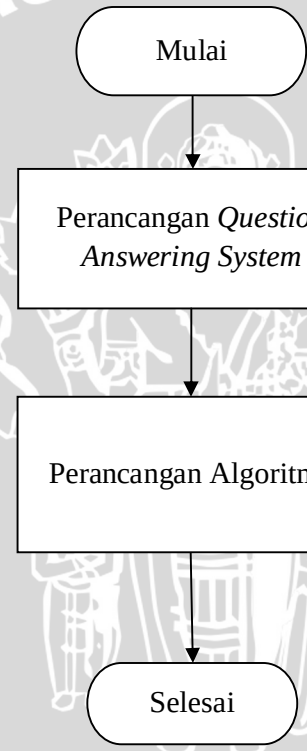
BAB IV

PERANCANGAN

Bab ini menjelaskan perancangan *Question Answering System* dengan menggunakan metode *Pattern Based Approach* sebagai produk akhir skripsi. Perancangan *Question Answering System* ini terdiri dari dua bagian yaitu Perancangan *Question Answering System* dan Perancangan Algoritma seperti yang dapat dilihat pada gambar diagram alir pada Gambar 4.1.



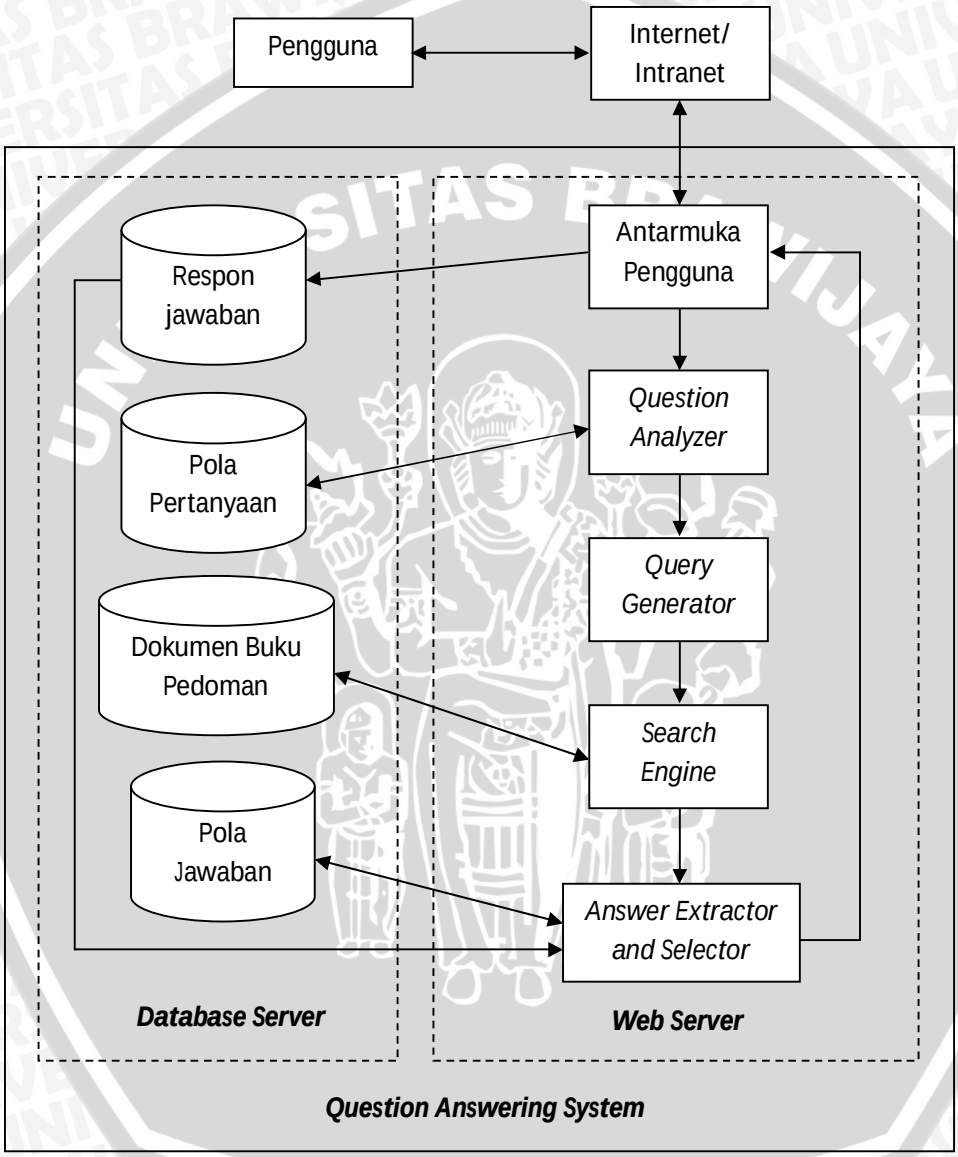
Gambar 4.1 Diagram Alir Perancangan

Sumber: Perancangan

4.1 Perancangan *Question Answering System*

Perancangan Question Answering System terbagi menjadi 3 bagian, yaitu *System Plan*, Arsitektur Sistem dan Perancangan *database.*

4.1.1 *System Plan Question Answering System*



Gambar 4.2 *System Plan Question Answering System*

Sumber: Perancangan

Pada Bagian ini dijelaskan *system plan* dari *Question Answering System* yang dirancang. Gambaran system plan *Question Answering System* yang dikembangkan dapat dilihat pada gambar 4.2

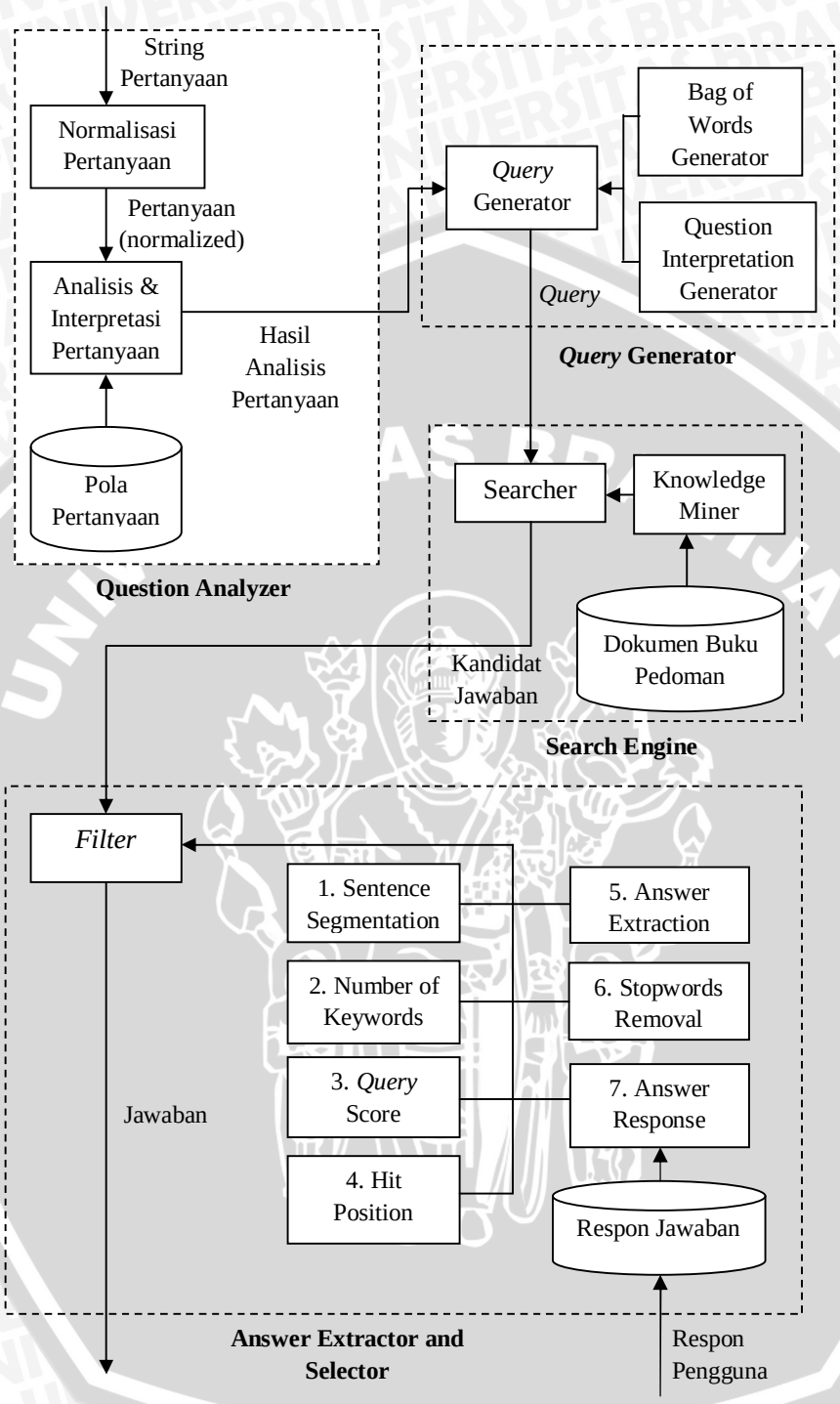
Question Answering System yang dirancang merupakan sistem yang berbasis *web* dan terhubung pada suatu *database*. Pertama pengguna memasukkan pertanyaan pada antarmuka pengguna yang berbasis *web*. Kemudian pertanyaan dilanjutkan pada *Question Analyzer*. *Question Analyzer* mengakses *database* untuk mendapatkan pola pertanyaan yang tersimpan. Dengan pola pertanyaan yang didapat dilakukan interpretasi pada pertanyaan. Hasil interpretasi pada *Question Analyzer* dilanjutkan ke *Query Generator* untuk dihasilkan *query* yang digunakan pada pencarian dokumen. Pencarian dokumen pada *Search Engine* dilakukan dengan mencari dokumen yang terdapat pada dokumen buku pedoman yang terdapat pada *database* untuk didapatkan dokumen yang relevan. Selanjutnya dokumen yang relevan diproses pada *Answer Extractor and Selector*. Pada *Answer Extractor and Selector* ini diakses pola jawaban pada *database* untuk dapat mengekstrak jawaban dari dokumen. Jawaban yang dihasilkan kemudian ditampilkan pada pengguna melalui antarmuka pengguna.

4.1.2 Arsitektur *Question Answering System*

Pada Bagian ini dijelaskan arsitektur dari sistem yang dirancang. Gambaran Arsitektur *Question Answering System* yang dikembangkan dapat dilihat pada Gambar 4.3. Arsitektur Question Answering System terdiri dari 4 komponen utama, yaitu *Question Analyzer, Query Generator, Search Engine* dan *Answer Extractor and Selector*.

1. *Question Analyzer*

Question Analyzer menerima *input* dari pengguna berupa *string* pertanyaan. *Input* berupa *string* pertanyaan harus diolah terlebih dahulu di dalam *Question Analyzer*. Pengolahan tersebut terbagi dalam dua tahap, pertama dengan melakukan normalisasi pada *string* pertanyaan, dan kemudian dilanjutkan dengan analisis terhadap pertanyaan dengan menggunakan pola pertanyaan.



Gambar 4.3 Arsitektur Sistem

Sumber: Perancangan

A. Normalisasi Pertanyaan

Tahap-tahap normalisasi pertanyaan adalah sebagai berikut:

1.) Menghilangkan tanda baca (koma, titik, tanda tanya, dsb).
Proses menghilangkan semua tanda baca serta karakter atau simbol-simbol lainnya dan hanya akan menyisakan huruf dan angka saja pada *string* pertanyaan.

2.) Mengganti huruf kapital dengan huruf kecil.
Proses ini mengganti semua huruf kapital yang terdapat pada *string* pertanyaan dengan huruf kecilnya.

3.) Mengganti beberapa kata dengan padanan katanya.
Proses ini adalah dengan mengubah kata atau bagian dari pertanyaan dengan padanan katanya, baik itu sinonim, singkatan, maupun bentuk kata bakunya.

Tahap normalisasi ini dilakukan agar setiap kata atau frase pada pertanyaan yang dimasukkan dapat dikenali dan diproses oleh sistem.

B. Analisis & Interpretasi Pertanyaan

Pada bagian ini pertanyaan yang telah melalui proses normalisasi dianalisis dan diinterpretasikan dengan pola pertanyaan. Hasil interpretasi pertanyaan ini adalah berupa properti, target dan konteks dari pertanyaan.

Daftar pola pertanyaan yang yang telah disiapkan pada sistem dapat dilihat pada lampiran 2. Beberapa contoh bentuk pola pertanyaan itu sendiri adalah sebagai berikut:

- `(dimana) letak <T> <C>`
- `(dimana) <C> <T>`
- `(dimana) <T> <C> berada`
- `(dimanakah) letak <T> <C>`
- `(dimanakah) <T> <C> berada`

Pola pertanyaan diatas adalah contoh pola pertanyaan yang mewakili tipe properti lokasi atau tempat. Kata didalam tanda kurung mewakili kata tanya untuk pertanyaan. *Tag* `<T>` mewakili target sedangkan `<C>` mewakili konteks. Target dan konteks serta tipe properti yang menyertai inilah yang menjadi hasil dari interpretasi pertanyaan.

Dari proses tersebut masih memungkinkan sebuah pertanyaan memiliki lebih dari satu interpretasi, karena \bisa saja pertanyaan cocok dengan beberapa pola pertanyaan. Untuk itu dari beberapa pola tersebut dicarilah pola pertanyaan yang paling spesifik, yaitu dengan menghitung berapa kata pada pola yang cocok dengan kata pada pertanyaan. Hasil interpretasi dari pola pertanyaan yang paling spesifik inilah yang dipakai pada proses selanjutnya.

2. *Query Generator*

Untuk dapat menghasilkan jawaban dari pertanyaan, sistem perlu mencari dokumen-dokumen yang relevan dengan pertanyaan yang diajukan. *Query Generator* berfungsi untuk menghasilkan *query* yang diperlukan dalam pencarian dokumen pada bagian *Search Engine*. Dalam menghasilkan *query* ini digunakan dua macam metode, yaitu *Bag of Words* dan *Question Interpretation Generator*.

A. *Bag of Words*

Metode ini menggunakan pertanyaan yang sudah dinormalisasi pada proses sebelumnya. Pada pertanyaan tersebut kemudian dihilangkan kata tanya. Selanjutnya dihilangkan kata-kata *stopword* yang terdapat pada pertanyaan. Kata-kata yang tersisa inilah yang akan menjadi kata kunci (*keyword*) dalam pencarian dokumen.

B. *Question Interpretation Generator*

Question Interpretation Generator menggunakan hasil interpretasi yang diperoleh dari *Question Analyzer*, yaitu target dan konteks dari pertanyaan. Target dan konteks dari pertanyaan inilah yang menjadi *keyword* dalam pencarian dokumen.

Kelebihan *Bag of Words* dibandingkan *Question Interpretation* adalah akan mendapatkan hasil pencarian yang lebih banyak, akan tetapi kekurangannya adalah lebih besar kemungkinan hasil pencarian tersebut tidak sesuai dengan yang seharusnya dicari. Sedangkan *Question Interpretation* memiliki keunggulan pada presisi atau ketepatan hasil pencarian yang diperoleh jika dibandingkan dengan *Bag of Words,* meskipun data yang diperoleh mungkin tidak sebanyak pada *Bag of Words.*

3. *Search Engine*

Search Engine mencari dokumen-dokumen yang relevan dengan jawaban yang diinginkan dari pertanyaan dengan menggunakan *query* yang dihasilkan *Query Generator*. Dokumen-dokumen yang diperoleh akan menjadi kandidat jawaban dan dokumen-dokumen ini masih perlu melalui beberapa proses sebelum didapatkan jawaban yang diinginkan.

Hasil dari ketiga proses ini akan tersimpan pada tabel pada *database* dan disertai frekuensi kemunculan setiap katanya. Frekuensi inilah yang akan menentukan dokumen mana yang lebih relevan dan lebih tinggi tingkat ketepatannya.

Dalam proses pencarian dokumen, sistem akan mendaftar dokumen mana saja yang memiliki kata yang terdapat pada kata kunci atau *keyword* dari hasil *query generator*. Tidak semua *keyword* harus ada pada dokumen-dokumen ini, bisa saja suatu dokumen hanya memiliki satu *keyword* saja. Sedangkan pada dokumen yang memiliki beberapa *keyword*, frekuensi pada setiap *keyword* yang ditemukan pada dokumen akan dijumlahkan dan akan menjadi skor pada dokumen tersebut untuk menentukan tingkat ketepatan dokumen.

4. *Answer Extractor and Selector*

Dokumen-dokumen hasil pencarian pada proses sebelumnya akan digunakan dalam penentuan kandidat jawaban pada proses ini. Kandidat jawaban harus melewati beberapa *filter* dan *scoring* untuk dapat menentukan jawaban yang paling tepat. *Filter* yang digunakan antara lain:

a. *Sentence Segmentation*

Dokumen hasil *Search Engine* dipecah menjadi bentuk kalimat-kalimat. Proses ini hampir mirip dengan proses *parsing*, tetapi bedanya dokumen-dokumen yang ada dipisahkan menjadi bentuk kalimat.

b. *Number of Keywords*

Proses ini membuang kalimat kandidat jawaban yang jumlah *keyword* dalam kalimat kandidat jawaban tidak memenuhi threshold atau suatu nilai

minimum tertentu. Hal ini dilakukan untuk membuang kalimat-kalimat yang kurang relevan dengan pencarian jawaban yang diinginkan.

c. *Query Score*

Proses ini memberikan skor kepada kalimat sesuai dengan *query* yang digunakan untuk mendapatkan dokumen. *Query* yang dimaksud adalah *Bag of Words* dan *Question Interpretation Generator* yang sebelumnya dijelaskan pada bagian *Search Engine*.

d. *Hit Position*

Proses ini memberikan skor kepada kandidat jawaban sesuai dengan total frekuensi *keyword* yang ditemukan pada dokumen darimana kalimat itu berasal.

e. *Answer Extraction*

Proses ini mengekstrak kata-kata atau bagian dari kalimat yang memenuhi kondisi sebagai jawaban. Untuk dapat mengekstrak jawaban digunakan pola-pola jawaban yang sebelumnya sudah disiapkan. Pola jawaban yang tersedia tercantum dalam lampiran 3.

f. *Stopwords Removal*

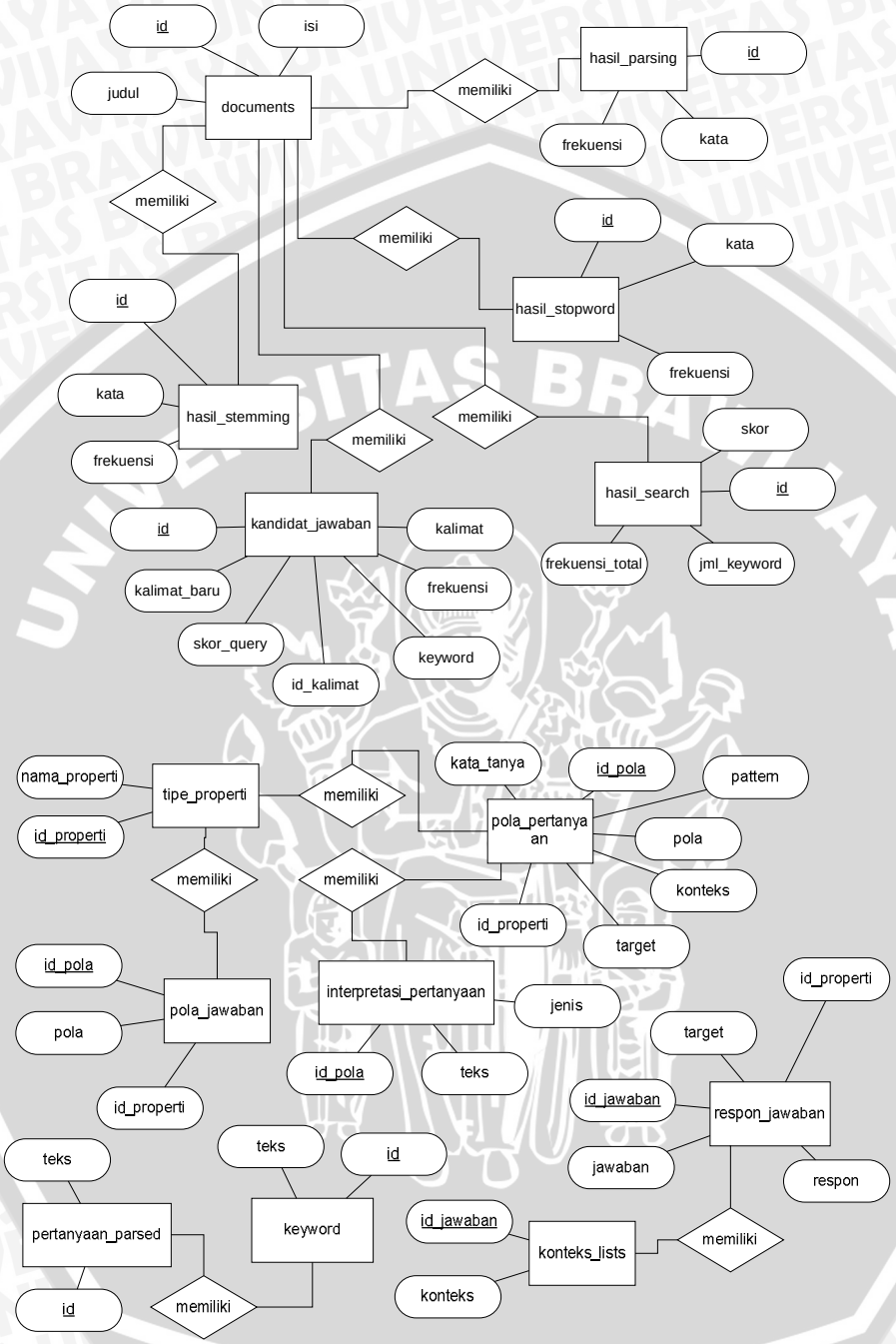
Proses ini membuang kata-kata yang termasuk dalam *stopword* yang terdapat pada kandidat jawaban. Daftar kata *stopword* tercantum dalam lampiran 1.

g. *Answer Response*

Proses ini memberikan atau bahkan mengurangi skor dari sebuah kandidat jawaban sesuai dengan respon dari pengguna yang sebelumnya telah menggunakan sistem ini.

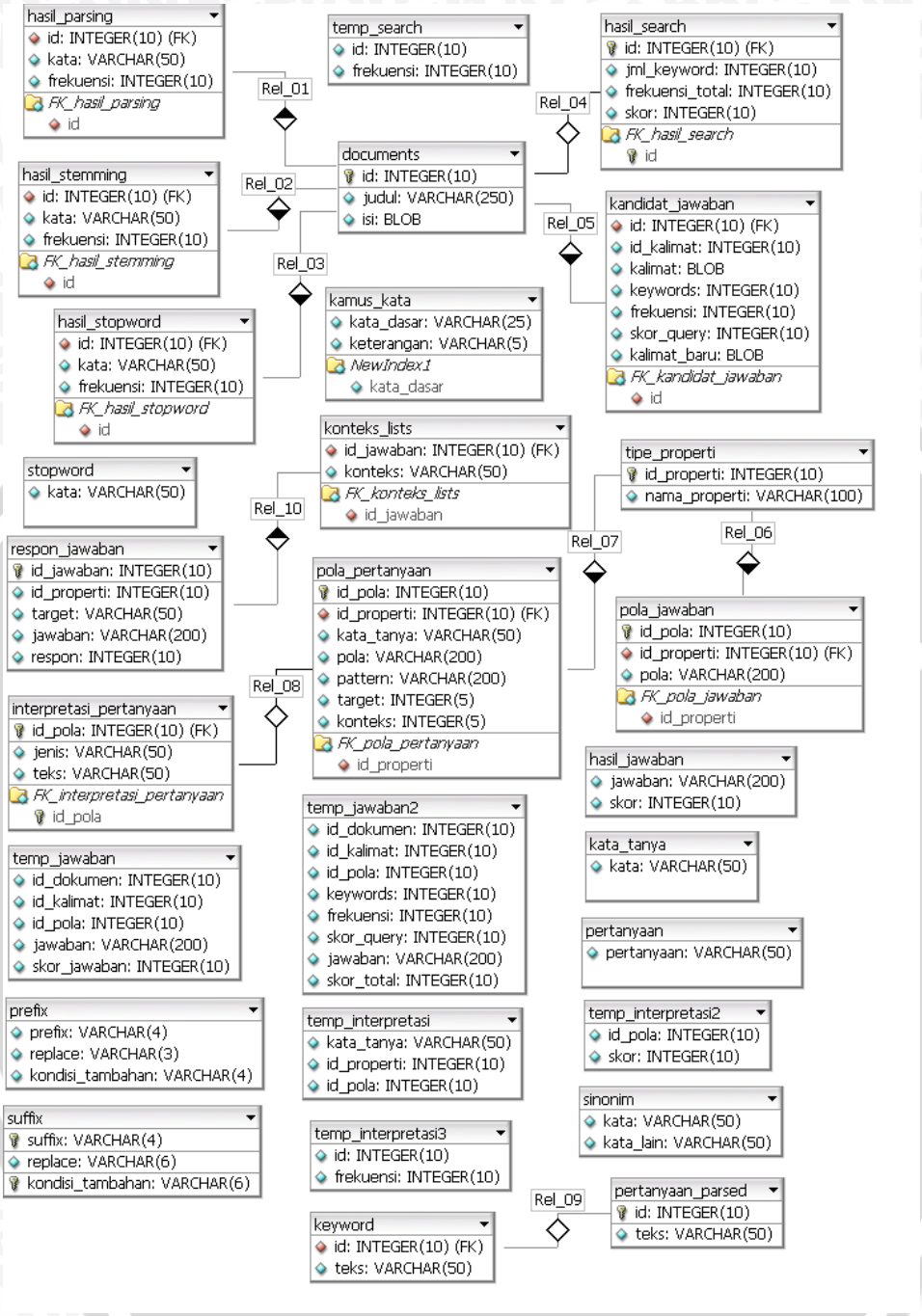
4.1.3 Perancangan *Database*

Question Answering System ini menggunakan *database* dalam menyimpan dan memproses data-data yang diperlukan dalam menghasilkan jawaban. Adapun perancangan tabel *Entity Relationship Diagram* basis data sistem ini diperlihatkan pada Gambar 4.4.



Gambar 4.4 *ERD Question Answering System*
Sumber: Perancangan

Implementasi dari *ERD* pada Gambar 4.4 dapat dilihat dalam bentuk *physical diagram* pada Gambar 4.5.



Gambar 4.5 *Physical Diagram database*

Sumber: Perancangan

4.2 Perancangan Algoritma

Pada bagian ini dijelaskan mengenai perancangan algoritma *Pattern Based Approach* yang digunakan dalam *Question Answering System*. Perancangan

algoritma ini meliputi bagaimana pengumpulan data-datanya, serta perancangan algoritma pada tiap modul-modulnya.

4.2.1 Pengumpulan Data

Pengumpulan data dilakukan untuk memperoleh data-data yang dibutuhkan oleh sistem, yang meliputi dokumen, pola pertanyaan, dan pola jawaban. Bagian ini menjelaskan bagaimana data didapatkan dan disimpan ke dalam *database*.

1. Dokumen

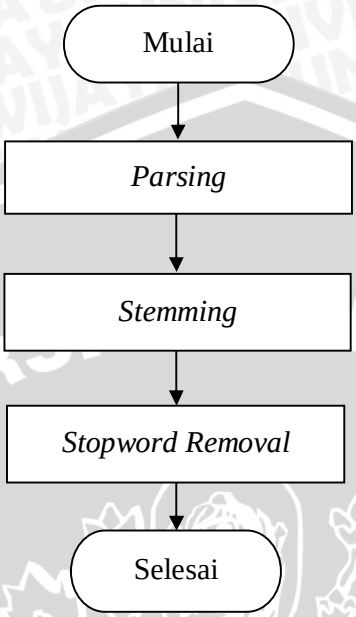
Dokumen yang digunakan dalam sistem ini berasal dari Buku Pedoman Pendidikan. Buku Pedoman yang digunakan adalah berupa *softcopy* Buku Pedoman dengan *form*at *pdf*. Dokumen-dokumen ini dimasukkan ke dalam *database* baik secara *manual* maupun secara otomatis dengan menggunakan beberapa *tools*.

Penggunaan *tools* dipilih supaya proses memasukkan data ke dalam *database* dapat dilakukan dengan lebih cepat. Data yang dimasukkan dengan menggunakan *tools* antara lain data mengenai mata kuliah yang memuat kode mata kuliah, beban studi dari mata kuliah, dan pustaka yang digunakan dalam mata kuliah tersebut. *Tools* yang digunakan berupa aplikasi berbentuk *php* dimana pengguna mengisikan *form* yang terdapat padanya dan *tools* secara otomatis memasukkan isian tersebut ke dalam dokumen seusai dengan bentuk dan *form*at yang diinginkan.

Proses memasukkan data secara manual juga dilakukan karena beberapa dokumen tidak memungkinkan untuk dimasukkan secara otomatis ataupun dengan menggunakan *tools*. Dokumen-dokumen tersebut meliputi dokumen yang berupa gambar, tabel, maupun diagram alir, dan dokumen-dokumen lain yang bukan berupa teks. Dalam memasukkan dokumen-dokumen tersebut, pertama dokumen diubah ke dalam bentuk teks secara manual dan kemudian dimasukkan ke dalam *database*.

Ketika semua dokumen telah dimasukkan ke dalam *database*, proses selanjutnya adalah dengan melakukan *preprocessing* pada dokumen tersebut. Tahap *preprocessing* ini meliputi *parsing*, *stemming*, dan *stopword removal* yang

dilihat pada diagram alir pada Gambar 4.6. Tahap *preprocessing* ini dilakukan untuk mempermudah dalam proses pencarian dokumen.



Gambar 4.6 Diagram Alir *Preprocessing* Dokumen

Sumber: Perancangan

a. *Parsing* adalah proses memisahkan dokumen menjadi bentuk kata. Dalam parsing ini yang digunakan untuk memisahkan antara satu kata dengan kata yang lain adalah spasi atau *whitespace* yang terletak diantara kata-kata. Kata-kata tersebut juga akan dihitung frekuensi kemunculannya di dalam dokumen. Kata dan frekuensi hasil dari proses ini akan tersimpan di dalam tabel yang memuat hasil parsing.

b. *Stemming* adalah proses menghilangkan imbuhan. Proses *stemming* ini diberlakukan pada kata-kata hasil *parsing* pada proses sebelumnya. Imbuhan yang terdapat pada kata terdiri dari *prefix* (awalan) dan *suffix* (akhiran). Langkah-langkah dalam proses *stemming* ini:

   i. Membandingkan kata yang akan dilakukan *stemming* dengan kata yang terdapat pada tabel yang berisikan kamus kata dasar dalam bahasa Indonesia. Jika kata tersebut ada di dalam tabel kata dasar maka tidak perlu dilakukan *stemming*, sedangkan jika tidak ada maka proses *stemming* dilanjutkan ke langkah berikutnya.

ii. Membandingkan awalan dan akhiran dengan daftar awalan dan akhiran yang terdapat pada tabel yang memuat *prefix* dan tabel yang memuat *suffix.*

iii. Awalan dan akhiran yang sesuai dengan isi tabel akan dihilangkan atau digantikan dengan huruf atau karakter tertentu sehingga kata berimbuhan menjadi kata dasar.

c. *Stopword removal* adalah proses yang membuang kata-kata yang termasuk dalam *stopword*. Proses ini dilakukan dengan membandingkan kata dengan tabel yang berisi daftar kata yang termasuk ke dalam *stopword*. Jika kata tersebut ada dalam tabel *stopword*, maka kata akan dibuang. Daftar kata *stopword* terdapat pada lampiran 1.

2. *Question Patterns*

Pola Pertanyaan (*Question Patterns*) diperoleh dari kumpulan sampel-sampel pertanyaan yang lazim digunakan. Dengan menggunakan sampel pertanyaan tersebut, pola pertanyaan dibentuk secara manual berdasarkan delapan tipe properti yang suda ditentukan. Tipe-tipe properti tersebut antara lain:

- *PEOPLE* (Orang)
- *TIME* (Waktu)
- *LOCATION* (Tempat)
- *ORGANIZATION* (Organisasi)
- *MEASURE* (Ukuran)
- *COUNT* (Angka)
- *OBJECT* (Obyek)
- *OTHER* (Lain)

Tahap-tahap yang dilalui dalam pengembangan setiap pola pertanyaan antara lain:

a. Menentukan kata tanya untuk setiap pola pertanyaan. Untuk dapat menentukan kata tanya dapat dilakukan dengan membandingkan dengan tabel yang memuat seluruh kata tanya pada *database.* Contoh:
Sampel pertanyaan “Berapakah kode mata kuliah Data Mining?”

Dengan membandingkan sampel pertanyaan dengan tabel yang memuat daftar kata tanya diperoleh kata tanya dari pertanyaan ini adalah "berapakah". Kemudian ditentukan bahwa pertanyaan ini mewakili properti *COUNT* atau angka.

b. Menentukan dimana letak kata atau posisi kunci dari pertanyaan, yaitu target <*T*> dan konteks <*C*> dari pertanyaan. Dari sampel pertanyaan kita ubah kata yang mewakili target dengan <T> dan kata yang mewakili konteks dengan <C>. Contoh: Dari sampel pertanyaan sebelumnya yang ingin ditanyakan adalah *"kode"* dari *"Data Mining"*. Oleh karena itu ditentukan *"kode"* adalah sebagai target sedangkan *"Data Mining"* sebagai konteksnya. Didapatkan:
"Berapakah <T> mata kuliah <C>?"

c. Menentukan kata atau frase lain yang dapat memberikan arti khusus bagi sebuah tipe pertanyaan. Contoh: Pada sampel pertanyaan sebelumnya, frase yang memiliki arti khusus bagi pertanyaan adalah "mata kuliah". Dari hasil pemrosesan sampel pertanyaan tersebut maka diperoleh pola pertanyaan dengan bentuk:

```
(berapakah) <T> mata kuliah <C>
```

Pola-pola pertanyaan yang didapatkan dari proses ini kemudian disimpan ke dalam sebuah tabel pada *database* yang memuat kumpulan pola-pola pertanyaan dari semua tipe properti.

3. *Answer Patterns*

Pola pertanyaan (*Answer Patterns*) dapat dibentuk secara otomatis yaitu melalui dua tahap, yaitu *Pattern Extraction* dan *Pattern Assessment*.

a. *Pattern Extraction*

Pada Bagian ini disiapkan kumpulan pertanyaan yang disertai dengan jawaban sebagai *data training*. Pertanyaan tersebut kemudian di-*input*-kan ke sistem dan pertanyaan akan mengalami pemrosesan seperti yang dijelaskan sebelumnya, mulai dari *question analyzer, query generator, search engine,* sampai pada *answer extractor and selector.*

Hasil interpretasi dari *question analyzer* digunakan untuk membentuk *query* untuk *search engine,* yaitu dengan menggunakan target dan konteks dari

pertanyaan disertai dengan jawaban dari pertanyaan pada training set. Pada contoh pertanyaan *"Dimanakah letak Universitas Brawijaya?"* dengan jawaban "kota Malang" akan didapat hasil interpretasi sebagai berikut:

- Properti: *LOCATION*
- Target: universitas
- Konteks: brawijaya

Dari hasil interpretasi pertanyaan tersebut kemudian dibentuklah sebuah *query* untuk pencarian dokumen yang terdiri dari target, konteks, dan jawaban dari pertanyaan. *Query* yang didapat adalah sebagai berikut:

```
"universitas" "brawijaya" "kota Malang"
```

Query digunakan pada *search engine* untuk memperoleh potongan-potongan teks yang sesuai. Kata atau frase yang mengandung target, konteks, dan jawaban (properti) digantikan dengan tag <*T*>, <*C*>, dan <*P*>. Selanjutnya Pola jawaban diekstrak.

Contoh kasusnya jika dari hasil pemrosesan didapatkan potongan teks sebagai berikut *"Universitas Brawijaya terletak di Kota Malang."* maka pola jawaban yang diekstrak dari potongan teks tersebut:

```
<T> <C> terletak di <P>
```

Pola yang diekstrak ini kemudian disimpan dalam sebuah tabel *database* dan termasuk dalam tipe properti *LOCATION*.

b. *Pattern Assessment and Filtering*

Bagian ini berfungsi untuk menilai pola yang didapatkan dari *Pattern Extraction* layak untuk digunakan atau tidak. Pada bagian ini juga digunakan kumpulan pertanyaan yang disertai dengan jawabannya sebagai *data training*. Pertanyaan juga di-*input*-kan ke dalam sistem dan akan melalui tahap-tahap pemrosesan pertanyaan. Pola jawaban yang terbentuk pada proses sebelumnya diuji pada tahap ini.

Pengujian dilakukan dengan mencatat seberapa sering sebuah pola jawaban dipakai dalam menjawab pertanyaan. Kemudian dilakukan penghitungan untuk setiap pola pertanyaan sebagai berikut:

Untuk setiap pola pertanyaan a, akan dicatat seberapa sering pola ini dapat digunakan untuk mengekstrak jawaban yang benar *(#correct$_a$)* dan seberapa sering didapatkan jawaban yang salah *(#incorrect$_a$)*. Kemudian untuk setiap tipe properti *p*, jumlah potongan teks juga dicatat *(# snippets$_p$)*. Nilai ini digunakan untuk menghitung pengukuran berikut [TOB-10]:

$$Confidence_a = \frac{\#correct_a}{\#correct_a + \#incorrect_a}$$

$$Support_a = \frac{\#correct_a}{\#snippets_p} \quad (4\text{-}1)$$

Dimana:

- *#correct$_a$* : banyaknya jawaban yang benar yang diekstrak dari pola *a*
- *#incorrect$_a$* : banyaknya jawaban yang salah yang diekstrak dari pola *a*
- *# snippets$_p$* : jumlah potongan teks untuk tipe properti *p*

Kemudian ditentukan nilai threshold untuk nilai minimal dari *confidence* dan *support* untuk menentukan suatu pola pertanyaan layak digunakan atau tidak. Pola dengan nilai *confidence* yang lebih rendah dibandingkan nilai *threshold* dibuang. Begitu pula pola dengan nilai *support* yang lebih rendah dari *threshold* juga akan dibuang. Sedangkan pola yang memenuhi nilai *threshold* akan tetap disimpan di dalam *database*. Pola yang masih tersimpan inilah yang digunakan sebagai pola jawaban pada *Question Answering System.*

4.2.2 Algoritma pada *Question Analyzer*

Bagian ini menjelaskan tentang algoritma yang akan diterapkan pada *Question Analyzer*. Pertama, dilakukan normalisasi pada *string* pertanyaan yaitu dengan menghilangkan semua tanda baca dan karakter lain sehingga hanya tersisa huruf dan angka saja. Kemudian proses selanjutnya dilanjutkan dengan mengubah huruf kapital menjadi huruf kecil. Kedua proses ini dapat dilakukan dengan operasi yang terdapat pada *php*. Untuk menghilangkan tanda baca atau karakter

lain digunakan perintah *preg_replace* dan *regular expression* yang dapat menghilangkan tanda baca dan karakter tertentu, sedangkan untuk mengubah ke huruf kecil digunakan perintah *strtolower*.

Proses selanjutnya adalah dengan mengubah kata atau bagian dari pertanyaan dengan padanan katanya, baik itu sinonim, singkatan, maupun kata bakunya. Hal ini dilakukan untuk mengupayakan agar setiap kata pada pertanyaan dapat dikenali oleh sistem. Untuk itu diperlukan sebuah tabel khusus yang memuat seluruh padanan kata, sinonim, singkatan, atau bentuk baku kata. Setiap kata pada pertanyaan akan dibandingkan dengan isi tabel tersebut, dan jika ditemukan maka kata pada pertanyaan tersebut akan langsung digantikan dengan kata yang terdapat pada tabel. Contohnya, jika pada pertanyaan terdapat kata "*ptiik*" maka sistem akan mengubahnya menjadi "*program teknologi informasi dan ilmu komputer*".

Proses selanjutnya adalah *Question Analyzer*, yaitu dengan menganalisis pertanyaan dengan membandingkan dengan pola pertanyaan untuk mendapatkan target dan konteks dari pertanyaan. Langkah-langkah yang ditempuh dalam proses ini antara lain:

1. Pertama dicari dahulu kata tanya yang terdapat pada pertanyaan yaitu dengan membandingkannya dengan tabel yang berisi kumpulan kata tanya.
2. Jika kata tanya sudah didapatkan langkah selanjutnya adalah dengan mencari pola pertanyaan yang memiliki kata tanya yang sama dengan pertanyaan.
3. Dari pola pertanyaan yang didapatkan tersebut, kemudian dicari pola yang paling mirip dengan pertanyaan dengan membandingkan tiap *string* katanya.
4. Jika sudah didapatkan pola yang paling mirip kemudian kita dapatkan tipe propertinya dan kita ambil dan simpan kata-kata yang mewakili target dan konteks. Dalam hal ini konteks bisa saja terdiri dari lebih dari satu kata, sedangkan target hanya boleh terdapat satu saja.

Contohnya kasusnya jika terdapat sebuah pertanyaan "*Dimanakah letak UB?*"

Pertama-tama dilakukan normalisasi pada pertanyaan tersebut:

b.) Menghilangkan tanda baca, hasilnya: "*Dimanakah letak UB"*

c.) Mengganti huruf kapital dengan huruf kecil, hasilnya: "*dimanakah letak ub"*

d.) Mengganti dengan padanan katanya: "*dimanakah letak universitas brawijaya"*

Dari pertanyaan hasil normalisasi dicari kata yang menjadi kata tanyanya, dan didapatkan kata "dimanakah" sebagai kata tanya. Kemudian pertanyaan dibandingkan dengan pola pertanyaan yang memiliki kata tanya "dimanakah" untuk dicari pola yang paling mirip dengan pertanyaan. Dari proses ini ditemukan pola pertanyaan yang mirip adalah:

- `(dimanakah) letak <T> <C>`
- `(dimanakah) <C> <T>`

Kedua pola itu memiliki kata tanya yang sama dengan pertanyaan dan juga memiliki pola yang mirip pula dengan pertanyaan. Untuk itu dicarilah pola yang paling spesifik dengan mencari jumlah kata yang sama. Pada pola pertama memiliki dua kata yang sama, yaitu kata "dimanakah" dan kata "letak". Sedangkan pada pola kedua hanya satu kata yang sama, yaitu kata "dimanakah" saja. Sehingga ditentukan Pola pertama yang digunakan untuk interpretasi.

Selanjutnya adalah dengan mengambil kata yang mewakili `<T>` dan `<C>` sebagai target dan konteks dari pertanyaan. Sedangkan pola yang digunakan berasal dari properti lokasi. Sehingga didapatkan

- Properti: *LOCATION* (tempat)
- Target: universitas
- Konteks: brawijaya

Hasil interpretasi suatu pertanyaan bisa saja menghasilkan lebih dari satu konteks. Sedangkan target hanya boleh terdapat satu saja dalam sebuah pertanyaan. Pada contoh kasus di atas sebenarya konteksnya adalah semua kata yang mengikuti atau terletak setelah kata yang yang menjadi target, tetapi berhubung kata target hanya diikuti oleh satu kata maka hanya didapatkan satu konteks saja.

Berikut *Pseudocode* yang pada *Question Analyzer* dapat dilihat pada Gambar 4.7.

```
1   input pertanyaan
2   remove tanda_baca from pertanyaan
3   pertanyaan to lowercase
4   replace words in pertanyaan with synonyms
5
6   get kata_tanya_pertanyaan from pertanyaan
7   for each pola_pertanyaan in pola_pertanyaan table do
8       if kata_tanya_pola=kata_tanya_pertanyaan
9           if pola_pertanyaan like pertanyaan
10              get properti
11              get target
12              get konteks
```

Gambar 4.7 *Pseudocode Question Analyzer*

Sumber : Perancangan

4.2.3 Algoritma pada *Search Engine*

Pencarian dokumen pada sistem ini menggunakan dua metode, yaitu *Bag of Words* dan *Question Interpretation Generator*. Untuk metode *Bag of Words*, langkah-langkahnya adalah sebagai berikut:

1. Pertama dilakukan *parsing* pada *string* pertanyaan sehingga menjadi beberapa kata.
2. Kemudian dilakukan *stemming* pada kata hasil parsing sehingga kata tersebut menjadi bentuk kata dasarnya.
3. Setelah *stemming* kemudian dari kata-kata tersebut dibuang kata yang termasuk dalam kata tanya dan *stopword*, yaitu dengan membandingkannya dengan tabel yang berisi kata tanya dan tabel yang berisi *stopword*.
4. Kata-kata yang tersisa inilah yang kemudian digunakan sebagai *keyword* atau kata kunci dalam pencarian dokumen.

Untuk metode *Question Interpretation Generator* digunakan target dan konteks dari hasil interpretasi pertanyaan sebagai kata kunci untuk pencarian dokumen.

Pencarian dokumen dilakukan dengan membandingkan kata kunci (*keyword*) yang sudah diperoleh dengan kata-kata dari dokumen yang sudah melalui proses *parsing*, *stemming*, dan *stopword* yang dijelaskan pada bagian pengumpulan data sebelumnya. Untuk setiap kata kunci yang ditemukan dalam dokumen dihitung juga frekuensi kemunculan katanya. Frekuensi dari kata kunci

tersebut kemudian dijumlahkan untuk setiap dokumennya untuk dapat menentukan dokumen mana yang lebih relevan. Hasil pencarian inilah yang akan digunakan sebagai kandidat jawaban pada proses selanjutnya.

Untuk setiap dokumen yang didapatkan dari proses *Bag of Words* diberikan skor sebanyak 1, sedangkan untuk hasil *Question Interpretation Generator* diberikan skor sebanyak 2. Hal ini dikarenakan metode *Question Interpretation Generator* lebih akurat dan spesifik dalam hasil pencariannya. Pemberian skor ini nantinya akan dipergunakan dalam proses *Answer Extractor and Selector* yang memang membutuhkan penilaian atau *scoring* pada setiap filternya.

Berikut *Pseudocode* yang pada *Search Engine* dapat dilihat pada Gambar 4.8.

```
1   parsing pertanyaan into pertanyaan_parsed
2   stemming pertanyaan_parsed
3   remove kata_tanya from pertanyaan_parsed
4   remove stopwords from pertanyaan_parsed
5
6   insert pertanyaan_parsed into keyword table of database
7
8   search document that have words similar to keywords
9
10  search document that have words similar to target and konteks
11
12  get search result
```

Gambar 4.8 *Pseudocode Search Engine*

Sumber : Perancangan

4.2.4 Algoritma pada *Answer Extractor and Selector*

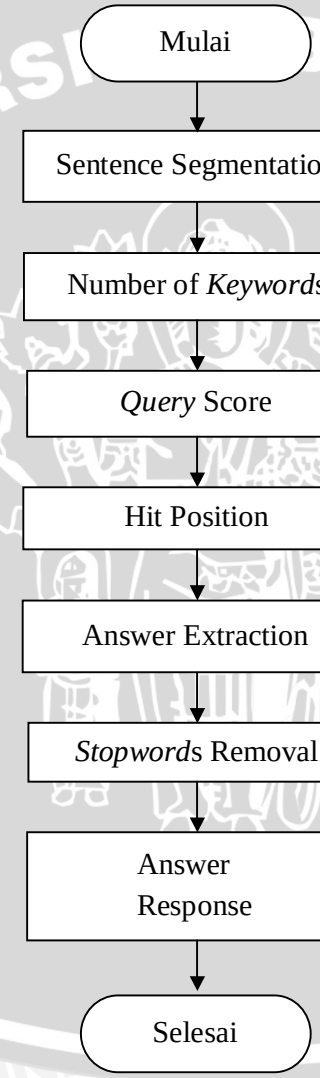
Pada bagian ini, kandidat jawaban melalui beberapa filter dan pemberian skor untuk dapat menentukan jawaban yang paling tepat. *Filter-filter* yang digunakan antara lain *Sentence Segmentation, Number of Keywords, Query Score, Hit Position, Answer Extraction, Stopwords Removal*, dan *Answer Response* seperti yang terlihat pada Gambar 4.9.

a. *Sentence Segmentation*

Kandidat jawaban yang masih berupa dokumen dipisahkan menjadi bentuk kalimat-kalimat. Pada proses memisahkan kalimat ini, yang digunakan sebagai penanda atau batas antara satu kalimat dengan kalimat yang lain

adalah tanda titik, tanda tanya, dan tanda seru. Proses yang dilakukan ini hampir sama dengan proses parsing pada *preprocessing* dokumen, hanya saja hasil dari *Sentence Segmentation* ini adalah berupa kalimat.

Selanjutnya tiap kalimat yang dihasilkan akan dimasukkan ke dalam sebuah tabel *database* yang memuat semua kalimat serta skor yang dimiliki kalimat tersebut. Pada proses ini, kalimat-kalimat tersebut kita sebut sebagai kalimat kandidat jawaban yang baru dan diberi skor awal 0.



Gambar 4.9 Diagram Alir Proses *Answer Extractor and Selector*

Sumber: Perancangan

b. *Number of Keywords*

Pada bagian ini dihitung jumlah *keyword* yang ada pada setiap kalimat akan dihitung. Yang dimaksud dengan *keyword* disini adalah kata kunci yang digunakan untuk pencarian dokumen pada proses sebelumnya. Kemudian jumlah *keyword* yang ditemukan di dalam kalimat dimasukkan ke dalam persamaan sebagai berikut [SCH-05]:

$$\boldsymbol{M \geq \lfloor \sqrt{K-1} \rfloor + 1} \quad (4\text{-}2)$$

Dimana:

- *K* = jumlah kata kunci atau *keyword* pada pertanyaan
- *M* = jumlah kata kunci atau *keyword* yang juga terdapat pada kalimat yang menjadi kandidat jawaban.

Kalimat-kalimat kandidat jawaban akan dibuang jika jumlah *keyword* yang terdapat pada kandidat jawaban tidak memenuhi kondisi di atas. Jika memenuhi kondisi ini, sebuah kalimat kandidat jawaban akan mendapat skor berdasarkan nilai *M*. Agar perbedaan skor tidak terlampau jauh maka perlu dilakukan normalisasi pada pemberian skor yaitu dengan skala skor antara 0 sampai 10. Perhitungan skor adalah sebagai berikut:

$$\boldsymbol{skor = \frac{M}{K}\ x\ 10} \quad (4\text{-}3)$$

c. *Query Score*

Pemberian skor ini berdasarkan dari jenis *query* yang digunakan untuk mendapatkan dokumen. Pada bagian ini dokumen yang diperoleh dari metode *Bag of Words* akan mendapatkan skor 1, sedangkan dokumen yang diperoleh dari metode *Question Interpretation* skor yang didapatkan adalah 2. Hal ini dikarenakan *Question Interpretation* memiliki presisi yang lebih tinggi dan lebih spesifik dibandingkan dengan metode *Bag of Words.*

d. *Hit Position*

Pada bagian ini skor yang diberikan pada kandidat jawaban adalah sesuai dengan total frekuensi kata kunci atau *keyword* yang muncul pada dokumen tempat kalimat kandidat jawaban berasal. Karena frekuensi total

pada setiap dokumen bisa saja sangat berbeda jauh, dimana suatu dokumen bisa saja memiliki frekuensi total sampai ratusan sedangkan dokumen lain bisa saja hanya memiliki 1 frekuensi total, maka perlu dilakukan normalisasi pada skor yang diberikan agar perbedaan skor antar dokumen tidak terlampau jauh. Skor yang diberikan pada bagian ini diberikan dengan skala 0 sampai 10. Untuk mendapatkan skor tersebut dilakukan perhitungan sebagai berikut:

$$\boldsymbol{skor = \frac{Fa - Fmin}{Fmax - Fmin}\ x\ 10} \quad (4\text{-}4)$$

Dimana:

- *Fa* = frekuensi kata kunci pada dokumen *a*
- *Fmax* = frekuensi kata kunci tertinggi dari seluruh dokumen
- *Fmin* = frekuensi kata kunci tertinggi dari seluruh dokumen

e. *Answer Extraction*

Pada bagian ini, kata-kata yang mewakili target dan konteks hasil interpretasi pertanyaan pada proses *Question Analyzer* akan digantikan dengan tag <*T*> untuk target, dan <*C*> untuk konteks. Proses ini dapat dilakukan dengan perintah *str_replace* yang terdapat pada *php*.

Selanjutnya kalimat kandidat jawaban ini dibandingkan dengan pola jawaban. Berikut beberapa pola jawaban yang mewakili tipe properti lokasi:

- `<T> <C> terletak di <P>`
- `<T> <C> ada di <P>`
- `<T> <C> berada di <P>`
- `<T> <C> adalah <P>`
- `<T> <C> bernama <P>`

Pada pola pertanyaan tersebut *tag* <*P*> mewakili potongan kalimat yang nantinya akan kita ekstrak dan akan menjadi jawaban dari pertanyaan.

Untuk dapat melakukan proses diatas, pertama kita cari dulu semua pola jawaban yang memiliki tipe properti yang sesuai dengan hasil interpretasi pertanyaan. Dari semua pola jawaban yang didapatkan, kemudian dicari pola yang paling mirip dengan kalimat kandidat jawaban. Untuk dapat menentukan

kemiripan dapat dilakukan dengan membandingkan posisi <T>, posisi <C>, serta kalimat yang menyertai *tag* <T> dan <C> tersebut.

Setelah didapatkan pola yang paling mirip, kemudian diekstrak potongan kalimat yang mewakili <P> pada kalimat kandidat jawaban. Potongan kalimat ini dapat berupa satu kata atau beberapa kata. Potongan kalimat inilah yang kemudian menjadi kandidat jawaban yang baru. Kandidat jawaban yang baru ini ini mewarisi skor dari kalimat kandidat jawaban sebelumnya.

Jika dari hasil pengekstrakan beberapa kalimat ditemukan potongan kalimat yang sama maka potongan kalimat tersebut akan menjadi satu kandidat jawaban saja. Sedangkan skornya merupakan penjumlahan dari skor tiap kandidat.

f. *Stopwords Removal*

Filter ini diterapkan pada semua kandidat jawaban dan membuang kandidat jika *string* jawaban mengandung kata *stopword* di dalamnya. Proses yang dilakukan pada bagian ini mirip dengan *stopword* removal pada *preprocessing*, yaitu dengan membuang kata pada kandidat jawaban yang terdapat pada tabel *stopword*.

g. *Answer Response*

Bagian ini yang membedakan *Question Answering System* yang dirancang dengan *OpenEphyra*. Untuk setiap jawaban yang dihasilkan, sistem meminta pengguna memberikan respon mengenai hasil jawaban yang dikeluarkan. Respon dari pengguna ini akan menentukan skor yang diberikan pada bagian ini, yaitu sebagai berikut:

- Respon “jawaban sesuai” akan memberikan skor +1
- Respon “ragu-ragu/tidak tahu” akan memberikan skor +0
- Respon “jawaban tidak sesuai” akan memberikan skor -1

Skor-skor pada bagian ini akan terus terakumulasi dan tersimpan di dalam tabel *respon_jawaban* untuk setiap jawaban yang sama dan berasal dari properti, target dan konteks yang sama pula.

Jawaban akhir yang ditampilkan kepada pengguna adalah kandidat jawaban yang memiliki skor tertinggi setelah melalui semua proses *filtering* dan *scoring* tersebut. Jika terdapat kandidat jawaban yang memiliki skor yang sama, maka yang ditampilkan adalah kandidat jawaban yang potongan kalimatnya lebih pendek.

Berikut *Pseudocode* yang pada *Search Engine* dapat dilihat pada Gambar 4.10.

```
parse search result into sentences
insert parsed search result into kandidat_jawaban

for each kandidat_jawaban do
     get number of keyword(M) in kandidat_jawaban
     if M < √(K-1)+1 do
          remove kandidat_jawaban
     else
          give score to kandidat_jawaban = M/K*10
          give query score to kandidat_jawaban
          give score to kandidat_jawaban =(Fa-Fmin)/(Fmax-
          Fmin)*10

          replace word in kandidat_jawaban where word=target
          with <T>
          replace word in kandidat_jawaban where word=konteks
          with <C>

          for each answer pattern where
          properti=answer_pattern.properti do
               for each kandidat_jawaban where
               kandiat_jawaban like answer_pattern do
                    extract jawaban
                    remove stopwords from jawaban
                    insert jawaban into hasil_jawaban table

print hasil_jawaban with most score
```

Gambar 4.10 *Pseudocode Answer Extractor and Selector*

Sumber : Perancangan

4.2.5 Contoh Implementasi Kasus

Berikut contoh dari perhitungan algoritma *Pattern Based Approach* yang diimplementasikan pada *Question Answering System*. Pada contoh kasus ini pengguna memasukkan pertanyaan “Apakah prasyarat mata kuliah Data Mining?”. Berikut Proses yang berjalan pada sistem:

1. ***Question Analyzer***
   a. Normalisasi pertanyaan

- Menghilangkan tanda baca: “Apakah prasyarat mata kuliah Data Mining”
- Mengganti huruf kapital dengan huruf kecil: “apakah prasyarat mata kuliah data mining”
- Mengganti dengan padanan katanya: “apakah prasyarat mata kuliah data mining”

b. Analisis/Interpretasi Pertanyaan

- Dicari kata tanya pada pertanyaan dan didapatkan kata tanya “apakah”.
- Dilakukan pencarian pada pola pertanyaan dengan kata tanya “apakah” dan memiliki pola yang mirip dengan pertanyaan. Didapatkan pola pertanyaan “apakah <T> mata kuliah <C>” dengan properti *OBJECT*.
- Diekstrak kata-kata yang mewakili target <T> dan konteks <C> dan didapatkan:
    - Target : “prasyarat”
    - Konteks : “data” dan “mining”

2. *Query Generator & Search Engine*

a. *Bag of Words*

- Parsing pertanyaan yang telah dinormalisasi, didapatkan kata-kata sebagai berikut: “apakah”, “prasyarat”, “mata”, “kuliah”, “data”, “mining”.
- Stemming pada kata-kata: “apakah”, “prasyarat”, “mata”, “kuliah”, “data”, “mining”, “syarat”.
- Membuang kata tanya dan stopword: “prasyarat”, “mata”, “kuliah”, “data”, “mining”, “syarat”.

Sehingga kata kunci untuk metode *bag of words* adalah “prasyarat”, “mata”, “kuliah”, “data”, “mining”, dan ”syarat”.

b. *Question Interpretation Generator*

Kata kunci untuk metode ini adalah target dan konteks hasil interpretasi sehingga didapatkan kata kunci sebagai berikut: “prasayarat”, “data”, “mining”.

3. *Answer Extractor and Selector*

a. *Sentence Segmentation*

Dokumen hasil pencarian dipisahkan mejadi kalimat-kalimat. Didapatkan sebuah dokumen: *"Kode Mata Kuliah Data Mining adalah IFK15032. <kal>Beban Studi Mata Kuliah Data Mining adalah 3 SKS. <kal> Prasyarat Mata Kuliah Data Mining adalah Basis Data."*

Tag <kal> yang terdapat pada dokumen digunakan sebagai pemisah antara satu kalimat dengan kalimat yang lain. *Tag* <kal> ini ditambahkan secara manual pada dokumen ketika proses pemgumpulan dokumen. Setiap string yang dipisahkan dengan <kal> akan menjadi satu kalimat sehingga dokumen tersebut akan terpecah menjadi 3 kalimat, yaitu

- Kalimat 1: *"Kode Mata Kuliah Data Mining adalah IFK15032."*
- Kalimat 2: *"Beban Studi Mata Kuliah Data Mining adalah 3 SKS."*
- Kalimat 3: *"Prasyarat Mata Kuliah Data Mining adalah Basis Data."*

Setiap kalimat itu menjadi kandidat jawaban dan memiliki skor awal 0.

b. *Number of Keywords*

Jumlah *keyword* (*M*) pada setiap kandidat jawaban dihitung sehingga didapatkan:

- Pada Kalimat 1, $M = 4$
- Pada Kalimat 2, $M = 4$
- Pada Kalimat 3, $M = 5$

Jumlah keyword pencarian (*K*) adalah 6, sehingga ketiga kalimat memenuhi kondisi pada persamaan (4-2). Maka perhitungan skor untuk setiap kalimat kandidat jawaban didapatkan:

- Skor Kalimat 1 = $\frac{4}{6} \boldsymbol{x}\ \mathbf{10} = \mathbf{6.67} \approx \mathbf{7}$
- Skor Kalimat 2 = $\frac{4}{6} \boldsymbol{x}\ \mathbf{10} = \mathbf{6.67} \approx \mathbf{7}$
- Skor Kalimat 3 = $\frac{5}{6} \boldsymbol{x}\ \mathbf{10} = \mathbf{8.33} \approx \mathbf{8}$

c. *Query Score*

Dikarenakan kalimat kandidat jawaban berasal dari dokumen yang sama maka kalimat kandidat jawaban memiliki skor yang sama, yaitu 2 karena dokumen dapat diperoleh melalui metode *Question Interpretation Generator*.

d. *Hit Position*

Skor diberikan sesuai dengan frekuensi kemunculan *keyword* pada dokumen. Didapatkan frekuensi tertinggi dari hasil pencarian adalah 146, sedangkan frekuensi terkecil adalah 1. Sedangkan Frekuensi total pada dokumen yang digunakan pada contoh kasus adalah 60. Perhitungan skornya:

$$\boldsymbol{skor = \frac{60 - 1}{146 - 1}\ x\ 10 = 4{,}06 \approx 4}$$

e. *Answer Extraction*

Setiap kata yang mewakili target dan konteks pada kalimat kandidat jawaban diubah dengan <T> dan <C> sehingga:

- Kalimat 1: *"Kode Mata Kuliah <C> adalah IFK15032."*
- Kalimat 2: *"Beban Studi Mata Kuliah <C> adalah 3 SKS."*
- Kalimat 3: *"<T> Mata Kuliah <C> adalah Basis Data."*

Setiap kalimat kandidat jawaban dibandingkan dengan pola jawaban dengan tipe properti *OBJECT* dan diekstrak kata atau potongan kalimat yang akan menjadi jawaban. Pada kalimat 1 dan 2 tidak ditemukan pola jawaban yang cocok karena kalimat 1 dan 2 tidak memiliki kata yang mewakili target dari perntanyaan. Sedangkan pada kalimat 3 ditemukan pola jawaban yang cocok, yaitu *"<T> <C> adalah <P>"*. Kemudian diekstrak potongan kata yang mewakili <P>, yaitu *"basis data"*. Potongan kata inilah yang menjadi kandidat jawaban yang baru dan mewarisi skor dari kalimat 3. Sedangkan bila ada kalimat lain yang menghasilkan kandidat jawaban *"basis data"* juga maka skornya akan diakumulasikan.

Jika dijumlahkan skor yang dimiliki kalimat 3 adalah 14, yaitu 8 dari *Number of Keywords*, 2 dari *Query Score*, dan 4 dari *Hit Position*.

f. *Stopwords Removal*

Karena tidak ada kata *stopwords* pada kandidat jawaban maka tidak ada kata yang dihilangkan.

g. *Answer Response*

Answer Response memberikan skor berdasarkan respon pengguna sebelumnya yang mengajukan pertanyaan yang sama dengan pertanyaan yang sedang diproses. Karena belum ada respon dari pengguna mengenai pertanyaan yang sedang diproses ini maka tidak ada skor yang diberikan.

Sedangkan bila pertanyaan yang sama pernah diajukan dan pengguna sebelumnya memberikan respon "tidak sesuai", maka skor akan berkurang -1 untuk setiap respon "tidak sesuai". Misalkan bila ada 3 kali respon "tidak sesuai" maka skor akhir dari kandidat jawaban "basis data" adalah 11.

4. Pemilihan Jawaban

Jawaban yang ditampilkan pada pengguna adalah kandidat jawaban yang memiliki skor akhir tertinggi. Bila terdapat beberapa kandidat jawaban dengan skor akhir sama, maka akan ditampilkan kandidat jawaban yang lebih pendek, yaitu dengan jumlah karakter/huruf yang lebih sedikit.