlah dokumen relevan yang tidak sesuai preferensi pengguna yang ter-*retrieve*. TN (*true negative*) merupakan jumlah dokumen relevan sesuai preferensi pengguna yang tidak ter-*retrieve*. Sedangkan *false negative* merupakan jumlah dokumen relevan yang tidak sesuai preferensi pengguna yang tidak ter-*retrieve*. Berdasarkan evaluasi menggunakan precission, maka efektivitas dari metode yang digunakan dapat diketahui.

$$\boldsymbol{Precision} = \frac{\boldsymbol{tp}}{\boldsymbol{tp+fp}}, \tag{3.8}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{tp}}{\boldsymbol{tp+tn}}, \tag{3.9}$$

Tabel 3.13 Tabel *Recall Precision*

		Document retrieval secara Manual	
		Dokumen Relevan yang sesuai Preferensi pengguna	**Dokumen Relevan yang Tidak sesuai Preferensi pengguna**
Document retrieval menggunakan TF.IDF.IBF.IPF∝	**Dokumen yang Ditemukan**	*True Positive (tp)*	*False Positive (fp)*
	Dokumen yang Tidak Ditemukan	*True Negative (tn)*	*False Negative (fn)*

Pada dasarnya, nilai *recall* dan *precision* berada pada rentang antara 0 sampai dengan 1. Oleh karena itu, suatu sistem temu kembali yang baik adalah yang dapat memberikan nilai *recall* dan *precision* mendekati 1.

Nilai *recall* atau *precision* saja belum cukup mewakili kinerja sistem. Oleh karena itu diperlukan metode evaluasi yang mengkombinasikan metode evaluasi *recall* dan *precision* metode evaluasi ini adalah *F-measure.* Formulasi *F-measure* dinyatakan dalam **Persamaan 3.10,** dengan *r* adalah *recall*, *p* adalah *precission*

$$\boldsymbol{F-measure} = \frac{\boldsymbol{2rp}}{\boldsymbol{r+p}} \tag{3.10}$$

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bagian ini dipaparkan hasil implementasi dari setiap langkah yang telah dipaparkan pada Bab 3. Kemudian dilanjutkan dengan menunjukkan hasil dari uji coba sesuai dengan skenario pengujian yakni perhitungan *precision, recall*, dan *f-measure* pada metode yang diusulkan dibandingkan dengan beberapa metode sebelumnya. Setelah itu dipaparkan evaluasi dan pembahasan hasil yang diperoleh pada bagian akhir bab ini.

4.1 Implemantasi

Metode yang diusulkan diimplementasikan dengan menggunakan bahasa pemrograman *Java* pada *platform Java Development 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 di atas *platform Microsoft Windows 8*, dengan spesifikasi *processor Core i3* dan *Memory DDR III 2 GB*.

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

4.1.1 Pembuatan Indeks Dokumen

Pembuatan indeks dokumen dilakukan melalui beberapa tahap. Tahap pertama adalah pengambilan data preferensi, buku dan isi dokumen dari database. Tahap selanjutnya adalah *preprocessing* yang mencakup *tokenisasi, filtering, stemming* dan penghapusan *stopword*. Hasil dari *preprocessing* tersebut adalah kumpulan *term* dari seluruh dokumen yang menjadi set fitur asli. Setelah itu dilakukan pembobotan dengan pembobotan TF, IDF, IBF, dan IPF pada masing-masing term tersebut. Kemudian dilakukan pembobotan $TF.IDF.IBF.IPF_{\propto}$.

4.1.2 Pengambilan data dari *database*

Data yang digunakan pada penelitian ini disimpan pada *database MySQL* yang terdiri dari tabel buku, tabel preferensi 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 perantaraan *driver* JDBC. Untuk melakukan 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 dBase; //nama database
[3] String username; //username database
[4] String passWord; //pasword database
[5] this.destroyConnection();  //method putus koneksi
[6] String dbDriver ; //driver JDBC
[7] String dbURL ; //Driver URL JDBC
[8] Class.forName(dbDriver);
[9] //untukkoneksi
[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 *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, diantanya adalah *tokenisasi, filtering, stemming*, dan *stopword removal*. Tokenisasi dilakukan untuk memecah keseluruhan isi dokumen menjadi suku kata tunggal. Sedangkan pada tahapan *filtering* dilakukan pembuangan harokat-harokat bahasa Arab. Penghilangan *stopword* dilakukan untuk menghilangkan kata-kata yang sering muncul dalam dokumen tetapi tidak mempunyai nilai yang berarti pada sebuah dokumen. Daftar stopword diambil dari website *http://arabicstemmer.codeplex.com/*. **Gambar** 4.2 menunjukkan potongan daftar stopword pada database. 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*.

id ▲ 1	word	stemmedword
1	الله	الله
2	بيد	بيد
3	وبيد	و-بيد
4	فبيد	ف-بيد
5	سوى	سوى
6	وسوى	و-سوى
7	فسوى	ف-سوى
8	غير	غير
9	بغير	ب-غير
10	كغير	ك-غير
11	لغير	ل-غير
12	لغير	ل-غير
13	وغير	و-غير
14	فغير	ف-غير
15	وبغير	و-ب-غير

Gambar 4.2 Daftar *Stopword*

```
[1] List<String> result = new ArrayList<>();
[2] Version version = Version.LUCENE_43;
[3] ArabicAnalyzer arabicAnalyzer = new ArabicAnalyzer(version);
[4] TokenStream stream  = analyzer.tokenStream(field, new
     StringReader(keywords));
[5] stream.reset();
[6] try {
[7]   while(stream.incrementToken()) {
[8]     String temp =
        stream.getAttribute(CharTermAttribute.class).toString();
[9]     temp=temp.replace("'", "");
[10]    temp=temp.replace("\"", "");
[11]    temp=temp.replace(" ", "");
[12]    temp=temp.replaceAll("[0-9]","");
[13]    temp=temp.trim();
[14]       if(!stopword.contains(StringUtils.lowerCase(temp)) &&
           temp.compareTo("")!=0) {
[15]      result.add(temp);
[16]    }
[17]  }
[18] }
[19] catch(IOException e) {
[20]     // not thrown b/c we're using a string reader...
[21] }
```

Gambar 4.3 Method untuk *preprocessing* dokumen

Pada **Gambar** 4.3 ditampilkan kode untuk *preprocessing* dokumen dengan menggunakan library *Lucene,* yaitu dengan pemanggilan modul *Arabic Analyzer* sebagaimana yang ditunjukkan pada baris ke-3. Pada tahap *preprocessing* juga dilakukan penghilangan karakter angka maupun tanda baca. Untuk setiap *term* yang telat di-token dilakukan pengecekan pada daftar *stopword*. Pengecekan tersebut ditunjukkan pada baris ke-14.

Preprocessing dilakukan terhadap setiap dokumen yang diambil dari database. Setelah dilakukan *preprocessing*, hasil term dari keseluruhan dokumen disimpan dalam database untuk diproses pada tahap pembobotan. Kode yang digunakan untuk mengambil data, memanggil kelas *preprocessing*, dan menyimpan *term* ditunjukkan pada **Gambar** 4.4. Baris ke-2 hingga baris ke-6 menunjukkan proses pengambilan data dari database. Baris ke-7 menunjukkan pemanggilan kelas

stemming untuk melakukan *preprocessing* data. Hasil *preprocessing* disimpan sementara pada list. Term yang telah pada list tersebut disimpan pada database dengan kode baris ke-8 hingga ke-10. Daftar *term* hasil tahap *preprocessing* yang telah disimpan pada database ditunjukkan pada **Gambar** 4.5. Dari 120 dokumen yang digunakan pada penelitian ini, diperoleh 16435 *term* hasil *preprocessing* dengan *distict term* sejumlah 1611.

```
[1] for (int i = 0; i <= 120; i++) {
[2]    sql = "SELECT id_doc, text FROM document WHERE id_doc=" + i + "";
[3]    ResultSet rs = theKoneksi.executeSelect(sql);
[4]     while (rs.next()) {
[5]          id_doc = Integer.parseInt(rs.getString(1));
[6]          text = rs.getString(2);
[7]          List result = Stemming.parseKeywords(text, 1, stopword, se);
[8]          for (int j = 0; j < result.size(); j++) {
[9]          sql2 = "INSERT INTO term (id_doc, term) "
                    + " VALUES ('" + id_doc + "', '" + result.get(j).toString()
                    + "')";
[10]         theKoneksi.executeUpdate(sql2);
[11]         }
[12]      }
[13] }
```

Gambar 4.4 Kode penyimpanan *term* hasil *preprocessing*

id_doc	term
1	كتب
1	صلي
1	ان
1	بدأ
1	كتب
1	صلي
1	صلي
1	قوا
1	ركن
1	يمن
1	حلا
1	قول
1	حلا
1	توب
1	قوم

Gambar 4.5 Daftar *term* hasil *preprocessing*

4.1.4 Pembobotan TF.IDF.IBF.IPF∝

Tahap selanjutnya setelah *preprocessing* adalah *term weighting*. Pembobotan ini dilakukan dengan cara menghitung TF (*Term Frequency*), IDF (*Inverse Document Frequency*), IBF (*Inverse Book Frequency*), kemudian IPF (*Inverse Document Frequency*) dari masing-masing *term* pada seluruh dokumen. Setelah diperoleh nilai TF, IDF, IBF, dan IPF dari masing-masing term, maka dilakukan pembobotan TF.IDF.IBF.IPF∝.

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

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-groub berdasarkan *term*. Langkah selanjutnya, perhitungan IDF dilakukan melalui program java dengan fungsi log pada java yaitu *Math.log10();*. Sehingga *idf = Math.log10(num_doc/df);*

Pembobotan IBF dilakukan dengan menghitung persebaran *term* pada seluruh buku dan IPF menghitung persebaran term pada seluruh kelompok preferensi. Nilai IBF dan IPF sebuah *term* masing-masing berbanding terbalik dengan jumlah buku dan kepompok preferensi yang mengandung *term* tersebut. Sama dengan perhitungan IDF, langkah pertama pembobotan IBF dan IPF dilakukan dengan kode SQL. Untuk mendapatkan bobot IBF, terlebih dahulu dilakukan perhitungan sebaran *term* pada kumpulan buku yang digunakan, hal ini dilakukan dengan kode *count(term)* yang di groub berdasarkan *id_book* dan *term*. Setelah proses *select* sebaran *term* tersebut maka dapat dilakukan perhitungan jumlah buku yang memuat *term* dengan kode *count(id_book)* dari sebaran *term.* Sehingga dapat diperoleh nilai *book frequency*(bf).

id_doc	id_book	id_prefer	term	TF
64	7	3	أبق	2
97	10	4	أبق	1
2	1	1	أبن	1
3	1	1	أبن	1
4	1	1	أبن	1
5	1	1	أبن	1
6	1	1	أبن	2
8	1	1	أبن	1
11	2	1	أبن	1
12	2	1	أبن	1
13	2	1	أبن	1
16	2	1	أبن	5
20	2	1	أبن	1
23	3	1	أبن	2
31	4	2	أبن	9

(a) Daftar Nilai TF

term	DF	IDF
أبق	2	1.7781512503836436
أبن	42	0.4559319556497244
أبه	12	1
أبو	1	2.0791812460476247
أبي	37	0.5109795219806299
أتم	1	2.0791812460476247
أتي	8	1.1760912590556813
أثر	12	1
أثمد	1	2.0791812460476247
أجب	3	1.6020599913279623
أجج	1	2.0791812460476247
أجر	2	1.7781512503836436
أحد	30	0.6020599913279624
أخا	1	2.0791812460476247
أخر	13	0.965237893740788

(b) DaftarNilai IDF

term	BF	IBF
أبق	2	1.7781512503836436
أبن	8	1.1760912590556813
أبه	5	1.380211241711606
أبو	1	2.079181246047625
أبي	8	1.1760912590556813
أتم	1	2.079181246047625
أتي	6	1.3010299956639813
أثر	8	1.1760912590556813
أثمد	1	2.079181246047625
أجب	3	1.6020599913279625
أجج	1	2.079181246047625
أجر	1	2.079181246047625
أحد	9	1.1249387366083
أخا	1	2.079181246047625
أخر	8	1.1760912590556813

(c) Daftar Nilai IBF

term	PF	IPF
أبق	2	1.3010299956639813
أبن	4	1
أبه	4	1
أبو	1	1.6020599913279625
أبي	4	1
أتم	1	1.6020599913279625
أتي	4	1
أثر	3	1.1249387366083
أثمد	1	1.6020599913279625
أجب	3	1.1249387366083
أجج	1	1.6020599913279625
أجر	1	1.6020599913279625
أحد	4	1
أخا	1	1.6020599913279625
أخر	4	1

(d) Daftar Nilai IPF

Gambar 4.6 Daftar bobot TF, IDF, IBF, dan IPF

Sedangkan untuk mendapatkan bobot IPF, terlebih dahulu dilakukan perhitungan sebaran *term* pada kumpulan preferensi yang ada, hal ini dilakukan dengan kode *count(term)* yang di groub berdasarkan *id_prefer* dan *term*. Setelah

proses *select* sebaran *term* tersebut maka dapat dilakukan perhitungan jumlah kelompok preferensi yang memuat *term* dengan kode *count(id_prefer)* dari sebaran *term.* Sehingga dapat diperoleh nilai *preference frequency*(pf).

Kemudian, sama dengan perhitungan IDF, perhitungan IBF dan IPF dilakukan dengan kode java, yaitu dengan pemanggilan fungsi *Math.log10()*;.Sehingga *ibf = Math.log10(num_book/bf);* dan *ipf = Math.log10(num_prefer/pf);*. **Gambar** 4.6 menunjukkan daftar nilai pembobotan TF, IDF, IBF, dan IPF yang sudah disimpan pada database.

```
[1] sql = "UPDATE weight  SET `" + tipe_weighting + "a`=" + tipe_weighting + "";
[2] theKoneksi.executeUpdate(sql);

[3] query = "SELECT input_term FROM q_term ";
[4] ResultSet rs = theKoneksi.executeSelect(query);
[5] while (rs.next()) {

[6]        sql = "UPDATE `weight` SET " + tipe_weighting + "a = " + tipe_weighting
                    + "*(1-((" + alpha + "/2)+0.5)) "
                    + "WHERE term ='" + rs.getString(1) + "' ";
[7]        theKoneksi.executeUpdate(sql);

[8]         sql = "UPDATE `weight` SET " + tipe_weighting + "a = " + tipe_weighting
                    + "*((" + alpha + "/2)+0.5) "
                    + "WHERE term ='" + rs.getString(1) + "' AND id_prefer="
                    + tipe_preferensi + "";
[9]           theKoneksi.executeUpdate(sql);
[10]       }
```

Gambar 4.7 Kode perhitungan TF.IDF.IBF.IPF∝

Setelah diperoleh bobot TF, IDF, IBF, dan IPF, maka pembobotan TF.IDF.IBF.IPF∝ dilakukan terhadap tiap *term* dengan cara mengalikan tiap bobot tersebut dan dikalikan dengan nilai ∝ sesuai dengan preferensi pengguna dan nilai yang dimasukkan. Implementasi terhadap pembobotan TF.IDF.IBF.IPF∝ dilakukan dengan melakukan update bobot yang sudah disimpan dalam database, yaitu menggunakan kode yang ditunjukkan pada **Gambar 4.7**. Baris ke-6 pada kode tersebut menunjukkan bahwa bobot dikalikan dengan 1-((∝/2)+0.5) ketika *term* dokumen sama dengan *term query*. Sedangkan baris ke-8 pada kode tersebut

menunjukkan bahwa bobot dikalikan dengan ((∝/2) + 0.5) ketika *term* dokumen sama dengan *term query* dan dokumen termasuk dalam kelompok preferensi pengguna. **Gambar** 4.8 menunjukkan nilai TF.IDF, TF.IDF.IBF, dan TF.IDF.IBF.IPF∝. TF.IDF.IBF.IPF∝ merupakan kombinasi relevansi dan preferensi dengan nilai penguat berupa ∝.

term	TF_IDF	TF_IDF_IBF	TF_IDF_IBF_IPF	TF_IDF_IBF_IPFa
أمم	1.0791812460476249	1.5940815562317963	2.0739479201922864	2.0739479201922864
أني	1	1.380211241711606	1.5526530905036269	1.5526530905036269
ائمة	1.7781512503836436	3.1618218692409155	4.113625092828789	4.113625092828789
استاذ	2.0791812460476247	4.322994653916154	6.925696777763742	6.925696777763742
ان	0.7781512503836436	0.960303389852321	0.960303389852321	0.960303389852321
بدأ	1.806179973983887	1.806179973983887	1.806179973983887	1.806179973983887
تلا	1.7781512503836436	3.1618218692409155	5.065428316416663	5.065428316416663
توب	5.334453751150931	11.09129619730142	17.768921889664576	17.768921889664576
تي	2.4717262228329564	2.780540574355311	2.780540574355311	2.780540574355311
جمع	0.40708338811190736	0.40708338811190736	0.40708338811190736	0.40708338811190736
حلي	1.380211241711606	2.0387393611352307	2.293456881189079	2.293456881189079
خصص	1.1249387366083	1.463579039611741	1.6464367557472208	1.6464367557472208
خلل	1.0377885608893997	1.3501940468740652	1.3501940468740652	1.3501940468740652
خول	1.6020599913279623	2.8487049767694255	3.7062506235742876	3.7062506235742876
خيم	2.0791812460476247	4.322994653916154	6.925696777763742	6.925696777763742

Gambar 4.8 Daftar Bobot TF.IDF, TF.IDF.IBF, TF.IDF.IBF.IPF dan TF.IDF.IBF.IPF∝

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 pembobotan TF.IDF.IBF.IPF∝. Pada implementasinya, perhitungan *cosine similarity* ini dilakukan dengan cara membandingkan kedekatan antara matriks *Vector Space Model query* dan matriks *Vector Space Model* masing-masing dokumen.

Sebelum dilakukan perhitungan ukuran kemiripan antara *query* dengan dokumen, terlebih dahulu dilakukan *preprocessing* dan pembobotan terhadap *query* pengguna. Sebagaimana *preprocessing* dokumen, tahap *preprocessing query* juga

dilakukan menggunakan *library lucene* dengan kode pada **Gambar** 4.3**.** Hasil *preprocessing query* disimpan dalam tabel *q_term* sebagai penyimpanan sementara pada database. Tabel tersebut akan diupdate sesuai dengan input pengguna. **Gambar** 4.9 menunjukkan daftar *term query* yang telah di-*preprocessing* dan disimpan dalam database dengan contoh *query* pengguna adalah " الماء للذي ينجس والذي لاينجس" (air yang najis dan yang tidak najis). **Gambar** 4.10 menunjukkan hasil pembobotan *query* dimana nilai IDF, IBF dan IPF diperoleh dari bobot *term* dokumen yang sama dengan *term query*.

input_term
ألم
نجس
نجس

Gambar 4.9 Daftar *term query*

input_term	tf	idf	ibf	ipf	bobot
ألم	1	0.2531064433467984	1	1	0.2531064433467984
نجس	2	0.4259687322722811	1	1	0.8519374645445622

Gambar 4.10 Daftar bobot *query*

Langkah untuk perhitungan *cosine similarity* adalah dengan menghitung perkalian antara vektor *query* dan vektor setiap dokumen kemudian menghitung panjang vektor. Panjang vektor merupakan nilai akar dari jumlah kuadrat bobot di masing-masing dokumen ataupun *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_ibf_ipfa)* 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_ibf_ipfa tf_idf_ibf_ipfa))* 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 + "a) "
        + "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 + "a*" + tipe_weighting
        + "a)) 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,  d.id_book,  r.id_prefer,  (p.dot_product/("
        + vectorQ + "*v.vector_length)) "
        + " FROM proses_dot_product p, vector_length v, document d, book r "
        + " WHERE p.id_doc=v.id_doc AND p.id_doc=d.id_doc AND
            d.id_book=r.id_book )";
[13] theKoneksi.executeUpdate(sql);
```

Gambar 4.11 Kode untuk perhitungan *cosine similarity*

Pada **Gambar** 4.11 ditunjukkan kode untuk perhitungan *cosine similarity*. Baris ke-1 menunjukkan perhitungan perkalian antara vektor *query* dengan vektor dokumen. Hasil perhitunagn perkalian antara vektor *query* dengan vektor dokumen ditunjukkan pada **Gambar** 4.12. Baris ke-5 menunjukkan perhitungan panjang vektor dokumen, sedangkan baris ke-7 menunjukkan perhitungan panjang vektor *query*. **Gambar** 4.13 menunjukkan hasil perhitungan panjang vektor *query,*

sedangkan **Gambar** 4.14 menunjukkan hasil perhitungan panjang vektor dokumen. Perhitungan jarak kosinus antara vektor dokumen dan vektor *query* ditunjukkan pada baris ke-12. Yaitu diperoleh dengan pembagian antara hasil perkalian vektor dengan hasil panjang vektor dokumen dikali hasil panjang vektor *query*.

id_doc	dot_product
2	0.04804715374774954
6	0.2935593862850699
7	0.6991356347214561
8	0.016015717915916515
10	0.10674039835274365
11	0.09072468043682715
13	0.5443480826209629
14	0.03203143583183303
15	0.20283470584824276
16	0.7151513526373726
21	0.8645692776791938
22	0.7257974434946172
23	0.8966007135110269
24	0.28818975922639795
25	0.016015717915916515

Gambar 4.12 Hasil perkalian anatara vektor *query* dengan vektor dokumen

vectorQ
0.8887408593950675

Gambar 4.13 Hasil perhitungan panjang vektor query

id_doc	vector_length
1	38.50862467215533
2	51.10501410632187
3	43.96805539303512
4	60.108893263187746
5	45.889244682776656
6	33.90117071363299
7	66.48519382837738
8	36.836107518824534
9	40.311471700497954
10	10.806775136470561
11	26.557533773404934
12	33.193611053925075
13	24.935398755207082
14	33.06218748791755
15	17.799193573627274

Gambar 4.14 Hasil perhitungan panjang vektor dokumen

id_doc	id_book	id_prefer	similarity ▼ 1
68	7	3	0.22469203496510845
62	7	3	0.13816009637098947
80	8	3	0.08647008526357357
65	7	3	0.07735483434329428
90	9	3	0.062221157384379645
61	7	3	0.06212207366527218
86	9	3	0.059560768647563864
114	12	4	0.044389304002895626
66	7	3	0.039804391000923374
43	5	2	0.03958687334512914
63	7	3	0.036939921419850995
57	6	2	0.03459923487865787
22	3	1	0.02827779509134242
13	2	1	0.02456321631218609
71	8	3	0.024419179944641684

Gambar 4.15 Hasil perhitungan *cosine similarity*

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 kemiripanya antara *query* dan dokumen tersebut. Berdasarkan nilai *cosine similarity* tersebut akan didapatkan hasil perangkingan dokumen sesuai dengan tingkat kemiripan dokumen terhadap *query* diurutkan dari yang paling mirip sampai yang kurang mirip. Hasil perhitungan *cosine similarity* sebagaimana pada **Gambar** 4.15**,** dimana dokumen diurutkan berdasarkan nilai similaritas mulai yang tertinggi hingga terendah.

4.2 Hasil dan Uji Coba

Subbab ini menampilkan hasil pengujian metode yang dikembangkan pada penelitian ini. Pengujian yang pertama yaitu uji coba variasi nilai $\propto$ dilakukan untuk mengetahui nilai $\propto$ maksimal serta pengaruh ukuran nilai $\propto$ terhadap perangkingan dokumen. Metode TF.IDF.IBF.IPF$_{\propto}$ dengan nilai $\propto$ maksimal berdasarkan uji coba pertama akan dibandingkan dengan metode pembobotan yang lainnya, yaitu TF.IDF dan TF.IDF.IBF. Pada tiap pengujian dilakukan pengukuran *precision*, *recall*, dan *F-measure*.

4.2.1 Lingkungan Ujicoba

Proses uji coba aplikasi dilakukan pada komputer dengan spesifikasi prosesor *Intel Core i3* dengan memori *RAM DDR III 4 GB* dan sistem operasi *Microsoft Windows 7*. Aplikasi yang dibangun berjalan di atas *Java Runtime Standard Edition* 7 dengan *database* dataset menggungakan *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, yang diambil dari 12 kitab (buku) dalam *e-book* Bahasa Arab (*http://shamela.ws/*). Masing-masing halaman kitab-kitab tersebut adalah sebuah dokumen. Dokumen yang digunakan adalah 10 dokumen dari

masing-masing buku, sehingga total dokumen sejumlah 120 dokumen. Dan dari seluruh dokumen dataset tersebut terdapat 1.611 kata bebeda (*distinct term*).

id_doc	id_book	page	text
1	1	1	[كتاب الصلاة] ...تم إنه بدأ بكتاب الصلاة؛ لأن الصلاة
2	1	2	[كيفية الوضوء] ... تم بدأ بتعليم الوضوء) فقال: (إذا)
3	1	3	[كيفية الدخول في الصلاة] ...قال (إذا أراد الرجل الدخ
4	1	4	[مكروهات الصلاة] ...قال: (ويكره في الصلاة تغطية الفم
5	1	5	[باب افتتاح الصلاة] ...قال: (وإذا انتهى الرجل إلى ال
6	1	6	[باب الوضوء والغسل] ...قال (يبدأ في غسل الجنابة بيدي
7	1	7	[باب البئر] ...قال (وإذا ماتت الفأرة في البئر ينزح م
8	1	8	[باب المسح على الخفين] ...اعلم) أن المسح على الخفين)
9	1	9	[المسح على العمامة والقلنسوة] ...قال (ولا يجوز المسح
10	1	10	[المسح على الجوربين] ...قال (وأما المسح على الجوربين
11	2	1	باب ما ينقض الوضوء وما لا ينقضه ... محمد عن يعقوب عن
12	2	2	باب المستحاضة ...محمد عن يعقوب عن أبي حنيفة في مستحا
13	2	3	باب ما يجوز به الوضوء وما لا يجوز ...محمد عن يعقوب

Gambar 4.16 Isi Dokumen Fiqih Berbahasa Arab

Salah satu contoh data uji perangkingan dokumen pada penelitian ini terlihat seperti pada **Gambar** 4.16 dimana dokumen berisi teks bahasa Arab yang diambil dari ebook bahasa arab. Dokumen-dokumen inilah yang diproses dari tahap *preprocessing,* penghapusan *stopword,* pembentukan kata dasar, dan proses perangkingan itu sendiri.

Tabel 4.1 Daftar *Query* uji coba

#	*Query*	Keterangan	Jumlah relevan	P1	P2	P3	P4
Q1	استعمال آنية من لا تحل له *isti'malu aaniyatu man laa tahillu dzabiihatahu* menggunakan bejana orang yang tidak halal sembelihannya	Hanabilah, buku al-mabda'u fiy syarhi al-muqhta'i, halaman 7	7	0	0	3	4
Q2	الماء الذي ينجس والذي لا ينجس *al-maa.ul-ladzi yanjisu wal-ladzi laa yanjisu* air yang najis dan yang tidak najis	Syafiiyah, buku al-umm, halaman 2	16	5	2	8	1
Q3	الماء المسخن *al-maa.ul-musakhkhon* air yang dipanaskan	Hanabilah, buku al mughni, halaman 9	4	0	0	1	3
Q4	الماء المشمس *al-maa.ul-musyammas* air yang terkena panas matahari	Hanabilah, buku al mughni, halaman 10	3	0	0	1	2
Q5	الوضوء من النوم *al-wudhuu.u minan-naumi* wudhu setelah bangun tidur	Malikiyah, buku al-Mudwinah, halaman 7	12	2	5	4	1
Q6	غسل الجنابة *ghoslu al-janaabatu* mandi janabah	Hanafiyah, buku al-mabsuth, halaman 6	10	2	3	4	1

Query yang digunakan adalah potongan kalimat yang diambil dari masing-masing kelompok preferensi. Untuk uji coba terdapat enam *query* yang digunakan. Daftar *query* sebagaimana yang ditunjukkan pada **Tabel** 4.1. Pada tabel tersebut juga ditunjukkan jumlah dokumen yang relevan dengan *query* serta sebarannya di masing-masing kelompok dokumen yang menjadi pilihan preferensi pengguna. Keterangan kode pilihan preferensi dapat dilihat pada **Tabel** 4.2. Beberapa *query* memiliki dokumen relevan yang tersebar merata dan terdapat pula *query* yang hanya memiliki dokumen relevan dari beberapa kelompok pilihan preferensi.

Ground thruth yang menjadi patokan dalam evaluasi uji coba penelitian ini dicantumkan pada **Lampiran 1**.

Masing-masing *query* dimasukkan ke dalam aplikasi perangkingan dokumen yang telah dibuat dan dilakukan pencarian berdasarkan *query* tersebut dengan kondisi preferensi pengguna yang beragam. Nama kelompok yang menjadi pilihan preferensi pengguna dalam penelitian ini adalah empat metodologi fiqih. Dalam uji coba, metodologi fiqih tersebut akan dituliskan dalam bentuk kode mulai dari P1 hingga P4 sebagaimana tabel **Tabel** 4.2**.**

Tabel 4.2 Pilihan preferensi pengguna

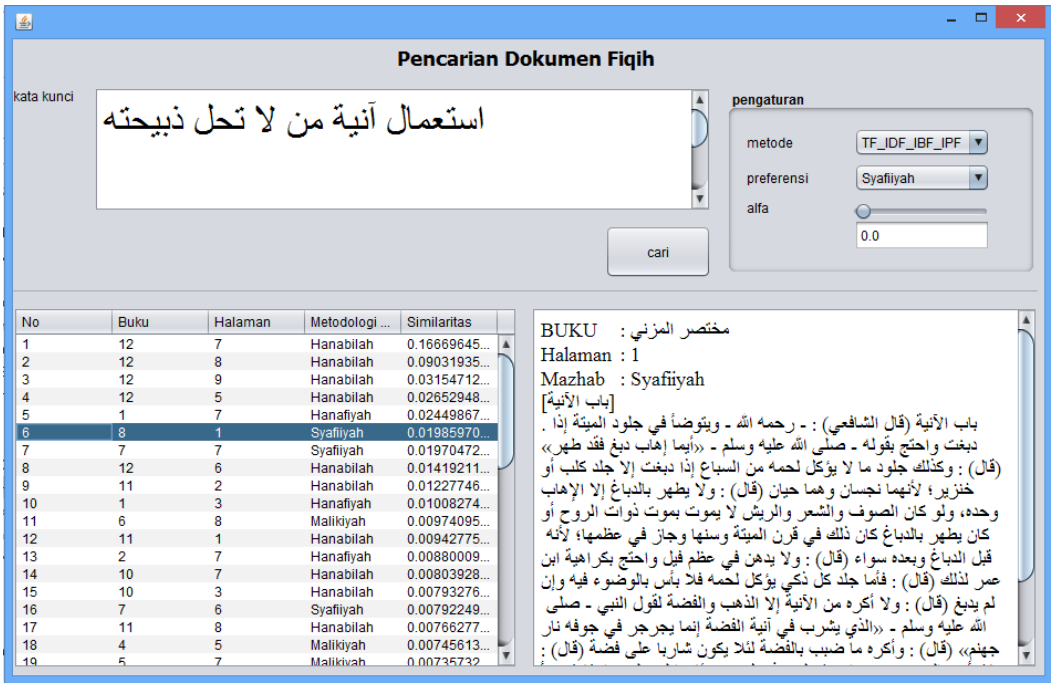
#	Pilihan Preferensi Pengguna
P1	Hanafiyah
P2	Malikiyah
P3	Syafiiyah
P4	Hanabilah

4.2.3 Peringkat Dokumen Hasil Pencarian

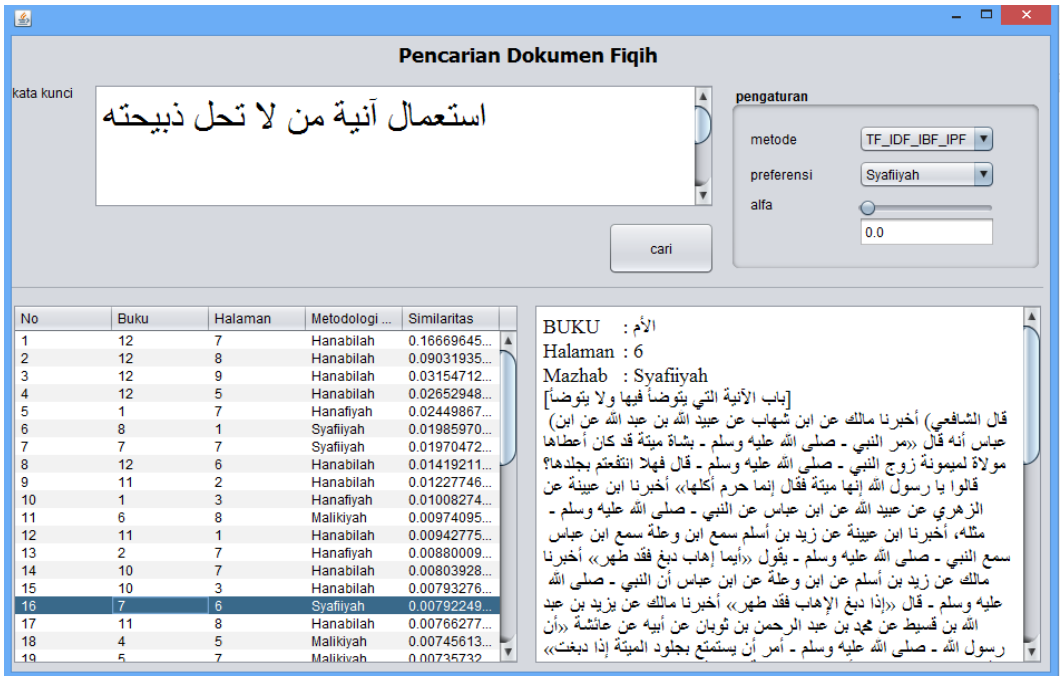
Evaluasi kemampuan sistem dalam merangkingkan dokumen berdasarkan preferensi pengguna dapat dilakukan dengan melihat posisi dokumen relevan yang sesuai dengan preferensi pengguna. Posisi hasil pencarian dokumen relevan terhadap ke-enam *query* yang diujikan dengan kondisi pilihan preferensi pengguna yang beragam serta dengan nilai $\propto$ mulai dari 0 hingga 1. Posisi dokumen relevan sesuai dengan preferensi pengguna secara keseluruhan dapat dilihat pada **Lampiran 2**. Sedangkan rata-rata posisi dokumen relevan sesuai dengan preferensi pengguna ditunjukkan pada **Tabel** 4.3.

Nilai rata-rata posisi tersebut diperoleh dengan menjumlahkan posisi dokumen relevan teratas dengan posisi dokumen relevan terbawah kemudian dibagi dengan dua. Sebagai contoh dilakukan pencarian terhadap query Q1 dengan preferensi pengguna adalah P3 dan nilai $\propto = 0$. Pada hasil pencarian tampak bahwa posisi dokumen relevan yang teratas adalah pada urutan ke-6 sebagaimana pada **Gambar** 4.17 sedangkan posisi dokumen relevan yang terbawah adalah pada

urutak ke-16 sebagaimana pada **Gambar** 4.18**.** Sehingga rata-rata posisi hasil pencarian untuk Q1_P3 dengan nilai $\propto = 0$ adalah (6+16)/2 = 11.



Gambar 4.17 Posisi dokumen relevan sesuai preferensi pengguna yang teratas pada hasil pencarian



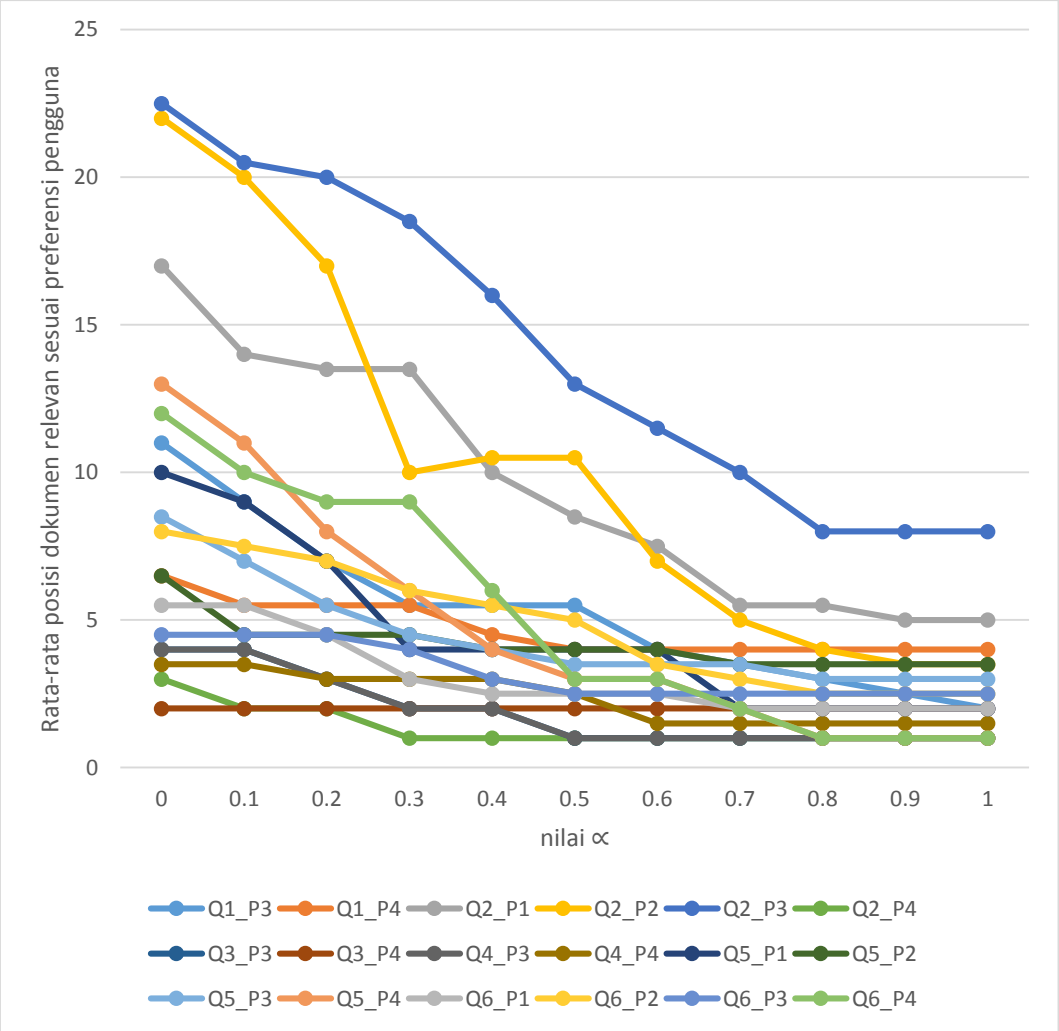
Gambar 4.18 Posisi dokumen relevan sesuai preferensi pengguna yang terbawah pada hasil pencarian

Tabel 4.3 Rata-rata posisi dokumen relevan sesuai preferensi pengguna pada hasil pencarian

Input	TF.IDF	TF.IDF.IBF	TF.IDF.IBF.IPF$\propto$										
			0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Q1_P1													
Q1_P2													
Q1_P3	11.5	11.5	11	9	7	5.5	5.5	5.5	4	3.5	3	2.5	2
Q1_P4	5	5.5	6.5	5.5	5.5	5.5	4.5	4	4	4	4	4	4
Q2_P1	16.5	16.5	17	14	13.5	13.5	10	8.5	7.5	5.5	5.5	5	5
Q2_P2	20.5	22	22	20	17	10	10.5	10.5	7	5	4	3.5	3.5
Q2_P3	25	24	22.5	20.5	20	18.5	16	13	11.5	10	8	8	8
Q2_P4	2	2	3	2	2	1	1	1	1	1	1	1	1
Q3_P1													
Q3_P2													
Q3_P3	4	4	4	4	3	2	2	1	1	1	1	1	1
Q3_P4	2	2	2	2	2	2	2	2	2	2	2	2	2
Q4_P1													
Q4_P2													
Q4_P3	4	4	4	4	3	2	2	1	1	1	1	1	1
Q4_P4	3.5	3.5	3.5	3.5	3	3	3	2.5	1.5	1.5	1.5	1.5	1.5
Q5_P1	7	8	10	9	7	4	4	4	4	2	2	2	2
Q5_P2	6.5	6.5	6.5	4.5	4.5	4.5	4	4	4	3.5	3.5	3.5	3.5
Q5_P3	8.5	8.5	8.5	7	5.5	4.5	4	3.5	3.5	3.5	3	3	3
Q5_P4	14	14	13	11	8	6	4	3	3	2	1	1	1
Q6_P1	5	5.5	5.5	5.5	4.5	3	2.5	2.5	2.5	2	2	2	2
Q6_P2	8	7.5	8	7.5	7	6	5.5	5	3.5	3	2.5	2.5	2.5
Q6_P3	4.5	4.5	4.5	4.5	4.5	4	3	2.5	2.5	2.5	2.5	2.5	2.5
Q6_P4	12	12	12	10	9	9	6	3	3	2	1	1	1

Perhitungan rata-rata posisi dilakukan terhadap variasi query lain dengan nilai $\propto = 0$ hingga $\propto = 1$. Selain itu, perhitungan rata-rata posisi juga dilakukan untuk dua metode lain, yaitu metode TF.IDF dan TF.IDF.IBF sehingga dapat

diketahui bagaimana perbandingan kemampuan metode dalam merangkingkan dokumen sesuai dengan preferensi pengguna. Semakin rendah rata-rata posisi menunjukkan bahwa dokumen relevan berada pada urutan atas, sebaliknya semakin tinggi nilai rata-rata posisi berarti dokumen relevan berada pada urutan bawah. Sehingga semakin kecil nilai rata-rata posisi maka sistem mampu memenuhi kebutuhan pencarian sesuai preferensi pengguna.



Gambar 4.19 Pengaruh nilai ∝ terhadap posisi dokumen relevan sesuai preferensi pengguna pada hasil pencarian

Pada **Tabel** 4.3, kolom yang kosong adalah ketika tidak ada dokumen relevan sesuai preferensi pengguna yang ditemukan. Dari tabel tersebut dapat diketahui bahwa posisi hasil pencarian dengan metode yang ajukan dengan nilai ∝

= 0 hampir sama dengan posisi hasil pencarian dengan metode TF.IDF maupun TF.IDF.IBF. Untuk metode TF.IDF.IBF.IPF$_{\propto}$, semakin tinggi nilai $\propto$ maka rata-rata posisi hasil pencarian yang relevan semakin kecil, ini menunjukkan bahwa nilai $\propto$ berpengaruh dalam menempatkan dokumen relevan sesuai preferensi pengguna pada peringkat yang lebih atas. Semakin tinggi nilai $\propto$ maka dokumen relevan sesuai preferensi pengguna berada pada peringkat semakin tinggi. Hal ini juga sebagaimana yang ditunjukkan pada **Gambar** 4.19.

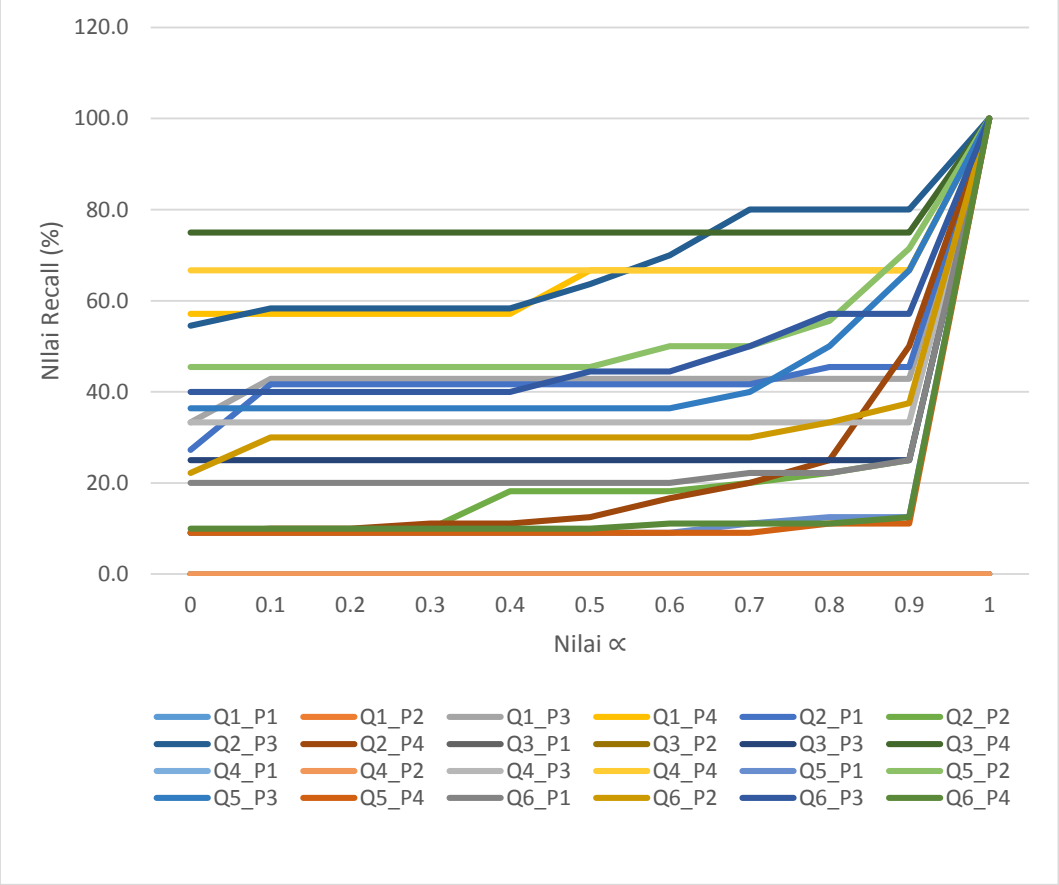
4.2.4 Uji Coba Variasi Nilai $\propto$

Uji Coba ini membandingkan hasil *recall, precision,* dan *f-measure* dari metode TF.IDF.IBF.IPF$_{\propto}$ dengan variasi nilai $\propto$ antara 0 sampai dengan 1. Dimana nilai $\propto$ digunakan untuk menyatakan tinggat preferensi pengguna. Secara keseluruhan, hasil *recall*, *precision*, dan *f-measure* dari variasi nilai $\propto$ antara 0 sampai dengan 1 ditampilkan pada **Lampiran 3**.

Evaluasi *recall* menunjukkan kemampuan sistem dalam me-*retrieve* dokumen yang relevan sesuai dengan preferensi pengguna. Pada **Gambar 4.20** tampak bahwa terhadap perubahan nilai $\propto$ untuk semua *query, recall* cenderung stabil dan meningkat perlahan hingga pada $\propto=0.9$ recall meningkat secara drastis menjadi 100% pada $\propto=1$. Hal ini karena pada $\propto=1$ hanya dokumen yang sesuai dengan preferensi pengguna yang ter-*retrieve* dan mengabaikan dokumen relevan dari kelompok pilihan preferensi yang lain. Sedangkan untuk $\propto=0$ hingga $\propto=0.9$ sistem tidak hanya me-*retrieve* dokumen yang sesuai preferensi pengguna namun juga dari kelompok pilihan preferensi yang lain.

Dari gambar tersebut pun tampak bahwa *recall* tertinggi ketika menggunakan query Q3 dengan pilihan preferensi P4, yaitu sebesar 75% untuk $\propto=0$ hingga $\propto=0.9$ dan mencapai 100% untuk $\propto=1$. Hal ini karena Q3 merupakan penggalan kalimat yang diambil dari pilihan preferensi P4 (lihat **Tabel 4.1**) sedangkan dari pilihan preferensi lain tidak ditemukan dokumen yang relevan sehingga *term query* Q3 dengan preferensi pengguna P4 memiliki signifikasi yang tinggi terhadap preferensi pilihan preferensi lain dibandingkan dengan *query* yang lainnya.

Namun demikian, nilai *recall* Q2_P3 lebih tinggi dibandingkan *recall* Q3_P4 pada kondisi $\propto$=0.7 hingga $\propto$=0.9 yaitu mencapai 80% dan 100 % pada $\propto$=1. Peningkatan tersebut cukup signifikan melihat posisi awal *recall* Q2_P3 hanya sebesar 54.5% pada $\propto$=0. Hal ini karena terdapat banyak dokumen relevan dengan query yang berada pada kelompok pilihan preferensi yang lainnya, sehingga ketika nilai $\propto$ kecil posisi dokumen relevan yang sesuai dengan preferensi pengguna masih berada pada peringkat yang lebih rendah dibandingkan dengan dokumen relevan dari kelompok pilihan preferensi lain.

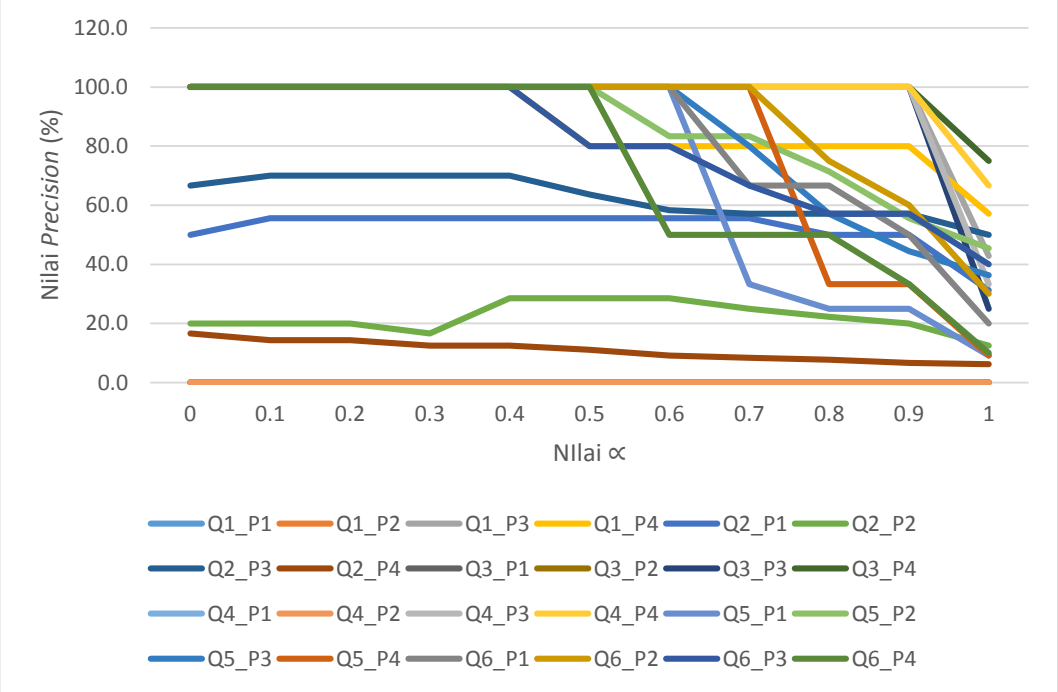


Gambar 4.20 Grafik *Recall* metode TF.IDF.IBF.IPF$_{\propto}$

Secara keseluruhan dari semua *query* yang diujikan, mayoritas peningkatan *recall* terjadi mulai drastis saat $\propto$=0.4 dan meningkat lagi saat $\propto$=0.7 hingga $\propto$=0.9. Oleh karena itu, dapat disimpulkan bahwa $\propto$ memiliki pengaruh dalam meningkatkan kemampun perangkingan dokumen yang sesuai dengan preferensi

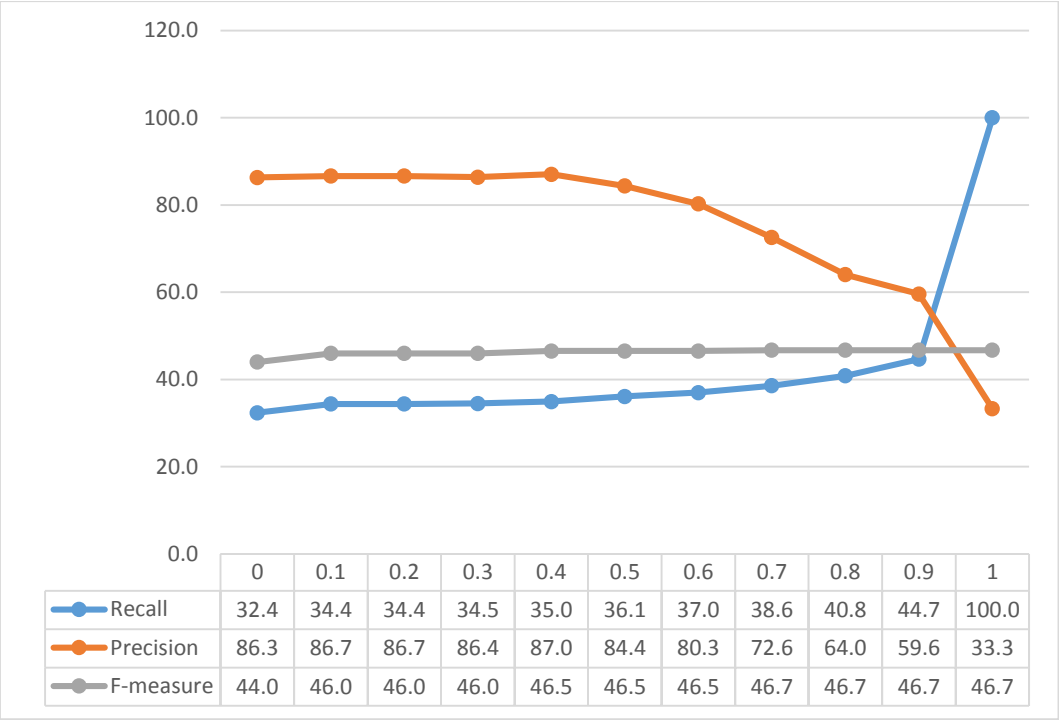
pengguna. Semakin tinggi nilai ∝ maka dokumen relevan yang sesuai preferensi penggguna dapat ter-*retrieve* dan berada pada posisi yang lebih atas dibandingkan dengan dokumen lain yang bukan termasuk preferensi pengguna.

Evaluasi *precision* digunakan untk mengetahui kemampuan sistem dalam me-*retrieve* dokumen. Berbeda dengan *recall*, nilai *precision* cenderung menurun karena semakin banyak dokumen yang ter-*retrive* maka dokumen yang tidak relevan pun akan semakin banyak dibandingkan dengan yang relevan. Hal ini pun tampak pada uji coba penelitian ini yang ditunjukkan pada **Gambar 4.21.** Hampir semua *query* mengalami penurunan nilai *precision* baik secara perlahan maupun drastis. Penurunan nilai *precision* secara drastis mulai tampak pada saat ∝=0.4 sebagaimana yang terjadi pada Q6_P3 yang menurun dari 100% menjadi 80%, hal ini berkebalikan dengan nilai *recall* yang meningkat. Namun juga terdapat query yang memiliki nilai *precision* yang stabil yaitu Q4_P4 yang tetap bernilai 100% hingga pada ∝=0.9. Hal ini karena pada Q4 untuk preferensi pengguna P4 tidak ditemukan dokumen relevan dari kelompok pilihan preferensi lain.



Gambar 4.21 Grafik *Precision* metode TF.IDF.IBF.IPF∝

Sistem yang baik adalah ketika memiliki kemampuan untuk me-*retrieve* dokumen dan mayoritas dari dokumen yang ter-*retrieve* adalah dokumen yang relevan. Oleh karena itu, untuk menentukan nilai $\propto$ yang optimal pada metode TF.IDF.IBF.IPF$_{\propto}$ yang diajukan ini perlu melihat perbandingan recall dan precision untuk setiap nilai $\propto$. Hal ini ditunjukkan pada **Gambar 4.22.** Pada gambar tersebut nilai rata-rata *recall*, *precision*, dan *f-measure* meruapakan rata-rata untuk semua *query* yang digunakan dengan mengabaikan variasi *query* dan preferensi pengguna yang tidak memiliki dokumen relevan sesuai dengan preferensi sehingga memiliki nilai recall 0% dan precision 0%.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Recall	32.4	34.4	34.4	34.5	35.0	36.1	37.0	38.6	40.8	44.7	100.0
Precision	86.3	86.7	86.7	86.4	87.0	84.4	80.3	72.6	64.0	59.6	33.3
F-measure	44.0	46.0	46.0	46.0	46.5	46.5	46.5	46.7	46.7	46.7	46.7

Gambar 4.22 Rata-rata nilai *recall*, *precision*, dan *f-measure* di setiap nilai $\propto$

Pada **Gambar 4.22** tampak bahwa rata-rata nilai *recall* cenderung naik. Sebaliknya, rata-rata nilai *precision* cenderung turun. Sedangkan rata-rata *f-measure* cenderung stabil. Pada penilitian ini, nilai $\propto$ dianggap optimal ketika memberikan *recall* yang tinggi dengan tetap memiliki kemampuan me-*retrieve* dokumen. Pada gambar tampak bahwa rata-rata *recall* tertinggi adalah ketika $\propto=1$ yaitu mencapai 100%. Namun, pada $\propto=1$ rata-rata *precision* hanya bernilai 33.3% karena hanya me-*retrieve* dokumen yang sesuai dengan preferensi pengguna. Oleh

karena itu, $\propto$ yang dianggap optimal adalah $\propto=0.9$ dengan rata-rata recall sebesar 44.7%, rata-rata *precision* sebesar 59.6%, dan rata-rata *f-measure* sebesar 46.7%.

4.2.5 Uji Coba Perbandingan Metode

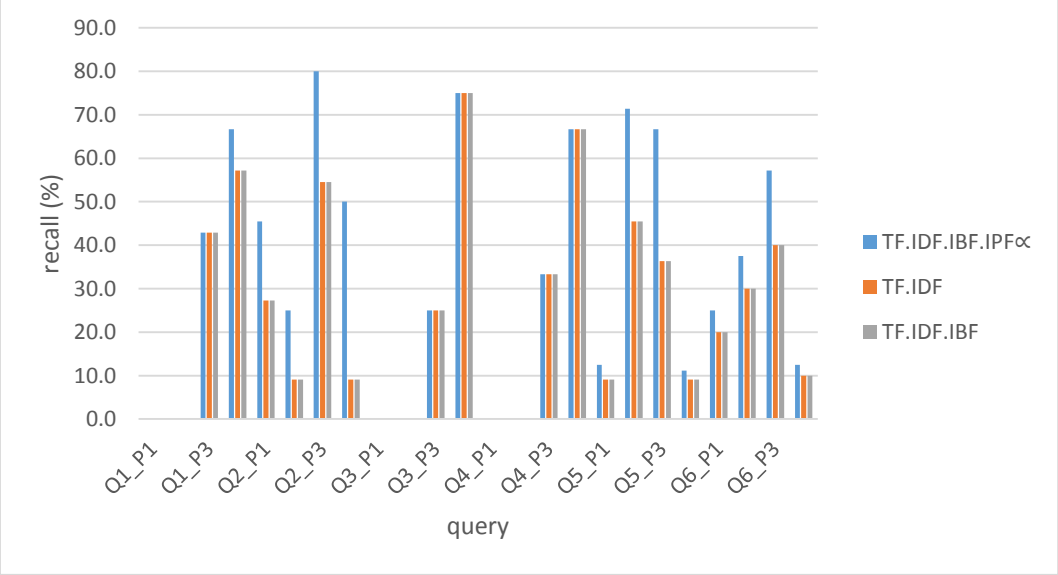
Uji Coba ini membadingkan kemampuan metode yang diajukan dengan metode pembobotan yang telah ada sebelumnya dalam perangkingan dokumen yang memenuhi preferensi pengguna. Metode TF.IDF.IBF.IPF$_\propto$ dibandingkan dengan metode TF.IDF serta TF.IDF.IBF. Untuk metode TF.IDF.IBF.IPF$_\propto$ yang dibandingkan adalah dengan $\propto=0.9$.

Tabel 4.4 Perbandingan *Recall*, *Precision*, dan *F-measure* Metode TF.IDF, TF.IDF.IBF, dan TF.IDF.IBF.IPF$_\propto$

Input	TF.IDF.IBF.IPF$_\propto$			TF.IDF			TF.IDF.IBF		
	R (%)	P (%)	F (%)	R (%)	P (%)	F (%)	R (%)	P (%)	F (%)
Q1_P1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q1_P2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q1_P3	42.9	100.0	60.0	42.9	100.0	60.0	42.9	100.0	60.0
Q1_P4	66.7	80.0	72.7	57.1	100.0	72.7	57.1	100.0	72.7
Q2_P1	45.5	50.0	47.6	27.3	50.0	35.3	27.3	50.0	35.3
Q2_P2	25.0	20.0	22.2	9.1	20.0	12.5	9.1	20.0	12.5
Q2_P3	80.0	57.1	66.7	54.5	66.7	60.0	54.5	66.7	60.0
Q2_P4	50.0	6.7	11.8	9.1	16.7	11.8	9.1	16.7	11.8
Q3_P1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q3_P2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q3_P3	25.0	100.0	40.0	25.0	100.0	40.0	25.0	100.0	40.0
Q3_P4	75.0	100.0	85.7	75.0	100.0	85.7	75.0	100.0	85.7
Q4_P1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q4_P2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q4_P3	33.3	100.0	50.0	33.3	100.0	50.0	33.3	100.0	50.0
Q4_P4	66.7	100.0	80.0	66.7	100.0	80.0	66.7	100.0	80.0
Q5_P1	12.5	25.0	16.7	9.1	100.0	16.7	9.1	100.0	16.7
Q5_P2	71.4	55.6	62.5	45.5	100.0	62.5	45.5	100.0	62.5
Q5_P3	66.7	44.4	53.3	36.4	100.0	53.3	36.4	100.0	53.3
Q5_P4	11.1	33.3	16.7	9.1	100.0	16.7	9.1	100.0	16.7
Q6_P1	25.0	50.0	33.3	20.0	100.0	33.3	20.0	100.0	33.3
Q6_P2	37.5	60.0	46.2	30.0	100.0	46.2	30.0	100.0	46.2
Q6_P3	57.1	57.1	57.1	40.0	100.0	57.1	40.0	100.0	57.1
Q6_P4	12.5	33.3	18.2	10.0	100.0	18.2	10.0	100.0	18.2
Rata-rata	44.7	59.6	46.7	33.3	86.3	45.1	33.3	86.3	45.1

Perbandingan metode secara keseluruhan ditunjukkan pada **Tabel 4.4**. Pada tabel tersebut dicantumkan nilai *recall*, *precision*, dan *f-measure* dari setiap input *query* dan preferensi pengguna. Disertakan pula nilai rata-rata *recall*, *precision*, dan *f-measure* dan semua input dengan mengabaikan input yang tidak memiliki dokumen relevan sesuai dengan preferensi pengguna.

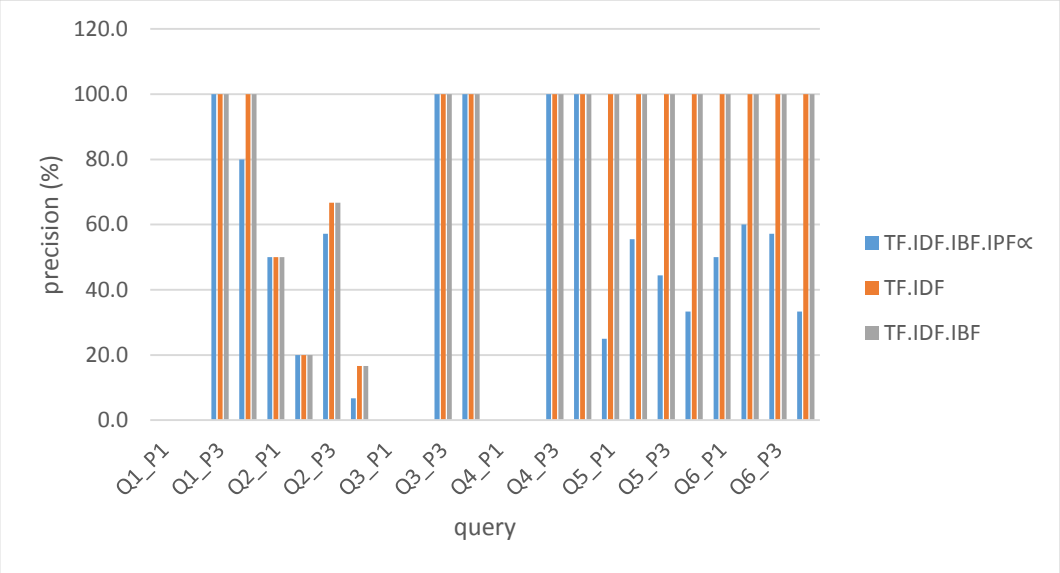
Dapat dilihat pada **Gambar 4.23**, dari 18 variasi input query dan preferensi pengguna terdapat 13 input yang memberikan nilai *recall* lebih unggul dibandingkan dengan metode TF.IDF dan TF.IDF.IBF. Sedangkan 5 input lain memiliki nilai *recall* yang sama dengan metode TF.IDF dan TF.IDF.IBF. Diantara query yang unggul adalah Q1_P4 dengan nilai 9.5% lebih unggul, Q2_P1 dengan nilai 18.2% lebih unggul, Q2_P2 dengan nilai 15.9% lebih unggul, Q2_P3 dengan nilai 22.5% lebih unggul, Q2_P4 dengan nilai 40.9% lebih unggul, demikian seterusnya. Ini menunjukkan bahwa metode TF.IDF.IBF.IPF∝ yang diajukan pada penelitian ini terbukti mampu melakukan perangkingan dokumen sesuai dengan preferensi pengguna dibandingkan dengan metode pembobotan kata yang sudah ada sebelumnya.



Gambar 4.23 Perbandingan *Recall*

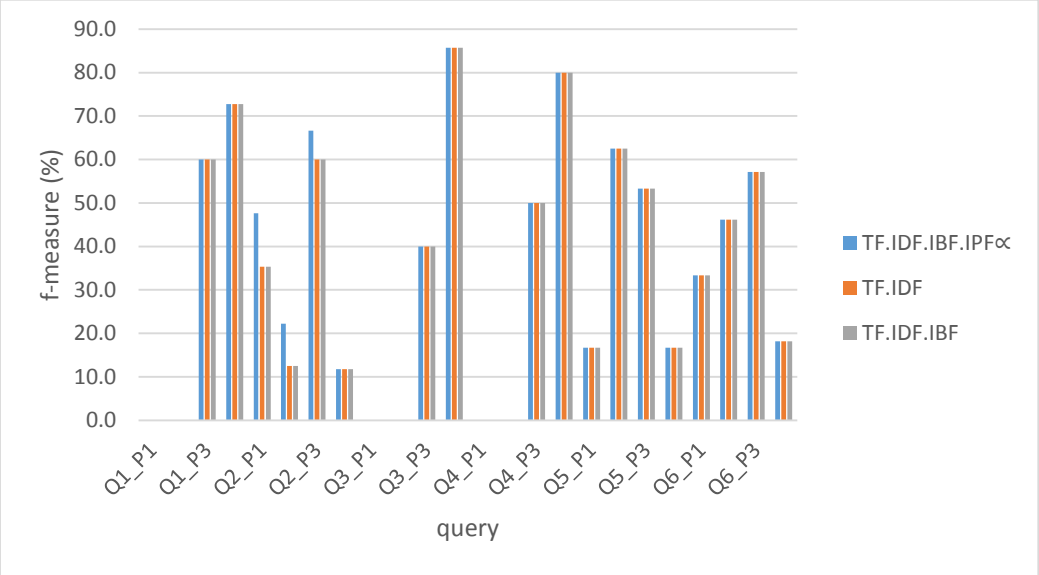
Gambar 4.24 menunjukkan perbandingan nilai *precision* antara metode TF.IDF.IBF.IPF∝ dengan metode TF.IDF dan TF.IDF.IBF. Dari gambar tersebut

diketahui bahwa metode TF.IDF dan TF.IDF.IBF memiliki nilai *precision* yang lebih tinggi dibandingkan dengan metode TF.IDF.IBF.IPF∝, yaitu pada 11 query yang digunakan, sedangkan 7 query lainnya memiliki nilai *precision* yang sama. Hal ini dikarenakan pada metode TF.IDF.IBF.IPF∝ meskipun dokumen relevan namun jika tidak sesuai dengan preferensi pengguna maka dokumen tersebut akan ditempatkan pada urutan yang lebih bawah dibandingkan dengan dokumen relevan dan sesuai dengan preferensi pengguna.



Gambar 4.24 Perbandingan *Precision*

Pada **Gambar 4.25** ditunjukkan perbandingan nilai *f-measure* dari semua input dengan metode TF.IDF, TF.IDF.IBF, dan TF.IDF.IBF.IPF∝. Berdasarkan gambar tersebut diketahui bahwa terdapat 3 input dengan metode TF.IDF.IBF.IPF∝ yang memberikan nilai *f-measure* lebih tinggi dibandingkan dengan metode TF.IDF dan TF.IDF.IBF. Sedangkan untuk 15 input lainnya metode TF.IDF.IBF.IPF∝ memberikan hasil nilai *f-measure* yang sama dengan dua metode lainnya. Tiga input tersebut adalah Q2_P1 dengan nilai 12.3% lebih tinggi, Q2_P2 dengan nilai 9.7% lebih tinggi, dan Q2_P3 dengan nilai 6.7% lebih tinggi dibandingkan dengan dua metode lain.



Gambar 4.25 Perbandingan *F-measure*

Berdasarkan perbandingan nilai *recall*, *precision*, dan *f-measure* antara metode TF.IDF.IBF.IPF$_{\propto}$ dengan metode TF.IDF dan TF.IDF.IBF maka dapat disimpulkan bahwa metode yang diajukan lebih unggul dalam melakukan perangkingan dokumen berdasarkan preferensi pengguna dibandingkan dengan dua metode lain. Hal ini pun tampak dari rata-rata *f-measure* ketika metode tersebut (lihat **Tabel 4.4**). Metode TF.IDF.IBF.IPF$_{\propto}$ memiliki rata-rata *f-measure* sebesar 46.7% sedangkan dua metode lain bernilai 45.1%.

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.IBF.IPF$_{\propto}$ dapat diaplikasikan pada perangkingan dokumen berbahasa Arab yang sesuai preferensi pengguna dengan nilai *recall*, *precision* dan *f-measure* mencapai 75%, 100%, dan 85.7% untuk query Q3 dengan pilihan preferensi pengguna P4.
2. Pada metode TF.IDF.IBF.IPF$_{\propto}$, nilai $\propto$ memberikan pengaruh tingkat preferensi pengguna untuk perangkingan dukumen. Semakin besar nilai $\propto$ maka dokumen yang termasuk pada preferensi pengguna akan berada pada peringkat yang lebih atas. Nilai $\propto$ optimal yang menempatkan dokumen relevan sesuai preferensi pengguna pada urutan atas dan tetap menampilkan dokumen relevan dari selain preferensi pengguna adalah saat $\propto = 0.9$.

5.2 Saran

Beberapa saran setelah dilakukan penelitian ini adalah sebagai berikut:

1. Preferensi pada penelitian ini berdasarkan preferensi tunggal. Oleh karena itu dapat dilakukan penelitian lebih lanjut untuk memenuhi kebutuhan multi preferensi.
2. Kesesuaian hasil pencarian juga ditentukan oleh *query* pengguna. Perlu adanya *query* expansion untuk meningkatkan kemampuan pencarian.

DAFTAR LAMPIRAN

LAMPIRAN 1

Ground Thruth Uji Coba

Ground Thruth Uji Coba (Query 1 sampai 4)

No	Query	Dokumen Relevan		
		Judul Buku	Dokumen ke-	Metodologi Fiqih
1	استعمال آنية الفضة حكمها	(المهذب، 7)	6 dan 7	(3)Syafi'iyah
		(مختصر المزني، 8)	1	(3)Syafi'iyah
		(الإقناع في فقه الإمام أحمد بن حنبل، 11)	1 dan 2	(4)Hanabilah
		(المبدع في شرح المقنع، 11)	5 dan 7	(4)Hanabilah
2	الماء الذي يبس والذي السجين	(المبسوط، 1)	7	(1)Hanafiyah
		(الجامع الصغير، 1)	5 dan 6	(1)Hanafiyah
		(مراقي الفلاح شرح نور الإيضاح، 3)	1 dan 3	(1)Hanafiyah
		(المدونة، 4)	1	(2)Malikiyah
		(بداية المجتهد ونهاية المقتصد، 5)	3	(2)Malikiyah
		(المهذب، 7)	1-6, dan 8	(3)Syafi'iyah
		(مختصر المزني، 8)	10	(3)Syafi'iyah
		(المبدع في شرح المقنع، 11)	4	(4)Hanabilah
3	الماء المسخن	(المهذب، 7)	1	(3)Syafi'iyah
		(المغني لابن قدامة، 11)	9 dan 10	(4)Hanabilah
		(الإقناع في فقه الإمام أحمد بن حنبل، 11)	1	(4)Hanabilah
4	الماء المشمس	(المهذب، 7)	1	(3)Syafi'iyah
		(المغني لابن قدامة، 11)	10	(4)Hanabilah
		(الإقناع في فقه الإمام أحمد بن حنبل، 11)	1	(4)Hanabilah

Ground Thruth Uji Coba (Query 5 dan 6)

No	Query	Dokumen Relevan		
		Judul Buku	Dokumen ke-	Metodologi Fiqih
5	الوضوء من لحوم الابل	المبسوط (1)	1	(1)Hanafiyah
		حاشية الروض المربع شرح زاد المستقنع (3)	7	(1)Hanafiyah
		المدونة (4)	7 dan 9	(2)Malikiyah
		بداية المجتهد ونهاية المقتصد (5)	2 dan 4	(2)Malikiyah
		الكافي في فقه أهل المدينة المالكي (6)	2	(2)Malikiyah
		المهذب (7)	9 dan 10	(3)Syafi'iyah
		مختصر المزني (8)	4	(3)Syafi'iyah
		الأم (9)	4	(3)Syafi'iyah
		الإقناع في فقه الإمام أحمد بن حنبل (11)	8	(4)Hanabilah
6	غسل الجنابة	المبسوط (1)	6	(1)Hanafiyah
		الجامع الصغير (1)	7	(1)Hanafiyah
		المدونة (4)	6	(2)Malikiyah
		بداية المجتهد ونهاية المقتصد (5)	5	(2)Malikiyah
		الكافي في فقه أهل المدينة المالكي (6)	4	(2)Malikiyah
		المهذب (7)	4 dan 10	(3)Syafi'iyah
		مختصر المزني (8)	7 dan 8	(3)Syafi'iyah
		الإقناع في فقه الإمام أحمد بن حنبل (11)	8	(4)Hanabilah

LAMPIRAN 2

Tabel Posisi Dokumen Relevan sesuai Preferensi Pengguna Pada Hasil Pencarian

Posisi Dokumen Relevan dan sesuai Preferensi Pengguna pada Hasil Pencarian dengan Metode TF.IDF, TF.IDF.IBF dan TF.IDF.IBF.IPF$_{\propto}$ (untuk $\propto = 0$ hingga $\propto = 0.3$)

Query	Preferensi Pengguna	**Posisi Dokumen Relevan dan sesuai Preferensi Pengguna pada Hasil Pencarian**											
		TF.IDF		TF.IDF.IBF		TF.IDF.IBF.IPF$_{\propto}$							
						0		0.1		0.2		0.3	
		A	B	A	B	A	B	A	B	A	B	A	B
Q1	P1	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-
	P3	5	18	5	18	6	16	6	12	4	10	3	8
	P4	1	9	1	10	1	12	1	10	1	10	1	10
Q2	P1	8	25	8	25	10	24	9	19	9	18	8	19
	P2	3	38	4	40	4	40	2	38	2	32	2	18
	P3	1	49	1	47	1	44	1	40	1	39	1	36
	P4	2	-	2	-	3	-	2	-	2	-	1	-
Q3	P1	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-
	P3	4	-	4	-	4	-	4	-	3	-	2	-
	P4	1	3	1	3	1	3	1	3	1	3	1	3
Q4	P1	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-
	P3	4	-	4	-	4	-	4	-	3	-	2	-
	P4	1	6	1	6	1	6	1	6	1	5	1	5
Q5	P1	7	-	8	-	10	-	9	-	7	-	4	-
	P2	1	12	1	12	1	12	1	8	1	8	1	8
	P3	2	15	2	15	2	15	2	12	1	10	1	8
	P4	14	-	14	-	13	-	11	-	8	-	6	-
Q6	**P1**	1	9	2	9	1	10	1	10	1	8	1	5
	P2	2	14	1	14	1	15	1	14	1	13	1	11
	P3	3	6	3	6	3	6	3	6	3	6	2	6
	P4	12	-	12	-	12	-	10	-	9	-	9	-

Posisi Dokumen Relevan dan sesuai Preferensi Pengguna pada Hasil Pencarian dengan Metode TF.IDF.IBF.IPF$_{\propto}$ (untuk $\propto$=0.4 hingga $\propto$= 1)

Query	Preferensi Pengguna	Posisi Dokumen Relevan dan sesuai Preferensi Pengguna pada Hasil Pencarian													
		TF.IDF.IBF.IPF$_{\propto}$													
		0.4		0.5		0.6		0.7		0.8		0.9		1	
		A	B	A	B	A	B	A	B	A	B	A	B	A	B
Q1	P1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P3	3	8	3	8	3	5	2	5	1	5	1	4	1	3
	P4	1	8	1	7	1	7	1	7	1	7	1	7	1	7
Q2	P1	6	14	4	13	4	11	3	8	3	8	3	7	3	7
	P2	1	20	1	20	1	13	1	9	1	7	1	6	1	6
	P3	1	31	1	25	1	22	1	19	1	15	1	15	1	15
	P4	1	-	1	-	1	-	1	-	1	-	1	-	1	-
Q3	P1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P3	2	-	1	-	1	-	1	-	1	-	1	-	1	-
	P4	1	3	1	3	1	3	1	3	1	3	1	3	1	3
Q4	P1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	P3	2	-	1	-	1	-	1	-	1	-	1	-	1	-
	P4	1	5	1	4	1	2	1	2	1	2	1	2	1	2
Q5	P1	4	-	4	-	4	-	2	-	2	-	2	-	2	-
	P2	1	7	1	7	1	7	1	6	1	6	1	6	1	6
	P3	1	7	1	6	1	6	1	6	1	5	1	5	1	5
	P4	4	-	3	-	3	-	2	-	1	-	1	-	1	-
Q6	**P1**	1	4	1	4	1	4	1	3	1	3	1	3	1	3
	P2	1	10	1	9	1	6	1	5	1	4	1	4	1	4
	P3	1	5	1	4	1	4	1	4	1	4	1	4	1	4
	P4	6	-	3	-	3	-	2	-	1	-	1	-	1	-

Keterangan:

A = Posisi peringkat atas

B = Posisi peringkat bawah

LAMPIRAN 3

Tabel *Recall*, *Precision*, dan *F-measure*

Recall, *Precision*, dan *F-measure* metode TF.IDF.IBF.IPF$_{\propto}$ (untuk $\propto = 0$ hingga $\propto = 0.2$)

		TF.IDF.IBF.IPF$_{\propto}$								
	$\propto$	**0**			**0.1**			**0.2**		
Query	**Preferensi Pengguna**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**
1	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	33.33	100.00	50.00	42.86	100.00	60.00	42.86	100.00	60.00
	4	57.14	100.00	72.73	57.14	100.00	72.73	57.14	100.00	72.73
2	**1**	27.27	50.00	35.29	41.67	55.56	47.62	41.67	55.56	47.62
	2	9.09	20.00	12.50	9.09	20.00	12.50	9.09	20.00	12.50
	3	54.55	66.67	60.00	58.33	70.00	63.64	58.33	70.00	63.64
	4	9.09	16.67	11.76	10.00	14.29	11.76	10.00	14.29	11.76
3	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	25.00	100.00	40.00	25.00	100.00	40.00	25.00	100.00	40.00
	4	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**
4	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	33.33	100.00	50.00	33.33	100.00	50.00	33.33	100.00	50.00
	4	66.67	100.00	80.00	66.67	100.00	80.00	66.67	100.00	80.00
5	**1**	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67
	2	45.45	100.00	62.50	45.45	100.00	62.50	45.45	100.00	62.50
	3	36.36	100.00	53.33	36.36	100.00	53.33	36.36	100.00	53.33
	4	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67
6	**1**	20.00	100.00	33.33	20.00	100.00	33.33	20.00	100.00	33.33
	2	22.22	100.00	36.36	30.00	100.00	46.15	30.00	100.00	46.15
	3	40.00	100.00	57.14	40.00	100.00	57.14	40.00	100.00	57.14
	4	10.00	100.00	18.18	10.00	100.00	18.18	10.00	100.00	18.18
Rata-Rata		**32.37**	**86.30**	**44.01**	**34.39**	**86.66**	**46.00**	**34.39**	**86.66**	**46.00**

Recall, *Precision*, dan *F-measure* metode TF.IDF.IBF.IPF$_{\propto}$ (untuk $\propto = 0{,}3$ hingga $\propto = 0.6$)

		TF.IDF.IBF.IPF$_{\propto}$											
	$\propto$	**0.3**			**0.4**			**0.5**			**0.6**		
Query	**Preferensi Pengguna**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**
1	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	42.86	100.00	60.00	42.86	100.00	60.00	42.86	100.00	60.00	42.86	100.00	60.00
	4	57.14	100.00	72.73	57.14	100.00	72.73	66.67	80.00	72.73	66.67	80.00	72.73
2	**1**	41.67	55.56	47.62	41.67	55.56	47.62	41.67	55.56	47.62	41.67	55.56	47.62
	2	10.00	16.67	12.50	18.18	28.57	22.22	18.18	28.57	22.22	18.18	28.57	22.22
	3	58.33	70.00	63.64	18.18	70.00	28.87	63.64	63.64	63.64	70.00	58.33	63.64
	4	11.11	12.50	11.76	58.33	12.50	20.59	12.50	11.11	11.76	16.67	9.09	11.76
3	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	25.00	100.00	40.00	25.00	100.00	40.00	25.00	100.00	40.00	25.00	100.00	40.00
	4	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**
4	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	33.33	100.00	50.00	33.33	100.00	50.00	33.33	100.00	50.00	33.33	100.00	50.00
	4	66.67	100.00	80.00	66.67	100.00	80.00	66.67	100.00	80.00	66.67	100.00	80.00
5	**1**	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67
	2	45.45	100.00	62.50	45.45	100.00	62.50	45.45	100.00	62.50	50.00	83.33	62.50
	3	36.36	100.00	53.33	36.36	100.00	53.33	36.36	100.00	53.33	36.36	100.00	53.33
	4	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67	9.09	100.00	16.67
6	**1**	20.00	100.00	33.33	20.00	100.00	33.33	20.00	100.00	33.33	20.00	100.00	33.33
	2	30.00	100.00	46.15	30.00	100.00	46.15	30.00	100.00	46.15	30.00	100.00	46.15
	3	40.00	100.00	57.14	40.00	100.00	57.14	44.44	80.00	57.14	44.44	80.00	57.14
	4	10.00	100.00	18.18	10.00	100.00	18.18	10.00	100.00	18.18	11.11	50.00	18.18
Rata-Rata		**34.51**	**86.37**	**46.00**	**35.35**	**87.03**	**45.10**	**36.11**	**84.38**	**46.54**	**37.01**	**80.27**	**46.54**

Recall, Precision, dan *F-measure* metode TF.IDF.IBF.IPF$_{\propto}$ (untuk $\propto = 0.7$ hingga $\propto = 1$)

	$\propto$	0.7			0.8			0.9			1		
Query	**Preferensi Pengguna**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**	**R (%)**	**P (%)**	**F (%)**
1	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	42.86	100.00	60.00	42.86	100.00	60.00	42.86	100.00	60.00	100.00	42.86	60.00
	4	66.67	80.00	72.73	66.67	80.00	72.73	66.67	80.00	72.73	100.00	57.14	72.73
2	**1**	41.67	55.56	47.62	45.45	50.00	47.62	45.45	50.00	47.62	100.00	31.25	47.62
	2	20.00	25.00	22.22	22.22	22.22	22.22	25.00	20.00	22.22	100.00	12.50	22.22
	3	80.00	57.14	66.67	80.00	57.14	66.67	80.00	57.14	66.67	100.00	50.00	66.67
	4	20.00	8.33	11.76	25.00	7.69	11.76	50.00	6.67	11.76	100.00	6.25	11.76
3	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	25.00	100.00	40.00	25.00	100.00	40.00	25.00	100.00	40.00	100.00	25.00	40.00
	4	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**	**75.00**	**100.00**	**85.71**	**100.00**	**75.00**	**85.71**
4	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	3	33.33	100.00	50.00	33.33	100.00	50.00	33.33	100.00	50.00	100.00	33.33	50.00
	4	66.67	100.00	80.00	66.67	100.00	80.00	66.67	100.00	80.00	100.00	66.67	80.00
5	**1**	11.11	33.33	16.67	12.50	25.00	16.67	12.50	25.00	16.67	100.00	9.09	16.67
	2	50.00	83.33	62.50	55.56	71.43	62.50	71.43	55.56	62.50	100.00	45.45	62.50
	3	40.00	80.00	53.33	50.00	57.14	53.33	66.67	44.44	53.33	100.00	36.36	53.33
	4	9.09	100.00	16.67	11.11	33.33	16.67	11.11	33.33	16.67	100.00	9.09	16.67
6	**1**	22.22	66.67	33.33	22.22	66.67	33.33	25.00	50.00	33.33	100.00	20.00	33.33
	2	30.00	100.00	46.15	33.33	75.00	46.15	37.50	60.00	46.15	100.00	30.00	46.15
	3	50.00	66.67	57.14	57.14	57.14	57.14	57.14	57.14	57.14	100.00	40.00	57.14
	4	11.11	50.00	18.18	11.11	50.00	18.18	12.50	33.33	18.18	100.00	10.00	18.18
Rata-Rata		**38.60**	**72.56**	**46.71**	**40.84**	**64.04**	**46.71**	**44.66**	**59.59**	**46.71**	**100.00**	**33.33**	**46.71**

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

Abdullah, M. Husain, (1995), *Al-Wadhîh fî Ushûl sl-Fiqh*, Beirut: Darul Bayariq

Cios, Krzysztof J. Etc., (2007), *Data Mining A Knowledge Discovery Approach*, New York: Springer.

Esraa, E.A., B.L. Nagma, and M.F. Tolba, (2010), "An Efficient Rangking Module for an Arabic Search Engine," *International Journal of Computer Science and Network Security*, vol. 10, no. 2, pp. 218-226.

Fauzi, M. A. (2013), "*Term Weighting* berbasis Indeks Buku dan Kelas untuk Perangkingan Dokumen Berbahasa Arab," *Thesis*, Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Indonesia.

Fuji, R., and G.S. Mohammad, (2013), "Class-indexing-based *term* weighting for automatic text classification," *Journal of Informetrics*, vol. 3, no. 1, pp. 72-77.

Garcia, DR. E., (2006), "*The Classic Vector Space Model*". Dapat diperoleh di *http://www.miislita.com/term-vector/term-vector-3.html*

Haidar, M., and L. Andrew, (2001), "Information Retrieval from Full-Text Arabic Databases: Can Search Engines Designe for English Do the Job?," *Libri*, vol. 51, epp. 63-74.

Harrag, F., A. Hamdi-Cherif, and E. El-Qawasmeh, (2008), "Vector space model for Arabic information retrieval — application to "Hadith" indexing," *Proceedings of the First IEEE Conference on the Applications of Digital Information and Web Technologies (ICADWIT 2008) (pp. 107-112). Czech Republic: IEEE*.

Ibrahim, A. El-Khair, (2007), "Arabic Information Retrieval", *Information Science and Technology*, vol. 41, no. 1, pp. 505-533.

Larkey, Leah S., Ballesteros, Lisa, and Connell, Margaret E., (2007), "Light Stemming for Arabic Information Retrieval", *Springer Link: Text, Speech and Language Technology*, Volume 38, 2007, pp 221-243

Leah, L.S., and B. Lisa, (2002), "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis," *Special Interest Group on Information Retrieval*, pp. 275-282.

Machnik L., (2004), "Documents Clustering Techniques". *IBIZA 2004*, Annales UMCS Informatica Poland.

Maksum, A., (1965), *Al-Amtsilatut Tashrifiyyah*, Indonesia: Maktabah Asy-Syaikh Saalim bin Sa'an Nabhaan.

Manning, C.D., R. Prabhakar, and S. Hinrich, (2008), *An Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press.

Mesleh, Abdelwadood Moh'd, (2007), *"*CHI Square Feature Extraction Based SVMs Arabic Language Text Categorization System*"*, *Journal of Computer Science*, vol. 3, no. 6, pp. 430--435.

Mesleh, A., M., dan Kanaan, G., (2008), *"*Support Vector Machine Text Classification System: Using Ant Colony Optimization Based Feature Subset Selection*"*, *IEEE*, 978-1-4244-2116-9/08.

Nahrawi, Ahmad, (1994), *Al-Imâm asy-Syâfi'i fî Mazhabayhi al-Qadîm wa al-Jadîd*, Kairo: Darul Kutub

Salton, G., and C. Buckley, (1988), "Term Weighthing approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, pp. 513-523.

Salton, G., (1989), *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer*: Addison-Wesly.

Siegel, Sidney, (1997), *Statistik Nonparametrik untuk Ilmu – Ilmu Sosial. Jakarta: PT Gramedia Pustaka Utama*

Sulaiman, S., Mohamed, H., Arshad, M. R. M., Rashid, N. A., & Yusof, U. K. (2009, November). *Hajj-QAES: A Knowledge-Based Expert System to Support Hajj Pilgrims in Decision Making. In Computer Technology and Development*, 2009. ICCTD'09. International Conference on (Vol. 1, pp. 442-446). IEEE.

BIODATA PENULIS

Khadijah Fahmi Hayati Holle lahir di Malang pada 26 Juni 1990. Penulis yang akrab dengan sapaan ”DJ” atau ”dee” ini merupakan putri ketiga dari pasangan Ali AR Holle dan Muslimatul Umroh (Almh.). Penulis telah menempuh pendidikan formal di SDN Lesanpuro IV Malang (1996-2002), MTs N Malang 2 (2002-2005), MAN 3 Malang (2005-2008) dan S1 Teknik Informatika UIN Maulana Malik Ibrahim Malang (2008-2012). Setelah mendapat gelar Sarjana, Penulis sempat menjadi dosen di STT STIKMA Internasional Malang sebelum akhirnya pada tahun 2013 diterima sebagai mahasiswa Pascasarjana Teknik Informatika ITS. Selain kegiatan akademik, Penulis juga aktiv di kegiatan kerohaniaan dan pada beberapa kesempatan Penulis menjadi pengisi acara training remaja, pembicara di beberapa radio, dan MC pada salah satu program televisi swasta lokal. Penulis dapat dihubungi melalui email dee.jafa@gmail.com, media sosial facebook http://www.facebook.com/dee.elizzah, ataupun twitter @KhadijahHolle.