ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - 141501

RANCANG BANGUN APLIKASI ANDROID HALAL NUTRITION FOOD MENGGUNAKAN KOMBINASI QUERY INDEPENDENT DAN QUERY DEPENDENT RANKING

DEVELOPING AN ANDROID APPLICATION FOR HALAL NUTRITION FOOD SEARCH SYSTEM USING QUERY INDEPENDENT AND QUERY DEPENDENT RANKING COMBINATION

AHMAD CHOIRUN NAJIB
NRP 5214100057

Dosen Pembimbing
Nur Aini Rakhmawati, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

Halaman ini sengaja dikosongkan

TUGAS AKHIR - 141501

RANCANG BANGUN APLIKASI ANDROID HALAL NUTRITION FOOD MENGGUNAKAN KOMBINASI QUERY INDEPENDENT DAN QUERY DEPENDENT RANKING

AHMAD CHOIRUN NAJIB
NRP 5214100057

Dosen Pembimbing
Nur Aini Rakhmawati, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS - 141501

DEVELOPING AN ANDROID APPLICATION FOR HALAL NUTRITION FOOD SEARCH SYSTEM USING QUERY INDEPENDENT AND QUERY DEPENDENT RANKING COMBINATION

AHMAD CHOIRUN NAJIB
NRP 5214100057

Supervisor
Nur Aini Rakhmawati, Ph.D

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Information Communication and Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI ANDROID HALAL NUTRITION FOOD MENGGUNAKAN KOMBINASI QUERY INDEPENDENT DAN QUERY DEPENDENT RANKING

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Akuisisi Data dan Diseminasi Informasi
Program Studi S1 Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

AHMAD CHOIRUN NAJIB
NRP: 5214100057

Surabaya, Januari 2018

PLH KEPALA
DEPARTEMEN SISTEM INFORMASI

Edwin Riksakomara, S.Kom., M.T
NIP. 196907252003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI ANDROID HALAL NUTRITION FOOD MENGGUNAKAN KOMBINASI *QUERY-INDEPENDENT* DAN *QUERY-DEPENDENT RANKING*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Analisa Data dan Diseminasi Informasi
Program Studi S1 Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

AHMAD CHOIRUN NAJIB
NRP: 05211440000057

Disetujui Tim Penguji: Tanggal Ujian: 12 Januari 2018
Periode Wisuda: Maret 2018

Nur Aini Rakhmawati, Ph.D **(Pembimbing 1)**

Faizal Johan Atletiko, S.Kom, M.T **(Penguji 1)**

Radityo Prasetianto W., S.Kom, M.Kom **(Penguji 2)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN APLIKASI ANDROID HALAL NUTRITION FOOD MENGGUNAKAN KOMBINASI QUERY INDEPENDENT DAN QUERY DEPENDENT RANKING

Nama	: AHMAD CHOIRUN NAJIB
NRP	: 5214100057
Departemen	: Sistem Informasi FTIK
Pembimbing I	: Nur Aini Rakhmawati, Ph.D

Abstrak

Pencarian informasi produk halal dapat dilakukan melalui website Lembaga Pengkajian Obat-obatan dan Kosmetika (LPPOM) MUI. Namun, informasi yang ada masih terbatas dan sulit untuk diakses oleh pengguna termasuk masyarakat umum.

Aplikasi Halal Nutrition Food merupakan sebuah aplikasi pencarian produk halal berbasis web yang menampilkan berbagai informasi produk halal secara rinci yang didapatkan dari gabungan dataset yang diintegrasikan dalam bentuk Linked Open Data. Hasil pencarian yang didapatkan pengguna berupa informasi rinci seperti nama dan bahan-bahan produk yang didapatkan melalui dokumen entitas data produk pada dataset. Namun, hasil pencarian yang ditampilkan hanya didasarkan pada dokumen produk sehingga menimbulkan keterbatasan pada hasil pencarian. Kebutuhan pengguna dalam melakukan pencarian tidak hanya terbatas pada nama produk saja, namun dapat berupa informasi bahan yang terdapat pada komposisi sebuah produk secara langsung atau informasi pada dokumen entitas lain yang terdapat pada dataset. Entity ranking merupakan salah satu metode perankingan terhadap dokumen entitas pada dataset, antara lain terdiri dari Query Independent Ranking dan Query Dependent Ranking. Query Independent Ranking memberikan skor terhadap dokumen entitas pada dataset

yang diatur dengan metode dan kriteria tertentu sebelum pencarian dilakukan. Sedangkan Query-Dependent Ranking memberikan bobot pada kueri pengguna setelah pencarian dilakukan. Kombinasi kedua metode ini dapat memberikan final-score atau skor final untuk melakukan ranking pada dokumen entitas dengan tujuan meningkatkan relevansi atau kesesuaian hasil pencarian, semakin besar nilai skor final maka dokumen entitas tersebut semakin relevan.

Android merupakan sebuah sistem operasi yang banyak digunakan pada perangkat mobile saat ini. Keunggulan adanya aplikasi android antara lain dapat menghemat bandwith dan memudahkan akses informasi, dengan adanya aplikasi android Halal Nutrition Food dengan metode sistem pencarian query independent dan query dependent ranking diharapkan dapat memudahkan pengguna dalam mengakses informasi produk halal.

Kata kunci: entity ranking, query independent ranking, query dependent ranking, linked data, aplikasi android halal nutrition food.

DEVELOPING AN ANDROID APPLICATION FOR HALAL NUTRITION FOOD SEARCH SYSTEM USING QUERY INDEPENDENT AND QUERY DEPENDENT RANKING COMBINATION

Name	: AHMAD CHOIRUN NAJIB
NRP	: 5214100057
Major	: Information Systems FTIK
Supervisor I	: Nur Aini Rakhmawati, Ph.D

Abstract

Searching for halal product information can be done through the website of the Institute For Foods, Drugs, And Cosmetics Indonesian Council Of Ulama. However, the information is still limited and difficult to access by users.

Halal Nutrition Food application is a web-based halal product search application which it results an information about various halal products. This application combines some datasets of Linked Open Data which contains many document entities. Result of search system return set of product that contain terms of the query. Users may need not only information of product but also composition of the product when searching an information in some queries. Entity ranking is one of the ranking methods in dataset such as Query Independent Ranking and Query-Dependent Ranking that get better search results relevance. Query Independent Ranking scores document entities on dataset compiled with certain criteria before the search is performed. While Query-Dependent Ranking assigns weights to the user's query after the search is done. The combination of these two methods result a final score of a document entity in order to improve the relevance of the search results. The greater the final score, the more relevant the document entity in search results.

Android is an operating system that widely used in mobile devices today. The advantages of android application platform are reduced bandwidth and ease to use. Halal Nutrition Food android application to query independent and query dependent ranking searching system allow users in finding and to accessing better relevance of halal product information search results.

Keywords: entity ranking, query independent ranking, query dependent ranking, linked data, halal nutrition food android application.

KATA PENGANTAR

Segala puji dan syukur pada Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Rancang Bangun Aplikasi Android Halal Nutrition Food Menggunakan Kombinasi *Query Independent* dan *Query Dependent Ranking*" dengan tepat waktu.

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi nyata bagi kampus Sistem Informasi, ITS, dan bangsa Indonesia.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin menyampaikan terimakasih kepada:

1. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc., Eng., Ph.D selaku dosen pembimbing penulis yang telah memberikan ide, bimbingan, saran, kritik, ilmu, dan pengalamannya yang sangat bermanfaat sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Ibu Erma Suryani, ST, MT, Ph.D selaku dosen wali penulis yang selalu membimbing dan memberikan arahan ke penulis.
3. Bapak Dr. Ir. Aris Tjahyanto, M.Kom. selaku Kepala Departemen Sistem Informasi yang telah memberikan ilmu dan pengalaman kepada penulis.
4. Seluruh dosen Departemen Sistem Informasi ITS yang telah memberikan ilmu pengetahuan dan pengalaman yang sangat berharga dan bermanfaat bagi penulis.
5. Seluruh keluarga besar saya khusunya Bapak, Ibu dan kedua Adik saya yang telah memberikan dukungan baik material

maupun non material serta semangat kepada penulis hingga akhirnya dapat menyelesaikan tugas akhir ini.

6. Rekan-rekan organisasi UKM Cinta Rebana ITS yang telah memberikan pengalaman, pelajaran berharga dan bermanfaat selama disana.
7. Rekan-rekan organisasi UKM Cinta Lab yang telah memberikan fasilitas dan kebersamaan selama pengerjaan tugas akhir.
8. Teman-teman saya Sistem Informasi angkatan 2014 (OSIRIS) yang senantiasa menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir.
9. Kakak dan adik angkatan 2012, 2013, 2014, 2015 dan 2016 yang selalu membantu dan memberikan semangat bagi penulis.
10. Serta seluruh pihak-pihak lain yang tidak dapat disebutkan satu per satu yang telah banyak membantu penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.

Tugas Akhir ini merupakan persembahan bagi penulis untuk kedua orang tua dan keluarga besar yang selalu memberikan motivasi terbaik bagi penulis untuk dapat menuntut ilmu setinggi-tingginya dan dapat meraih kesuksesan.

Tugas Akhir ini juga masih jauh dari kata sempurna, sehingga penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk perbaikan ke depan. Semoga Tugas Akhir ini dapat bermanfaat bagi perkembangan ilmu pengetahuan dan semua pihak.

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

Pada bab pedahluan ini akan membahas terkait latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi terhadap pengerjaan tugas akhir.

1.1 Latar Belakang

Agama Islam merupakan agama dengan penganut terbesar di Indonesia dengan persentase kurang lebih 88,1 persen dari total 250 juta penduduk di Indonesia [7]. Dalam agama Islam terdapat aturan-aturan yang mengikat penganutnya, salah satunya adalah produk yang dikonsumsi umat islam harus bersifat halal. Oleh karena itu pemerintah Indonesia membentuk lembaga yang berperan untuk memberikan pengawasan dan sertifikasi terhadap produk yang beredar di Indonesia baik produk makanan, minuman, obat-obatan dan kosmetik.

Lembaga Pengawasan Pangan, Obat-obatan, dan Komestika Majelis Ulama Indonesia (LPPOM MUI) merupakan lembaga sertifikasi halal yang melakukan pemeriksaan/audit, penetapan fatwa, dan menerbitkan sertifikat halal. LPPOM MUI bekerja sama dengan berbagai lembaga untuk melakukan fungsinya dalam melakukan sertifikasi halal. LPPOM MUI juga bertanggung jawab terhadap penyediaan informasi produk halal. Salah satu media informasi produk halal dapat di akses di http://halalmui.org melalui pencarian produk halal berdasarkan kategori tertentu.

Pada *website* Halal MUI, pengguna dapat melakukan pencarian ter-

hadap informasi produk halal yang telah didaftarkan oleh pemiliki produk dan disertifikasi oleh MUI. Hasil pencarian yang ditampilkan berupa nama produk, nomor sertifikat dan nama perusahaan yang mengajukan produknya untuk disertifikasi. Hasil pencarian tidak menampilkan informasi komposisi yang terdapat pada masing-masing produk, sehingga pengguna tidak bisa melakukan pencarian berdasarkan komposisi yang terdapat pada sebuah produk yang tidak terdaftar atau belum tersertifikasi oleh MUI.

Aplikasi Halal Nutrition Food merupakan sebuah aplikasi pencarian produk halal berbasis web yang menampilkan berbagai informasi produk halal secara rinci yang didapatkan dari gabungan dataset yang diintegrasikan dalam bentuk Linked Open Data. Dataset tersebut terdiri dari dataset produk halal, dataset zat aditif yang diintegrasikan dengan dataset dari PubChem, DBPedia dan MeSH, serta Muslim Consumer Group sebagai sumber data terkait status halal atau tidaknya komposisi zat kandungan pada produk [8]. Hasil pencarian yang didapatkan pengguna pada aplikasi ini berupa informasi singkat seperti nama, komposisi dan status halal dari komposisi produk yang didapatkan melalui entitas-entitas pada gabungan dataset.

Hasil pencarian pertama kali yang didapatkan berupa informasi singkat berdasarkan produk yaitu nama dan komposisi dari produk, informasi ini didapatkan dari entitas pada dataset produk halal. Hasil pencarian yang hanya berdasarkan produk ini menimbulkan keterbatasan pada hasil pencarian. Kebutuhan pengguna dalam melakukan pencarian tidak hanya terbatas pada produk saja, namun dapat berupa bahan yang terdapat pada komposisi sebuah produk, status halal zat aditif atau informasi pada entitas lain yang terdapat pada dataset.

Query-Independent Ranking merupakan metode pemberian skor terhadap entitas pada dataset yang diatur dengan metode dan krite-

ria tertentu sebelum pencarian dilakukan dengan kueri tertentu, sedangkan *Query-Dependent Ranking* merupakan metode yang pemberian bobot pada kueri pengguna setelah pencarian dilakukan dengan kueri tertentu [3]. Kombinasi kedua metode ini dapat memberikan fleksibilitas dan relevansi hasil pencarian. Sehingga pencarian dilakukan oleh pengguna menjadi lebih sesuai.

Android merupakan sebuah sistem operasi yang banyak digunakan pada perangkat mobile. Aplikasi Android dapat menghemat bandwith dalam pertukaran data dan memudahkan akses dalam mendapatkan informasi, sehingga dengan adanya Aplikasi Android Halal Nutrition Food dapat memudahkan pengguna/masyarakat dalam mengakses informasi produk halal.

Tugas akhir ini akan memberikan penambahan fungsi pada aplikasi Halal Nutrition Food yaitu pencarian entitas pada dataset yang terhubung pada aplikasi Halal Nutrition Food menggunakan kombinasi metode *query-independent* dan *query-dependent ranking* untuk meningkatkan relevansi hasil pencarian serta menampilkannya melalui aplikasi android Halal Nutrition Food.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, tugas akhir yang akan diajukan ini menitikbertatkan permasalahan pada beberapa hal sebagai berikut:

1. Bagaimana pengguna mendapatkan informasi komposisi gizi serta karakteristik zat pada produk halal secara relevan?
2. Bagaimana menerapkan kombinasi metode *query-independent* dan *query-dependent ranking* untuk memudahkan pencarian produk halal?

3. Bagaimana menampilkan hasil pencarian melalui aplikasi android Halal Nutrition Food?

1.3 Batasan Masalah

Batasan-batasan dalam pembuatan Usulan Tugas Akhir adalah sebagai berikut:

1. Hasil akhir dari penelitian ini adalah pengembangan aplikasi web dan android berbasis linked open data yang memiliki fitur pencarian data produk.
2. Jumlah sampel data hanya dibatasi 300 produk halal yang diperoleh dari website LPPOM MUI.
3. Sistem pencarian data aplikasi web menggunakan kombinasi metode *query-independent* dan *query-dependent ranking*

1.4 Tujuan

Tujuan dari penelitian ini adalah mengembangkan sistem pencarian produk halal dari aplikasi web yang sudah dibuat pada penelitian sebelumnya yaitu Halal Nutrition Food. Sistem pencarian akan menggunakan metode kombinasi metode *query-independent* dan *query-dependent ranking* yang akan memudahkan menemukan hasil pencarian yang relevan yaitu data produk yang telah diberi label halal oleh MUI atau informasi entitas lain yang terdapat di dataset aplikasi Halal Nutrition Food.

1.5 Manfaat

Manfaat yang akan diperoleh dengan tugas akhir ini antara lain:

1. Bagi pengguna, dapat dengan mudah mengetahui karakteristik, komposisi, dan gizi dari produk halal pada yang disetujui oleh MUI.
2. Bagi peneliti, penelitian ini akan menghasilkan perangkat lunak berbasis open linked data yang mana kedepannya bisa kebangkan menjadi lebih baik lagi.

1.6 Relevansi

Tugas akhir ini berkaitan dengan pembangunan perangkat lunak dengan metode penyimpanan dan publikasi data berupa Linked Open Data dengan pengembangan sistem pencarian kombinasi metode *query-independent* dan *query-dependent ranking*. Tugas akhir ini layak dijadikan sebagai tugas akhir pada tingkat S1, karena tugas ini memecahkan masalah yaitu dalam hal mempermudah pencarian informasi terkait produk makanan atau minuman halal beserta informasi komposisi masing-masing produk.

Selain itu tugas akhir ini berkaitan dengan mata kuliah Pemrograman Integratif, Pemorgaman Berbasis Objek, Pemrograman Berbasis Web, dan Pengantar Basis Data sehingga layak untuk dijadikan sebagai tugas akhir departemen Sistem Informasi.

Halaman ini sengaja dikosongkan

BAB 2

TINJAUAN PUSTAKA

Pada bab ini dijelaskan mengenai dasar-dasar teori yang bersumber dari buku, jurnal, artikel, dan penelitian sebelumnya, yang selanjutnya dijadikan acuan pada tugas akhir ini. Dasar teori ini akan memudahkan dan memberikan gambaran umum dalam memahami tugas akhir ini.

2.1 Penelitian Sebelumnya

2.1.1 Rancang Bangun Perangkat Lunak Linked Open Data Halal Dan Gizi Pada Makanan Dan Minuman

Penelitian ini berjudul "*Rancang Bangun Perangkat Lunak Linked Open Data Halal Dan Gizi Pada Makanan Dan Minuman*" oleh Jauhar Fatawi [8]. Peneliti membuat aplikasi berbasis web *Halal Nutrition Food* yang memberikan informasi dan rekomendasi tentang status gizi dari produk dan minuman yang halal. Penelitian ini mengintegrasikan dataset produk halal, zat aditif, dan perangkat lunak "Linked Open Data Halal dan Gizi". Dengan penelitian ini, pengguna dapat mencari informasi detail mengenai komposisi dari bahan-bahan yang terdapat pada sebuah produk baik berupa status gizi maupun status halal dari masing-masing bahan yang terdapat pada sebuah produk.

2.1.2 Peningkatan Relevansi Pencarian Produk Halal dalam Aplikasi Halal Nutrition Menggunakan Algoritma OKAPI BM25F

Penelitian ini berjudul "*Peningkatan Relevansi Pencarian Produk Halal dalam Aplikasi Halal Nutrition Menggunakan Algoritma OKAPI BM25F*" oleh Adnan Mauludin Fajriyadi [1]. Dalam penelitian ini, peneliti mengembangkan aplikasi Halal Nutrition Food dengan menambahkan fitur pencarian produk halal menggunakan algoritma OKAPI BM25F agar pencarian produk lebih relevan dan akurat. Pencarian produk halal ini terbatas pada produk, pencarian yang dilakukan oleh pengguna tidak hanya berupa produk, akan tetapi dapat berupa zat aditif atau entitas lain yang terdapat pada dataset.

Tugas akhir ini akan melanjutkan penelitian tentang aplikasi Halal Nutrition Food dengan mengembangkan pencarian entitas yang terdapat pada dataset yang terdapat pada Halal Nutrition Food, sehingga pencarian menjadi lebih relevan [3].

2.1.3 Hierarchical Link Analysis for Ranking Web Data

Penelitian ini mengusulkan metode baru dalam melakukan ranking dataset dan entitas. Metode ini diperkenalkan dengan istilah DING (Dataset Ranking) yang menggunakan model pendekatan dua layer [4]. Pada layer atas dilakukan perhitungan *dataset ranking* dan layer bawah dilakukan perhitungan *entity ranking* kemudian melakukan kombinasi keduanya untuk menghasilkan bobot ranking paling baik. Pada penelitian ini, penulis memilih metode *enitity ranking* yang akan dijadikan acuan oleh penulis dalam mengembangkan fitur pencarian aplikasi Halal Nutrition Food.

2.1.4 Sindice at SemSearch 2010

Penelitian ini membandingkan metode ranking web data yaitu *query-independent* dan *query-dependent ranking* dan mengkombinasikan keduanya untuk menghasilkan bobot terbaik yaitu final entity score [3]. Penelitian ini mendeskripsikan model pada Sindice (Semantic Web Documents Search Engine) yang digunakan untuk mendefinisikan, mengindeks dan melakukan ranking pada entities yang terdapat pada *Web of data*. Pada penelitian ini, penulis memilih metode *query-independent ranking* berupa *Weighted LinkCount* dan *query-dependent ranking* berupa *tf-ief (term frequency - inverse entity frequency)* yang akan dijadikan acuan oleh penulis sebagai metode utama dalam mengembangkan fitur pencarian pada aplikasi Halal Nutrition Food.

2.2 Dasar Teori

2.2.1 Produk Halal

Produk adalah barang dan/atau jasa yang terkait dengan makanan, minuman, obat, kosmetik, produk kimiawi, produk biologi, produk rekayasa genetik, serta barang gunaan yang dipakai, digunakan, atau dimanfaatkan oleh masyarakat. Produk Halal adalah produk yang telah dinyatakan halal sesuai dengan syariat Islam.

2.2.2 Query-Independent Ranking

Query-Independent Ranking disebut juga sebagai *static ranking* [4] merupakan salah satu skema ranking yang menggunakan fitur tertentu yang telah diatur sebelum kueri pencarian dilakukan [2]. Fitur

yang sering digunakan adalah *LinkCount*. LinkCount merupakan salah satu jenis dari metode *in-degree counting link* yang menghitung besar bobot entitas berdasarkan jumlah link yang mengarah kepada entitas tersebut *incoming-links* [10]. Jumlah link yang mengarah pada entitas j dari entitas i seperti rumus berikut :

$$r(j) = \sum_{l_{\sigma,i,j}} w\left(l_{\sigma,i,j}\right) \tag{2.1}$$

2.2.3 Query-Dependent Ranking

Query-Dependent Ranking merupakan salah satu skema ranking yang menggunakan fitur tertentu yang berdasarkan kueri pencarian dilakukan [3]. Fitur yang sering digunakan seperti penggunaan TF-IDF. Di dalam RDF, penentuan skor dari subjek atau entity (e) didapatkan melalui penyatuan skor dari skor predikat (a) dan skor objek (v) yang didapatkan dari TF-IEF ($termfrequency - inverseentityfrequency$) [5]. Adapun rumus TF-IEF seperti rumus 2.2 seperti berikut :

$$tf.ief(t,e) = \frac{n_{t,e}}{\sum_k n_{t,e}} \times log \frac{|V^E|}{1 + n_{t,e}} \tag{2.2}$$

Sedangkan skor dari subjek, predikat, dan objek didapatkan melalui rumus berikut :

$$score(q,v) = \sum_{t \in q} tf.ief(t,e) \tag{2.3}$$

$$score(q, a) = \sum_{c \in p} score(q, v) \tag{2.4}$$

$$score(q, e) = \sum_{p \in s} score(q, a) \times spread(q, e) \tag{2.5}$$

dengan

$$spread(q, e) = \frac{(|q| - |v_t/v_n|) + 1}{|q|} \tag{2.6}$$

dimana $|q|$ menunjukkan jumlah dari term yang berbeda pada query q dan $|v_t/v_n|$ menunjukkan jumlah pembagian antara jumlah objek yang mengandung term dengan jumlah total objek yang terdapat pada sebuah entitas. Faktor normalisasi $spread$ berlaku pada term kueri untuk objek tunggal. Jika term kueri tersebar pada beberapa objek, maka faktor normalisasi $spread$ bernilai lebih kecil. Sebagai contoh jika entitas memiliki 3 objek dan kueri terbentuk dari 3 term, dan semua term terdapat pada objek tunggal, maka faktor normalisasi $spread$ sama dengan $\frac{(3-1/3)+1}{3} = 1,23$. Sebaliknya jika term tersebar pada beberapa objek yang berbeda, maka faktor normalisasi $spread$ sama dengan $\frac{(3-3/3)+1}{3} = 1$.

2.2.4 Semantic Web

Semantic Web merupakan sebuah fungsi tambahan dari sebuah web dimana memberikan cara yang lebih mudah untuk menemukan, berbagi, menggunakan kembali, dan menggabungkan informasi. Kemampuan ini dibentuk dengan menggabungkan kemampuan teknologi XML untuk membentuk tagging schemes dan RDF's (Resource Descripton Framework) sebagai pendekatan fleksible yang

mewakili data. Semantic web menyediakan format umum untuk pertukaran data. Selain itu Semantic web juga menyediakan bahasa umum untuk merekam bagaimana data berelasi dengan obyek-obyek dunia nyata, memungkinkan orang atau sebuah mesin memulai pada satu database kemudian berhubungan degan database lain dan terkonseksi satu sama lain [14].

2.2.5 Linked Data

Linked data merupakan salah satu bagian dari pembangunan web semantik. Linked data adalah sebuah pendakatan dimana menghubungkan dan membagikan data pada web. Dengan linked data sebuah website yang memiliki padanan yang sama bisa dihubungkan satu sama lain dengan menggunakan semantic queries. Sebagai contoh apabila kita ingin mendapatkan deskripsi kota surabaya, dengan menghubungkan dengan dbpedia kita tidak perlu menuliskannya lagi.
Kriteria-kriteria yang terdapat data yang dapat dihubungkan adalah sebagai berikut:

- Tersedia di internet
- Memiliki struktur data yang dapat dimengerti oleh mesin
- Tersedia dalam format non-proprietary
- Menggunakan standar dari W3C untuk open data
- Terhubung dengan sumber data lainnya di internet

2.2.6 RDF

Resource Description Framework (RDF) adalah kerangka untuk meneraangkan informasi dari sumber-sumber data. Sumber-sumber tersebut dapat berupa apapun, termasuk dokumen, orang, benda fi-

sik, dan konsep-konsep abstrak. RDF ini muncul saat ini dimana Web perlu di proses oleh aplikasi, bukan hanya ditampilkan kepada orang. RDF menyediakan framework umum untuk menginformasikan data sehingga dapat dilakukan pertukaran data antar apliaksi tanpa kehilangan makna [11].

RDF data model mirip dengan model konseptual sederhana seperti *entity relationship model* atau *class diagram*, namun paada RDF didasarkan pada pembuatan model berdasrkan pernyataan tentang sumber daya / resources (pada web) ke dalam bentuk subject-predicate-obyek. Bentuk ini dikenal dengan nama triples pada terminologi RDF. Subyek menunjukkan sumber daya / resources, predikat menunjukkan ciri-ciri atau aspek sumber daya dan menghubungkan antara subyek dan obyek. Untuk lebih jelasnya dapat dilihat ilustrasi di bawah ini:

Ahmad	minum	kopi
(subyek)	(predikat)	(obyek)

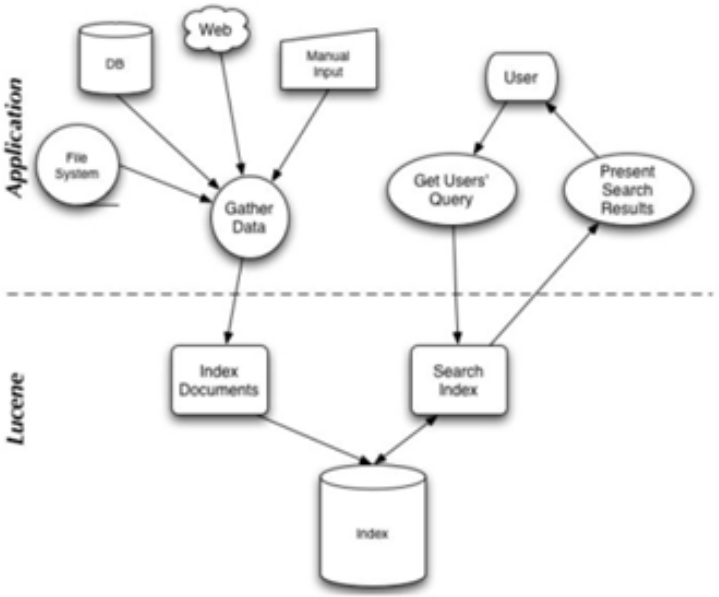
Subyek merupakan suatu hal yang dideskripsikan. Sedangkan obyek merupakan data berupa angka, string, tanggal, ataupun URI dari suatu hal atau benda lain yang memiliki hubungan dengan subjek. Predikat merupakan merupakan suatu URI yang digunakan untuk mendeskripsikan hubungan antara subjek dengan objek. URI dari predikat diambil dari vocabularies, suatu kumpulan URI yang dapat digunakan untuk merepresentasikan informasi terkait bidang tertentu [**?**]. RDF triples memiliki dua tipe, sebagai berikut:

- Literal Triples, merupakan triples dengan RDF literal berupa string, angka, atau tanggal sebagai objek. Literal triples digunakan untuk mendeskripsikan sifat / properti dari suatu hal / data.
- RDF Links, merepresentasikan hubungan antara dua sumber data. RDF links terdiri dari tiga referensi URI. URI yang di-

gunakan pada subjek dan objek untuk mengidentifikasi sumber data yang saling terkait, serta URI pada predikat untuk mendefinisikan keterkaitan antar data.

2.2.7 Apache Lucene

Apache Lucene adalah library mesin pencari teks yang memiliki kinerja tinggi dengan fitur lengkap yang seluruhnya ditulis dengan bahasa Java [9]. Teknologi ini cocok untuk hampir semua aplikasi yang memerlukan pencarian teks lengkap, terutama cross-platform. Skema Lucene dengan aplikasi dijelaskan pada Gambar 2.1



Gambar 2.1: Layer aplikasi dengan Lucene

2.2.8 Laravel

Laravel adalah salah satu *framework* pempograman PHP yang dibuat oleh Taylor Otwell. Seperti kebanyakan framework PHP lainnya, Laravel di kembangkan dengan menggunakan struktur *Model-View-Controller* (MVC). Menurut Sitepoint.com merupakan *framework* terpopular pada survey yang dilakukannya pada tahun 2015. Pada penelitian ini menggunakan *framework* ini dikarenakan beberapa alasan berikut.

1. *RESTful routing*
   RESTful adalah cara baru dalam mengelola request. Dengan REST dapat dengan mudal mengelola metode-metode request seperti GET, POST, PUT, PATCH, DELETE, STORE, dll.
2. *Composer ready*
   Composer sendiri adalah *Depedency Management PHP* yang membantu kita untuk mencari library yang akan dipakai dan menginstalnya. Dalam melakukan installasi laravel pun juga perlu menggunakan composer.
3. *Template engine*
   Templating engine adalah program yang mengubah *syntax template engine* tersebut ke HTML. Laravel sendiri memiliki *templating engine* bernama blade.
4. *Fitur lainnya*
   Selain itu laravel memiliki beberapa fitur lain seperti *SSH*, *authentication*, *aagination*, *Session*, *Schema Builder*, *Validator*, dan *Session*

Untuk lebih jelasnya dapat *framework* laravel dapat dipelajari pada laman https://laravel.com/.

2.2.9 Semantic Search

Semantic search adalah teknik data pencarian di mana permintaan pencarian bertujuan untuk tidak hanya menemukan kata kunci, tetapi untuk menentukan maksud dan makna kontekstual dari kata-kata seseorang menggunakan untuk pencarian [13]. Hasil pencarian Semantic Search memiliki makna tertentu dengan memahami frasa, menemukan hasil pencarian yang paling relevan pada repository data yang dituju. Semantic Search memperhatikan sinonim dari istilah, variasi kata, unsur bahasa alami lainnya sebagai bagian dari pencarian.

2.2.10 GSON Library

GSON merupakan sebuah library java yang berfungsi untuk melakukan serialisasi/deserialisasi dalam mengubah objek Java ke JSON atau sebaliknya [6]. GSON menyediakan metode yang sederhana untuk mengkomversi objek Java ke JSON atau sebaliknya.

2.2.11 Retrofit Android Library

Retrofit merupakan sebuah *library* Android dan Java yang berfungsi sebakai klien HTTP yang dapat melakukan operasi REST seperti GET, POST, PUT, DELETE dan HEAD [12]. Retrofit mendukung pengguna dalam melakukan koneksi dengan server dan bertukar data dengan berbagai jenis tipe data seperti JSON dan XML.

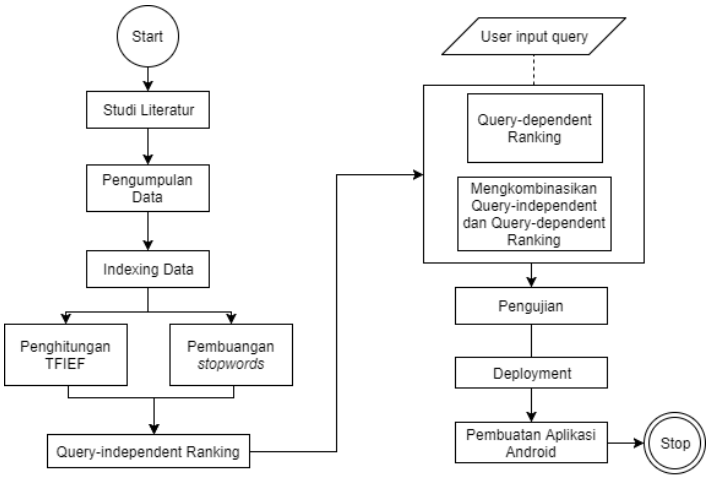
BAB 3

METODOLOGI

Pada bab metodologi akan menjelaskan bagaimana langkah pengerjaan tugas akhir dengan disertakan deskripsi dari setiap penjelasan untuk masing-masing tahapan beserta jadwal kegiatan pengerjaan tugas akhir.

3.1 Tahapan Pengerjaan Tugas Akhir

Pada sub bab ini akan dijelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi tersebut dapat dilihat pada Gambar 3.1



Gambar 3.1: Metodologi Tugas Akhir

3.1.1 Studi literatur

Pada tahap studi litertatur dilakukan pengumpulan literatur terkait dengan Linked Data, Semantic Search, Metode Perankingan khususnya metode *Query-independent* dan *Query-dependent Ranking* yang bertujuan untuk memahami konsep, teori, landasan dan teknologi yang digunakan sebagai acuan pada penelitian ini.

3.1.2 Pengumpulan data

Pengumpulan data pada penelitian ini berupa pengumpulan data produk halal. Data produk halal pada aplikasi Halal Nutrition Food berjumlah 150 entri. Pada penelitian ini ditambahkan 150 entri produk halal, sehingga keseluruhan total data menjadi 300 entri produk halal.

3.1.3 Pengindeksan data

Pada tahap ini produk halal yang tersimpan pada database masih berupa data mentah, sehingga query oleh pengguna harus diproses lagi untuk disesuaikan dengan data pada database mempermudah dalam melakukan proses pencarian. Hal tersebut dilakukan dengan tokenisasi dan pembuangan *stopwords*.

1. Pembuangan *stopwords* Pembuangan Stopwords dilakukan untuk memilah kata kunci utama dengan kata penghubung pada sebuah query oleh pengguna. Misalnya seperti kata penghubung ”pada”, ”dan”, ”di” dan kata penghubung lainnya.
2. Perhitungan TFIEF pada terms Penghitungan TFIEF dilakukan untuk menilai bobot TFIEF untuk setiap terms yang ter-

dapat pada dokumen menggunakan Apache Lucene. Penghitungan pada dokumen *turtle* menggunakan TFIEF seperti yang telah dijelaskan pada bagian 2.2.3

3.1.4 Query-Independent Ranking

Pada tahap ini dilakukan pembobotan entitas berdasarkan *LinkCount*.

(a) LinkCount
LinkCount menunjukkan seberapa populer sebuah entitas dilihat dari seberapa banyak jumlah link yang mengarah kepada sebuah entitas seperti bagian 2.2.2. Pada tahap ini penentuan skor entitas menggunakan rumus 2.1.

(b) Static Score
Static Score merupakan akumulasi dari skor LinkCount $S_s = S_l$ dimana S_l merupakan skor LinkCount.

3.1.5 Query-Dependent Ranking

Pada tahap ini dilakukan pembobotan berdasarkan term yang terdapat pada kueri pengguna. Term yang didapatkan dari kueri dibobotkan berdasarkan TF-IEF seperti bagian 2.2.3. Pembobotan TF-IEF membobotkan subjek yang berasal dari bobot skor predikat dan objek, selanjutnya bobot dari subjek ini akan mewakili bobot entitas. Pembobotan berdasarkan TF-IEF ini dihitung menggunakan rumus 2.5 yang selanjutnya akan mewakili *Query Score*.

3.1.6 Kombinasi Query-Independent dan Query Dependent Rakning

Pada tahap ini dilakukan ranking ulang terhadap entitas berdasarkan pembobotan berdasarkan perhitungan dari *query-independent* dan *query-independent ranking* yang menghasilkan skor final. Skor final akan menentukan ranking dari entitas pada pencarian dengan kueri tertentu. Skor final didapatkan dari *Query Static* S_s dan *Query Score* S_q. *Query Score* dilakukan normalisasi menggunakan logaritma log(S_q) dan *Static Score* menggunakan fungsi sigmoid [2] dengan parameter $w = 1.8$, $k = 1$ dan $a = 0.6$. Sehingga menghasilkan skor final :

$$S_f = log(S_q) + w * \frac{S_s^a}{k^a + S_s^a} \tag{3.1}$$

Contoh perhitungan rumus diatas seperti tabel 3.1 sebagai contoh berikut :

Selanjutnya, akan dilakukan kueri dengan keyword "*monosodium*". Pada tabel 3.1 didapatkan nilai-nilai yang dapat dijadikan sebagai perhitungan *query-independent* dan query-dependent.

Query-Independent: *Perhitungan LinkCount berdasarkan rumus 2.1*

Query-dependent: *Perhitungan TF-IEF berdasarkan rumus 2.5*

Kombinasi Query-Independent dan query-dependent: *Perhitungan kombinasi berdasarkan rumus 3.1*

Berdasarkan perhitungan pada tabel 3.7, maka didapatkan skor urutan ranking kueri *monosodium* pada dataset sebagai berikut.

Tabel 3.1: Contoh data pada dataset

Dok	Entitas	Subjek	Predikat	Objek
1.	Monosodium Glutamate	dbr:MnGlm	rdf:type	owl:Thing
	Monosodium Glutamate	dbr:MnGlm	foaf:name	Monosodium glutamate
2.	Happy Tos Rasa Jagung Bakar	hllv:2	hllv:contain	dbr:MnGlm
	Happy Tos Rasa Jagung Bakar	hllv:2	hllv:certif	100061230412
3.	Serena Snack	hllv:3	hllv:contain	dbr:MnGlm
	Serena Snack	hllv:3	hllv:certif	100039300306
4.	Walls Cornetto Black and White	hllv:3	hllv:contain	dbr:PtsSb
	Walls Cornetto Black and White	hllv:3	hllv:certif	290047180208

Tabel 3.3: Perhitungan *LinkCount* entitas pada dataset

Dok	LinkCount $\sum_{l_{\sigma,i,j}} w\left(l_{\sigma,i,j}\right)$	*Static Score*
1.	2	2
2.	1	1
3.	1	1
4.	1	1

Tabel 3.5: Perhitungan *Query Score* entitas pada dataset

Dok	TFIEF ($\frac{n_{t,e}}{\sum_k n_{t,e}} * log \frac{\lvert V^E \rvert}{1+n_{t,e}}$)	Query Score (TFIEF * spread(t,e))
1.	$2/3 * log \frac{4}{1+2} = 0,10034$	0,10034 * ((1-2/2)+1/1) = 0,10034
2.	$1/3 * log \frac{4}{1+1} = 0,10034$	0,10034 * ((1-1/2)+1/1) = 0,15051
3.	$1/3 * log \frac{4}{1+1} = 0,10034$	0,10034 * ((1-1/2)+1/1) = 0,15051
4.	$0/3 * log \frac{4}{1+0} = 0$	0 * ((1-1)+1/1) = 0

Tabel 3.7: Perhitungan *Final Score* entitas pada dataset

Dok	S_s	S_q	Final Score $log(S_q) + w * \frac{S_s^a}{k^a + S_s^a}$
1.	2	0,12494	$log(0,10034)+1,8*(2^{0,6})/(1+(2^{0,6})) = 0,08598$
2.	1	0,15051	$log(0,15051)+1,8*(1^{0,6})/(1+(1^{0,6})) = 0,07757$
3.	1	0,15051	$log(0,15051)+1,8*(1^{0,6})/(1+(1^{0,6})) = 0,07757$
4.	1	0	$log(0) + 1,8 * (1^{0,6})/(1 + (1^{0,6})) = 0$

Tabel 3.9: Ranking *Final Score* kueri *monosodium* pada dataset

Ranking	Dokumen	Final Score
1.	1	0,08598
2.	2	0,07757
3.	3	0,07757

3.1.7 Pengujian

Pada tahap ini dilakukan uji coba secara keseluruhan untuk mengetahui sistem kerja dalam pencarian data produk halal dan komposisi produk pada dataset apakah terdapat kesalahan atau error. Aplikasi diuji menggunakan metode sebagai berikut:

- Pengujian Fungsional, merupakan pengujian terhadap fungsionalitas dari perangkat lunak untuk memastikan fungsiona-

litas dari perangkat lunak telah berjalan. Sisi fungsional yang akan diuji menggunakan skenario tertentu, salah satunya dengan mencoba beberapa query pencarian produk dan komposisi produk. Keberhasilan ditentukan berdasarkan hasil evaluasi pengguna terhadap hasil pencarian dengan *keyword* tertentu.
Contoh beberapa skenario yang akan dilakukan adalah pada Tabel 3.11.

Tabel 3.11: Skenario Pengujian Fungsional

No.	Faktor	Detail	Ukuran
1.	Ketepatan	Dimasukkan query dengan 1 suku kata	Persentase relevansi
2.	Ketepatan	Dimasukkan query dengan n suku kata yang memfokuskan pada nama produk	Persentase relevansi
3.	Ketepatan	Dimasukkan query dengan n suku kata yang memfokuskan pada bahan-bahan produk	Persentase relevansi

- Pengujian Penambahan Stopwords, merupakan pengujian dengan memasukkan stopwords yang diambil dari beberapa term yang sering paling banyak muncul pada index lucene kedalam analyzer Apache Lucene. Kemudian dilakukan index ulang untuk menciptakan index baru dengan penggunaan stopwords dari hasil index awal, selanjutnya dari hasil index ini dilakukan pencarian yang menggunakan analyzer dengan stopwords dari index awal. Hasil pengujian berupa perbandingan analisa hasil pencarian meliputi jumlah dokumen yang

ditemukan dan perubahan skor dokumen yang ada di kedua hasil pencarian.

- Pengujian Perbandingan dengan Sistem Pencarian Sebelumnya, merupakan pengujian dengan membandingkan sistem pencarian yang menggunakan metode kombinasi *query-independent* dan *query-dependent ranking* dengan sistem pencarian sebelumnya *OKAPI BM25F*. Hasil pengujian berupa perbedaan hasil pencarian yang ditampilkan.

3.1.8 Deployment

Setelah pengujian secara fungsional berhasil, perangkat lunak akan dijalankan pada server dan aplikasi android yang telah dibuat siap untuk di publikasikan ke Google Play dan siap digunakan oleh masyarakat luas.

3.1.9 Pembuatan Aplikasi Android

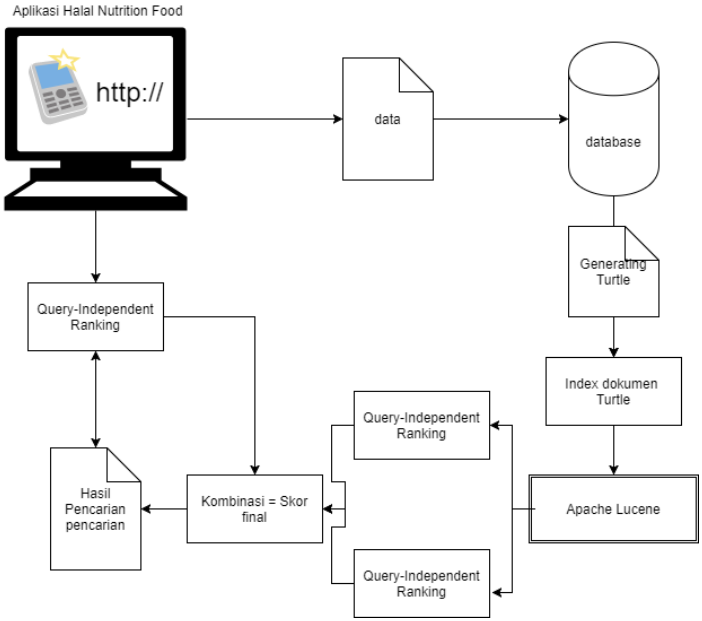
Setelah deployment dilakukan akan dilakukan pembuatan aplikasi android dengan fungsional pencarian seperti halnya pencarian yang terdapat pada aplikasi web Halal Nutrition Food. Aplikasi android melakukan permintaan request pada *url* tertentu yang berfungsi untuk mendapatkan informasi hasil pencarian berdasarkan kueri tertentu melalui data dengan format JSON. Selanjutnya dari data yang diperoleh ini dilakukan parsing data untuk ditampilkan melalui aplikasi android Halal Nutrition Food. Aplikasi Halal Nutrition Food yang berbasis android dapat menghemat bandwith dan mempermudah masyarakat dalam mendapatkan informasi produk halal.

3.1.10 Pembuatan buku Tugas Akhir

Pembuatan buku tugas akhir akan dilakukan secara bersamaan dengan pembuatan sistem. Buku ini berisi tentang semua aktivitas yang ada dan proses yang berhubungan dalam pengerjaan tugas akhir.

3.2 Arsitektur Sistem

Arsitektur sistem dari aplikasi Halal Nutrition Food dibuat melalui 2 layer server yang saling terkoneksi. Penjelasannya seperti Gambar 3.2



Gambar 3.2: Arsitektur Sistem

Penjelasan arsitektur sistem:

1. Server, di dalamnya terdapat dua sistem yang berjalan yakni basis data dan aplikasi web Halal Nutrition Food.
    - Sistem basis data yang digunakan adalah MySQL, skema basis datanya akan dijelaskan pada bagian desain basis data. Sistem kerjanya adalah ketika pengguna memasukkan data produk halal maka aplikasi web akan menyimpan data-data produk tersebut ke dalam basis data MySQL.
    - Aplikasi web Halal Nutrition Food yang berjalan pada server ini dibangun menggunakan framework PHP Laravel yang di dalamnya terdapat fungsi untuk menghubungkan website dengan server Apache Solr, proses penghubungan keduanya digunakan ketika pengguna mengirimkan kueri pencarian produk melalui halaman aplikasi web kemudian aplikasi web akan mengirimkan kueri tersebut untuk diproses menggunakan metode kombinasi *query-independent* dan *query-dependent ranking* di dalam server Apache Solr, selanjutnya server Apache Solr akan mengembalikan daftar hasil pencarian ke aplikasi web.

2. Apache Lucene, digunakan untuk memproses pencarian yang dilakukan pengguna dan menyimpan indeks dari dokumen. Dalam Apache Lucene akan dilakukan dua proses:
    - Proses *indexing* field-field dari dokumen turtle agar pengguna bisa melakukan pencarain berdasarkan data dalam field tersebut.
    - Proses *scoring* menggunakan metode kombinasi *query-independent* dan *query-dependent ranking* sehingga hasil pencarian akan diurutkan berdasarkan bobot penialian metode tersebut.

3.3 Penjelasan Sistem

Sistem pencarian yang dibuat dalam aplikasi Halal Nutrition Food ini adalah sistem pencarian entitas yang tersimpan dalam dokumen turtle. Apache Lucene akan melakukan indexing terhadap dokumen turtle. Kueri yang dimasukkan oleh pengguna selanjutnya akan dihitung kemunculan term pada koleksi dokumen oleh Apache Lucene dan library Halal Ranker akan menghitungnya menggunakan kombinasi *query-independent* dan *query-dependent ranking* dan menampilkannya dalam bentuk data yang akan diolah oleh aplikasi android berbentuk JSON. Metode ranking ini sendiri dipilih sebagai pencarian dalam aplikasi karena meotde ini dapat melakukan ranking skor yang lebih baik melalui final skor seperti yang dijelaskan pada bagian 3.1.6.

Sesuai dengan rumusan masalah yang pertama, bagaimana pengguna mendapatkan informasi komposisi gizi serta karakteristik zat pada produk halal secara relevan dapat terjawab melalui sistem pencarian menggunakan meotde kombinasi *query-independent* dan *query-dependent ranking* yang dapat melakukan raking skor yang lebih baik sehingga pengguna mendapatkan informasi yang relevan sesuai dengan kueri yang dilakukan pengguna. Sistem pencarian ini menampilkan informasi berdasarkan entitas yang terdapat pada koleksi dokumen, berupa produk, bahan produk, dan informasi entitas lain yang terdapat pada dataset.

Rumusan masalah yang kedua, bagaimana menerapkan kombinasi metode *query-independent* dan *query-dependent ranking* untuk memudahkan pencarian produk halal dapat terjawab melalui perhitungan yang dilakukan melalui Apache Lucene dan libray Halal Ranker. Berdasarkan perhitungan yang telah dijelaskan pada bagian 3.1.6, ranking skor pada setiap entitas dilakukan untuk melakukan ranking terbaik sesuai dengan kueri yang dilakukan oleh

pengguna.

Rumusan masalah yang ketiga, bagaimana menampilkan hasil pencarian melalui aplikasi android Halal Nutrition Food dapat terjawab melalui penjelasan yang terdapat bada bagian 3.1.9.

Halaman ini sengaja dikosongkan

BAB 4

PERANCANGAN

Pada bab ini membahas terkait alur perancangan terkait beberapa hal yang diperlukan dalam proses pembuatan aplikasi sesuai dengan alur yang dijelaskan pada bab 3.

4.1 Pengumpulan Data

Pada tahap ini dilakukan proses pengumpulan dan memasukkan data produk halal yang telah tersertifikasi oleh MUI kedalam database aplikasi Halal Nutrition Food. Pengumpulan data diambil dari produk yang beredar di pasar, dari website produk makanan dan minuman. Produk tersebut dimasukkan melalui fitur input produk oleh pengguna di website Halal Nutrition Food.

4.2 Generating Turtle

Pada tahap ini dilakukan proses membuat file turtle (.ttl) produk dan komposisinya yang diambil dari database. Selanjutnya digunakan untuk proses indexing Hasil dari file turtle terdiri dari informasi mengenai produk, informasi komposisi produk berbentuk RDF seperti kode 4.2 berikut ini.

```
halalf:Happy_Tos_Rasa_Jagung_Bakar a halalv:
    FoodProduct;
        halalv:foodCode "8993027163754";
        rdfs:label "Happy Tos Rasa Jagung Bakar
            ";
```

```
halalv:manufacture "PT. Sinar Kencana
    Agung";
halalv:netWeight 55;
halalv:calories 280;
halalv:fat 14;
halalv:saturatedFat 6;
halalv:sodium 99.9;
halalv:fiber 4;
halalv:sugar 1;
halalv:protein 4;
halalv:vitaminA 0;
halalv:vitaminC 0;
halalv:calcium 0;
halalv:iron 0;
halalv:foodproductId 2.
halalv:manufacture halalm:2.
halalv:certificate halalc:2.
halalv:containsIngredient halali:
    Whole_Corn, halali:Palm_Oil, halali:
    Monosodium_glutamate, halali:
    Flavour_Enhacer, halali:
    Sunset_Yellow_FCF.
```

4.3 Desain Index Lucene

Data produk dan komposisi produk yang sudah tersimpan dalam bentuk file *turtle* akan diindex untuk mengetahui masing-masing bobot term yang didapat dari *term-frequency* dan *document-count* pada masing-masing file turtle.

Isi data dari file turtle akan dibaca dan diindex. Atribut file turtle yang akan digunakan dalam pembobotan TF-IEF untuk setiap term yang terdapat pada produk adalah :

1. label
2. containsIngredient

Hal ini didasarkan pada hasil survei pada penelitian sebelumnya[1], yang menunjukkan atribut dengan bobot tertinggi dalam pencarian produk.

4.3.1 Pre-processing Data

Proses *indexing* file turtle dilakukan menggunakan library Apache Jena yang berfungsi untuk membaca nilai subjek, predikat, objek pada dokumen turtle.

Berikut adalah contoh hasil pembacaan pada file turtle menggunakan library Apache Jena :

```
predikat : http://halal.addi.is.its.ac.id/halalv.ttl#containsIngredient
objek    : http://halal.addi.is.its.ac.id/resources/ingredients/
           Monosodium_glutamate

predikat : http://www.w3.org/2000/01/rdf-schema#label
objek    : Happy Tos Rasa Jagung Bakar

predikat : http://halal.addi.is.its.ac.id/halalv.ttl#netWeight
objek    : 55(ttp://www.w3.org/2001/XMLSchema#integer
```

Hasil pembacaan library Apache Jena pada predikat dan objek dari file turtle masih berupa URI. Selanjutnya dilakukan pemotongan karakter untuk mendapatkan nilai yang sebenarnya dari setiap predikat dan objek yang terdapat pada file turtle. Berikut adalah hasil pembacaan setelah pembacaan predikat dan objek diambil dari potongan URI :

```
predikat : containsIngredient
objek    : Monosodium glutamate
```

```
predikat : label
objek    : Happy Tos Rasa Jagung Bakar

predikat : netWeight
objek    : 55
```

Nilai predikat dan objek yang telah dibersihkan kemudian ditokenisasi dan disimpan ke dalam index. Berikut adalah skema index Lucene yang siap digunakan untuk mencari produk terkait seperti pada gambar 4.1

Doc : Happy Tos Rasa Jagung Bakar

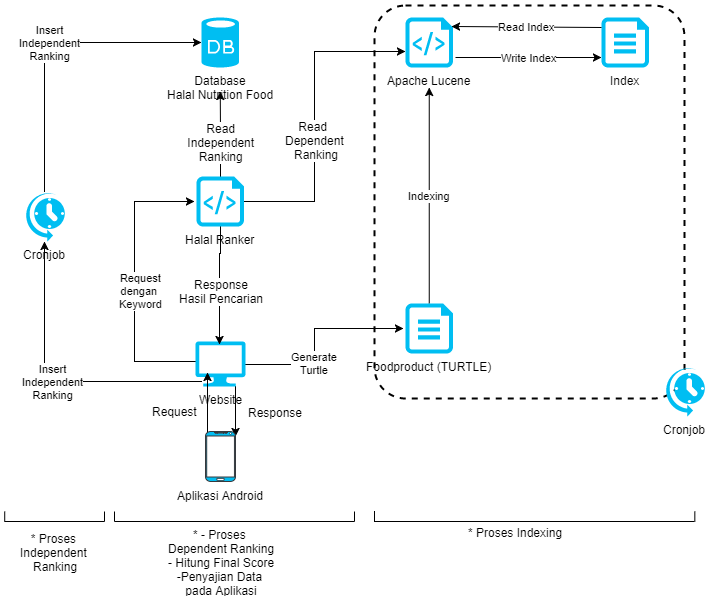
label
bakar;happy;jagung;rasa;tos;

containsIngredient
corn;enhacer;fcf;flavour;glutamate;monosodium;oil;
palm;sunset;whole;yellow;

Gambar 4.1: Desain index Lucene

4.4 Desain Sistem

Sistem pencarian produk terkait pada aplikasi Halal Nutrition Food ini terbagi ke dalam 5 proses, yaitu proses indexing dan proses query produk terkait seperti pada gambar 4.2



Gambar 4.2: Desain sistem dalam aplikasi

Penjelasan desain sistem:

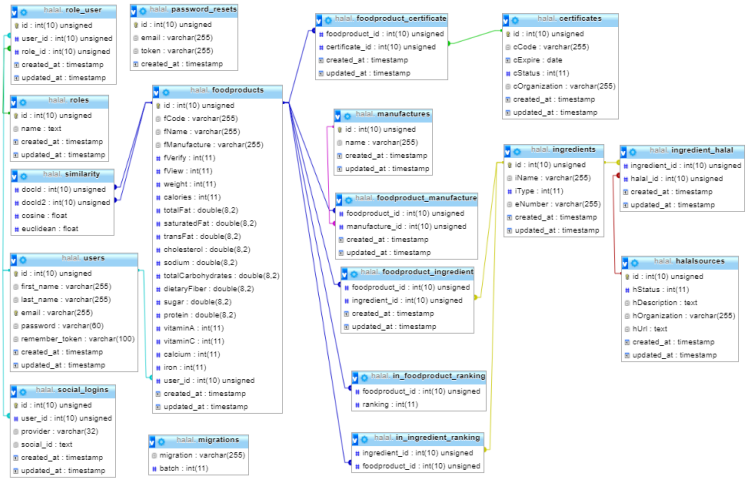
1. Proses *Indexing*, proses ini dilakukan oleh Apache Jena dan Lucene. Apache Jena berfungsi untuk membaca dan mengambil predikat yang akan dijadikan sebagai *field* dan objek pada proses Indexing Apache Lucene. Selanjutnya setiap term yang terdapat pada objek dihitung *term-frequency*-nya. Nilai masing-masing *term-frequency* pada setiap term beserta *field*-nya disimpan ke dalam file index. File index kemudian dibaca kembali oleh Apache Lucene untuk mendapatkan nilai *TF-IEF* pada masing-masing term. Selanjutnya nilai ini akan dikalikan dengan normalisasi *spread* yang akan menghasilkan *query-score*. Proses indexing dilakukan dengan bantuan cronjob yang akan dieksekusi 1 bulan sekali untuk menda-

patkan nilai yang mutakhir dari setiap dokumen.

2. Proses *Dependent Ranking*, proses ini dilakukan untuk mendapatkan nilai *query-score* yang dilakukan oleh library Halal Ranker dibuat pada tugas akhir ini. Proses penghitungan ini dilakukan dengan mengalikan nilai index *TF-IEF* yang didapatkan dari proses *indexing* dengan nilai normalisasi *spread* yang didapatkan dari persebaran *keyword* yang dimasukkan oleh pengguna pada aplikasi Halal Nutrition Food pada objek sebuah dokumen.
3. Proses *Independent Ranking*, proses ini dilakukan untuk mendapatkan nilai *static-score* yang dilakukan dengan cara mendapatkan *link-count* yang terdiri dari link masuk dan link keluar dari setiap entitas. Banyaknya linkcount akan menentukan popularitas sebuah entitas pada sebuah dataset. Sehingga, besar kemungkinan entitas dengan *static-score* yang tinggi akan menempati urutan teratas pada sebuah hasil pencarian. Proses ini dilakukan dengan bantuan cronjob yang akan dieksekusi 1 bulan sekali untuk mendapatkan nilai *linkcount* yang mutakhir dari setiap entitas.
4. Proses Penghitungan *Final Score*, pada proses penghitungan nilai *final-score* didapatkan melalui penghitungan kombinasi *static-score* yang berasal dari proses *independent-ranking* dan *query-score* seperti rumus 3.1. Proses ini akan menghasilkan niali ranking dokumen yang menentukan urutan relevansi pada hasil pencarian.
5. Penyajian Data pada Aplikasi, data luaran hasil pencarian yang berhasil diurutkan selanjutnya akan ditampilkan oleh sistem melalui aplikasi website atau android yang nilai dokumennya diurutkan berdasarkan *final-score* yang telah dihitung sebelumnya. Semua proses diatas diawali dengan permintaan yang dilakukan oleh pengguna dengan melakukan pencarian pada aplikasi Halal Nutrition Food baik melalui website maupun android.

4.5 Desain Basis Data

Skema basis data meneruskan dari skema basis data aplikasi Halal Nutrition Food sebelumnya [1], dengan penambahan tabel in_foodproduct_ranking dan in_ingredient_ranking. Kedua tabel ini berfungsi untuk menyimpan ranking dari setiap entitas atau dokumen yang terdapat pada dataset. Skema basis data yang digunakan dalam aplikasi Halal Nutrition Food ini digambarkan pada Gambar 4.3



Gambar 4.3: Skema basis data yang digunakan dalam aplikasi

Penjelasan tabel dalam skema basis data:

- foodproducts berisi data-data utama dari produk makanan dan minuman seperti kode produk, nama produk, produsen, kandungan gizi, dan nama pengguna yang memasukkan produk

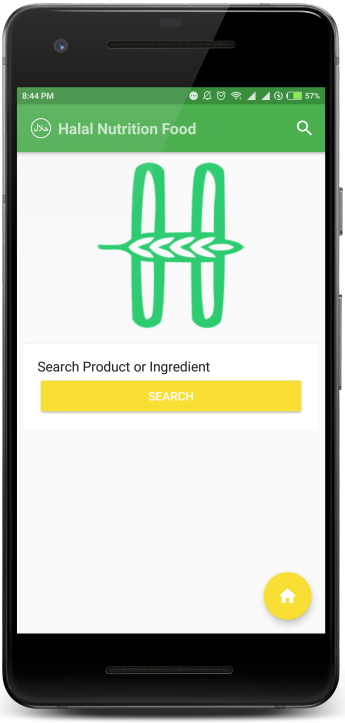
tersebut. Tabel ini menjadi referensi untuk beberapa tabel berikutnya.

- foodproducts_ingredient adalah tabel relasi antara tabel foodproducts dan tabel ingredient. Di dalamnya berisi kolom id produk dan id ingredient yang ada di dalam setiap produk.
- foodproducts_manufacture adalah tabel relasi antara foodproducts dan tabel manufacture. Di dalamnya berisi kolom id produk dan id manufacture.
- foodproducts_certificate adalah tabel relasi antara tabel foodproducts dan tabel certificate. Di dalamnya berisi kolom id produk dan id certificate yang dimiliki oleh setiap produk.
- in_foodproduct_ranking adalah tabel yang menyimpan nilai ranking *link-count* dari entitas produk. Jumlah *linkcount* didapatkan dari banyaknya komposisi sebuah produk yang memiliki kesamaan dengan resource DBpedia. Semakin banyak komposisi yang dimiliki oleh sebuah produk, maka semakin besar nilai *linkcount*-nya.
- in_ingredient_ranking adalah tabel yang menyimpan nilai ranking *link-count* dari entitas sebuah komposisi produk. Jumlah *linkcount* didapatkan dari banyaknya komposisi produk terdapat pada produk yang terdapat pada dataset. Semakin banyak komposisi yang dimiliki pada produk-produk yang terdapat pada dataset, maka semakin besar peluang nilai *linkcount*-nya.
- ingredients berisi data mengenai bahan-bahan atau zat aditif yang terkandung dalam setiap produk. Di dalamnya berisi kolom nama zat, tipe zat, eNumber zat, dsb.
- ingredients_halal adalah tabel relasi antara tabel ingredients dengan tabel halalsources. Di dalamnya berisi kolom id ingredients dan id dari sumber halal. Fungsinya untuk melihat apakah bahan tersebut termasuk bahan yang halal atau tidak.
- similarity adalah tabel yang menyimpan nilai kemiripan dua

produk. Di dalamnya berisi kolom docId dan docId2 yang berasal dari id produk, nilai *cosine* dan *euclidean similarity* antara dua produk.
- halalsources berisi data mengenai penjelasan sumber halal.
- manufactures berisi data mengenai produsen makanan dan minuman tersebut.
- certificates berisi data mengenai sertifikat halal yang dimiliki oleh sebuah produk dari berbagai sumber.
- users berisi data pengguna yang telah terdaftar dalam aplikasi yang merupakan tabel referensi untuk menentukan role dari setiap pengguna.
- roles berisi data mengenai role pengguna yang ada di dalam aplikasi.
- role_user adalah tabel relasi antara tabel users dan tabel roles. Di dalamnya berisi kolom id user dan id role. Fungsinya untuk menjelaskan seorang pengguna mempunyai role sebagai apa.
- social_login adalah tabel yang berisi data mengenai pengguna yang login ke aplikasi menggunakan sosial media yang dimilikinya.
- migrations adalah tabel bawaan dari Laravel yang berfungsi untuk menyimpan *migration* yang dimiliki oleh aplikasi.

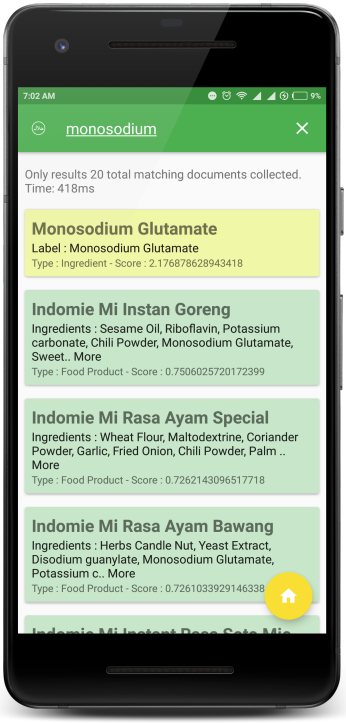
4.6 Desain Antarmuka Pengguna

Tugas akhir ini berfokus untuk menampilkan hasil pencarian entias yang paling relevan dengan keyword yang dimasukkan oleh pengguna. Entitas dapat berupa produk atau sebuah komposisi produk.



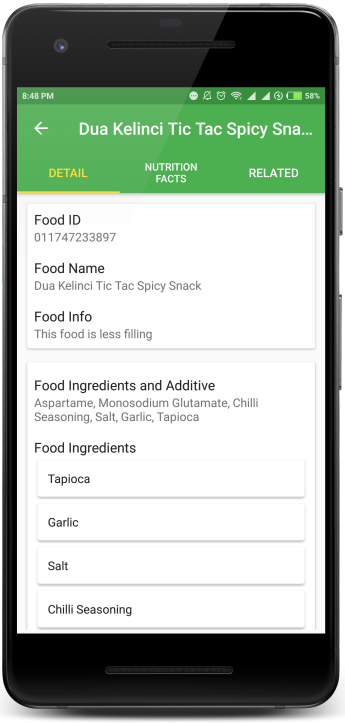
Gambar 4.4: Halaman depan aplikasi

Halaman depan aplikasi 4.4 menampilkan tombol yang berfungsi untuk melakukan pencarian entitas.



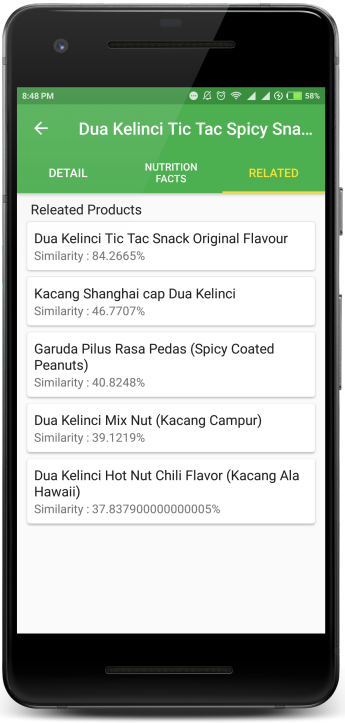
Gambar 4.5: Hasil pencarian dengan keyword ”monosodium”

Hasil pencarian yang digambarkan pada Gambar 4.5 ini menampilkan hasil pencarian berupa nama entitas dengan masing-masing nama entitas maupun komposisi produk jika entitas merupakan sebuah produk. Hasil pencarian diurutkan sesuai dengan relevansi yang dihitung berdasarkan skor pada masing-masing entitas dengan keyword tertentu.



Gambar 4.6: Detail Informasi Produk

Gambar 4.7: Detail Nutrition Facts



Gambar 4.8: Detail Related Product

Setelah hasil pencarian diklik, maka akan ditampilkan *activity* detail informasi dari entitas produk. Informasi detail produk dan komposisi produk ditampilkan seperti gambar 4.6. Informasi *nutrition facts* ditampilkan seperti gambar 4.7. Informasi produk terkait ditampilkan seperti gambar 4.8.

BAB 5

IMPLEMENTASI

Pada bab ini akan dijelaskan terkait proses implementasi pada perangkat lunak yang dirancang.

5.1 Lingkungan Implementasi

Pada bagian ini dibahas terkait lingkungan pengujian yang digunakan dalam implemetasi tugas akhir terkait perangkat yang digunakan baik perangkat keras maupun perangkat lunak. Tabel 5.1 yang berisikan spesifikasi perangkat keras dan perangkat lunak untuk implementasi pada tugas akhir ini.

Tabel 5.1: Spesifikasi Perangkat Keras

Perangkat	Spesifikasi
Jenis	ASUS ROG GL553VD
Processor	Intel Core i7
RAM	16GB
Hard Disk Drive	1125GB

Kemudian untuk perangkat lunak yang digunakan dalam implementasi model ditunjukkan dalam tabel 5.2.

5.2 Implementasi

Pada proses implementasi akan dilakukan realisasi dari perancangan yang sudah dibuat sebelumnya. Berikut adalah proses imple-

Tabel 5.2: Spesifikasi Perangkat Lunak

Nama Perangkat Lunak	Kegunaan dalam Implementasi
Xampp 7.0.15 dengan PHP 5.6.14	Webserver
Apache Lucene	Search Engine
IntelliJ IDEA 2017.2	Java IDE
Visual Code Studio	Text Editor
Google Chrome 62	Web Browser

mentasi yang dilakukan:

1. Konfigurasi Server
2. Pembacaan File Turtle
3. Proses Indexing
4. Proses *Query Independent*
5. Proses *Independent Ranking*
6. Proses Penghitungan *Final Score*
7. Penyajian Data pada Aplikasi

Pada proses pembacaan file turtle hingga proses penghitungan *final-score* akan dilakukan menggunakan library *Halal Ranker* berbentuk file dengan ekstensi *.jar yang telah dibuat pada tugas akhir ini.

5.2.1 Konfigurasi Server

Konfigurasi server yang dimaksud meliputi konfigurasi untuk Apache Web Server dan MySQL.

- Apache Web Server berjalan pada port 80 sebagai server dari aplikasi Halal Nutrition Food.
- MySQL berjalan pada port 3306 sebagai database yang me-

nampung data produk makanan, komposisi, produk terkait, dan konfigurasi aplikasi Halal Nutrition Food.

- Java Runtime Environment (JRE) yang terinstal adalah versi 1.8.0_151.

5.2.2 Pembacaan File Turtle

File turtle yang telah dihasilkan dibaca menggunakan library dari Apache Jena. Konfigurasi untuk pembacaan file turtle yang digunakan seperti kode 5.1 sebagai berikut :

Kode 5.1: Kode untuk membaca dokumen turtle

```
//membuat object list untuk menyimpan data
    TurtleModel kedalam list sementara
private List<TurtleModel> spo;
//membuat method dengan kembalian list
    TurtleModel
public List<TurtleModel> readTurtle(String path)
    throws IOException{
        this.spo = new ArrayList<>();

        //membuat model untuk membaca turtle
        Model model = ModelFactory.
            createDefaultModel();

        Path sfilePath = Paths.get(path);

        try(InputStream in = Files.
            newInputStream(sfilePath)) {
            if (in == null) {
                throw new
                    IllegalArgumentException("
                    File: " + inputFileName + "
                    not found");
            }
```

```
model.read(in, null, "TURTLE");

//mendefinisikan query untuk membaca
    dokumen turtle ?s ?p ?o

final QueryExecution exec =
    QueryExecutionFactory.create(
    query_id, model);
final ResultSet rs = exec.execSelect
    ();
while (rs.hasNext()) {

    final QuerySolution qs = rs.next
        ();

    //mendapatkan nilai subjek
    String s = qs.get("s").toString
        ();
    String sub[] = s.split("/");
    String subjek = sub[sub.length -
        1];

    //mendapatkan nilai predikat
    String p = qs.get("p").toString
        ();
    String pred[] = p.split("#");
    String predikat = pred[pred.
        length - 1];

    //mendapatkan nilai objek
    String o = qs.get("o").toString
        ();
    String objek = o;


    if (o.contains("^^")) {
        String obj[] = o.split("
```

```
                \\^\\^");
            objek = obj[0];;
        } else if (o.contains("/")) {
            String obj[] = o.split("/");
            //objek = obj[obj.length -
                2] + " " + obj[obj.length
                 - 1];
            objek = obj[obj.length - 1];
            objek = objek.replace("
                halalv.ttl#", "");
        }
        else {
            objek = o;
        }
        //menambahkan objek turtle model
             ke list
        spo.add(new TurtleModel(subjek,
            predikat, objek));

    }
}

//mengembalikan nilai List TurtleModel
    yang telah berhasil terbaca
return spo;

}
```

Method diatas tersebut akan membaca subjek, predikat, objek file turtle dari inputStream yang berupa path yang merujuk pada file turtle yang telah diatur pada parameter method readTurtle.

Pada method diatas juga dilakukan pembersihan simbol yang terdapat pada URI seperti *underscore*, *slash* dan simbol-simbil lain. Method diatas juga akan menghilangkan bagian URI yang tidak dibutuhkan dalam proses indexing dan hanya nilai dari masing-masing

subjek, predikat, dan objek.

Nilai subjek, predikat, objek yang sudah bersih kemudian ditambahkan ke dalam List<TurtleModel>untuk disimpan sementara untuk kemudian dikembalikan dalam bentuk List TurtleModel yang akan diindex.

5.2.3 Proses *Indexing*

Data produk diindex menggunaan Apache Lucene untuk menghitung nilai *TF-IEF* pada setiap terms yang terdapat pada sebuah produk atau komposisi produk.

Berikut adalah konfigurasi yang digunakan untuk melakukan indexing terhadap field dari file turtle dengan bantuan Apache Lucene seperti kode 5.2 :

Kode 5.2: Kode untuk mendefinisikan jenis Field

```
FieldType TYPE_STORED = new FieldType();

TYPE_STORED.setTokenized(true);
TYPE_STORED.setStored(true);
TYPE_STORED.setStoreTermVectors(true);
TYPE_STORED.setIndexOptions(IndexOptions.
    DOCS_AND_FREQS);
TYPE_STORED.freeze();
```

Konfigurasi diatas merupakan konfigurasi untuk menentukan jenis dari *field* dari tiap dokumen, dalam hal ini *field* diwakili oleh predikat yang terdapat pada dokumen turtle. Konfigurasi tersebut akan meminta agar proses indexing melakukan tokenisasi pada field, menyimpan nilai dari field, dan menyimpan *term vector* dari field. Term Vector akan digunakan untuk menghitung kemiripan antar

entitas satu dengan entitas lainnya.

Dalam membuat index, diperlukan direktori untuk menyimpan hasil index dan objek IndexWriter untuk melakukan proses index. IndexWriter berfungsi untuk menuliskan index ke dalam direktori yang telah diatur sebelumnya, konfigurasi ini diatur pada IndexWriterConfig. Selain mengatur tentang lokasi index, IndexWriterConfig juga akan menentukan *analyzer* dan pembobotan masing-masing term pada sebuah dokumen. Bobot ini nantinya akan digunakan dalam penentuan *query-score* yang akan dijelaskan pada bab berikutnya. Berikut merupakan kode 5.3 dari kelas IndexWriter dan IndexWriterConfig :

Kode 5.3: Kode untukmelakukan indexing pada dokumen

```
/*menentukan direktori index*/
Directory directory = FSDirectory.open(Paths.get
    (INDEX_PATH));
Analyzer analyzer = new StandardAnalyzer();

/*membuat instance IndexWriterConfig*/
IndexWriterConfig iwc = new IndexWriterConfig(
    analyzer);
iwc.setSimilarity(new TFIEFSimilarity());

/*membuat instance IndexWriter*/
IndexWriter writer = new IndexWriter(directory,
    iwc);
```

Pada konfigurasi IndexWriterConfig, digunakan StandardAnalyzer untuk proses analisa dan tokenisasi teks sesuai dengan Work Break Rules (http://unicode.org/reports/tr29/) yang berfungsi untuk melakukan tokenisasi berdasarkan spasi dan tanda baca.

Pada IndexWriterConfig diatur bobot *TF-IEF* untuk menentukan bobot dari masing-masing term menggunakan kelas TFIEFSimila-

rity. Implementasi TFIEFSimilarity dilakukan dengan melakukan *extend* kelas Similarity dari library Apache Lucene. Konfigurasi indexing *TF-IEF* ini digunakan ketika proses penambahan field dan kontennya ke dalam dokumen index. Untuk menambahkan field dan kontennya pada dokumen index, dilakukan menggunakan method indexDoc. Method indexDoc akan memanggil method readTurtle yang telah dibuat pada bagian sebelumnya untuk menambahkan filed berserta kontennya. Method indexDoc akan melakukan iterasi terhadap *field-field* yang terdapat pada dokumen dan menambahkan kontennya kedalam dokumen index. Berikut adalah detail dari untuk memasukkan field dan konten ke dalam dokumen index seperti kode 5.4 berikut ini :

Kode 5.4: Kode untuk menambahkan dokumen kedalam field tertentu dengan value tertentu

```
//membuat object list TurtleModel untuk
    menyimpan list objek sementara
private static List<TurtleModel> turtleModels;
//method untuk mengindex sebuah dokumen
static void indexDoc(IndexWriter writer, Path
    file, long lastModified) throws IOException {
try (InputStream stream = Files.newInputStream(
    file)) {

    Document doc = new Document();

    ReadTurtle readTurtle = new ReadTurtle();

    turtleModels = readTurtle.readTurtle(file.
        toAbsolutePath().toString());

    Field pathField = new StringField("path",
        file.toString(), Field.Store.YES);
    doc.add(pathField);

    for (int i = 0; i < turtleModels.size(); i
```

```
        ++) {
            //menambahkan field beserta kontennya
                untuk diindex
            Field predicate = new Field(turtleModels
                .get(i).getPredicate(), turtleModels.
                get(i).getObject().replace("_", "␣"),
                 TYPE_STORED);
            doc.add(predicate);
        }

        //memberikan tanda bahwa dokumen berhasil di
            tambahkan pada index
        if (writer.getConfig().getOpenMode() ==
            IndexWriterConfig.OpenMode.CREATE) {
            // New index, so we just add the
                document (no old document can be
                there):
            System.out.println("adding␣" + file);
            writer.addDocument(doc);
        } else {
            // Existing index (an old copy of this
                document may have been indexed) so
            // we use updateDocument instead to
                replace the old one matching the
                exact
                    // path, if present:
            System.out.println("updating␣" + file);
            writer.updateDocument(new Term("path",
                file.toString()), doc);
        }
}}
```

5.2.4 Proses Dependent Ranking

Pada proses ini dilakukan penghitungan *query-score* yaitu dengan mengalikan skor *TF-IEF* dari term kueri *keyword* yang dimasukkan oleh pengguna dengan normalisasi spread. Perhitungan skor *TF-IEF* dilakukan melalui kelas TFIEFSimilarity pada method score seperti berikut ini :

Penghitungan *query-score* dilakukan pada kelas DependentRanking dengan method getSpreadValue yang mengembalikan nilai normalisasi spread dengan type double yang didapatkan dari perhitungan term dan field yang terdapat pada dokumen. Berikut ini kode 5.5 untuk menghitung skor dari tiap dokumen.

Kode 5.5: Kode untuk melakukan penilaian skor dokumen

```
 @Override
public float score(int i, float f) throws
    IOException {
     //menghitung skor TF–IEF
     return (float) ((TFIEFSimilarity.this.tf(f))
          * Math.log10(weight.docCount / (1 +
         TFIEFSimilarity.this.tf(f))));
}
```

Berikut ini adalah kode 5.6 detail dari method getSpreadValue.

Kode 5.6: Kode untuk menghitung nilai spread dari dokumen

```
//method dengan parameter array term dan list
    dari fields untuk menghitung nilai
    normalisasi spread
public Double getSpreadValue(String[]
    uniqueTerms, List<IndexableField> fields) {
     //jumlah dari term yang terdapat pada kueri
     double q = uniqueTerms.length;
```

```
//nilai default term yang pasti terdapat
    pada satu objek pada dokumen minimal
    berjumlah 1
double v = 1;
//variable untuk mengecek apakah field sama
    dengan field perulangan berikutnya
String atributeTemp = "";
//perulangan diulang sebanyak jumlah dari
    term yang terdapat pada kueri
for (int j = 0; j < uniqueTerms.length; j++)
    {
    //perulangan diulang sebanyak jumlah
        field
    for (int k = 0; k < fields.size(); k++)
        {
        //jika nama field merupakan sama
            dengan subject atau
            containsIngredient atau path maka
             akan dihiraukan, karena sudah
            temasuk nilai default term yang
            pasti terdapat pada dokumen
        if ((fields.get(k).name().
            equalsIgnoreCase(Settings.
            FIELD1_LABEL)
                || fields.get(k).name().
                    equalsIgnoreCase(Settings
                    .
                    FIELD2_CONTAINSINGREDIENT
                    )
                || fields.get(k).name().
                    equalsIgnoreCase(Settings
                    .FIELD_EX_SPREAD0_SUBJECT
                    )
                || fields.get(k).name().
                    equalsIgnoreCase(Settings
                    .FIELD_EX_SPREAD1_PATH)))
                     {
```

```
            } else {
                //jika nilai dari field
                    mengandung term maka akan
                    menambah nilai dari v
                if (fields.get(k).stringValue().
                    toLowerCase().contains(
                    uniqueTerms[j])) {
                     if (!atributeTemp.
                         equalsIgnoreCase(fields.
                         get(k).name())) {
                                 v++;
                          atributeTemp = fields.
                              get(k).name();
                     } else {

                     }
                          } else {

                }
                     }
        }

    }
    //menghitung sesuai dengan rumus yang telah
        dijelaskan pada bab sebelumnya
    Double spread = (Math.abs(q)-Math.abs(v/
        fields.size())+1)/Math.abs(q);

    return spread;

}
```

5.2.5 Proses Independent Ranking

Pada proses ini dilakukan penghitungan *linkcount* dari masing-masing entitas (produk dan komposisi produk) yang terdapat pada dataset. *Linkcount* akan menentukan *static-score* pada sebuah entitas. Semakin banyak *linkcount* yang dimiliki sebuah entitas, maka akan berpeluang untuk muncul pada hasil pencarian. Untuk menghitung *linkcount* produk dilakukan dengan menghitung banyaknya komposisi produk yang terkadung didalamnya. Sedangkan untuk *linkcount* komposisi produk dihitung berdasarkan banyaknya komposisi tersebut terkandung pada produk yang terdapat pada dataset. Berikut ini adalah kdoe 5.7 detail kode untuk melakukan penghitungan *linkcount* pada produk maupun komposisi produk :

Kode 5.7: Kode untuk membobotkan skor linkcount pada entitas produk makanan

```
public function makeIndependentRanking() {
    //menghapus isi dari database untuk
        memastikan tidak terjadi duplikasi dan
        perulangan ranking
        DB::table('in_ingredient_ranking')->
            truncate();
        DB::table('in_foodproduct_ranking')->
            truncate();
         //memilih data produk yang sudah
            terverifikasi
    $foodProducts = FoodProduct::where('fVerify'
        ,1)->get()->toArray();
    foreach ($foodProducts as $fp => $val) {
      $getIngFK = DB::select('select * from
          foodproduct_ingredient where
          foodproduct_id = ?', [$foodProducts[$fp
          ]['id']]);
      $foodproduct_id = $foodProducts[$fp]['id'
          ];
```

```
        $ranking = 0;
          foreach ($getIngFK as $id => $val) {

                    $ingredient[$id] = Ingredient::
                        findOrFail($getIngFK[$id]->
                        ingredient_id);
                    $ing_id = $getIngFK[$id]->
                        ingredient_id;

                    $DBpedia = @file_get_contents("
                       URL_QUERY_RESOURCE_DBPEDIA");
                if($DBpedia == "true") {
                        $ranking = $ranking+1;
                        DB::table('
                            in_ingredient_ranking')->
                            insert(
                             ['ingredient_id' =>
                                 $ing_id, '
                                 foodproduct_id' =>
                                 $foodproduct_id]
                    );
                } else{ }

            //menambahkan nilai ranking produk ke
                database
            DB::table('in_foodproduct_ranking')->
                insert(
                ['foodproduct_id' => $foodproduct_id
                    , 'ranking' => $ranking]
                    );
    }
}
```

Setelah proses penghitungan *linkcount* produk dan komposisi produk selesai, maka dilakukan pembacaan terhadap *linkcount* pada masing-masing produk atau komposisi produk yang sesuai dengan

keyword pengguna. Pembacaan *linkcount* dilakukan pada kelas IndependentRanking dengan method getIndependentRanking dengan parameter typeEntity dan id dari entitas. Berikut ini kode 5.8 detail dari method getIndependentRanking :

Kode 5.8: Kode untuk mengambil skor linkcount pada entitas produk makanan

```
//membuat method untuk menghitung independent
    ranking dari sebuah entitas
public static Double getIndependentRanking(
    String typeEntityParams, int idParams) {
//membuat koneksi ke mysql dengan jdbc

        //jika entitas merupakan produk maka
            kueri dilakukan pada tabel
            in_foodproduct_ranking
        if (typeEntityParams.equalsIgnoreCase("
            FoodProduct")) {
            String query = "SELECT * FROM
                in_foodproduct_ranking where
                foodproduct_id ="+idParams;

            Statement st = conn.createStatement
                ();

            ResultSet rs = st.executeQuery(query
                );

            Double ranking = 0.0;
            while (rs.next())
            {
                //mengambil nilai ranking
                ranking = rs.getDouble("ranking"
                    );
            }
            //menutup statement dan resultset
            st.close();
```

```
            rs.close();
            return ranking;
        //jika entitas merupakan produk maka
            kueri dilakukan pada tabel
            in_ingredient_ranking
        } else if (typeEntityParams.
            equalsIgnoreCase("FoodAdditive") ||
            typeEntityParams.equalsIgnoreCase("
            Ingredient")) {
                String query = "SELECT count(
                    ingredient_id) as
                    ingredient_ranking FROM
                    in_ingredient_ranking where
                    ingredient_id = "+idParams;
                Statement st = conn.createStatement
                    ();
                ResultSet rs = st.executeQuery(query
                    );

                Double ranking = 0.0;
                while (rs.next())
                {
                    //mengambil nilai ranking
                    ranking = rs.getDouble("
                        ingredient_ranking");
                }
                //menutup statement dan resultset
                st.close();
                rs.close();
                return ranking;
        } else {
            return  0.0;
        }

}
```

5.2.6 Proses Penghitungan *Final Score*

Pada proses ini dilakukan perhitungan *final-score* yang didapatkan dari perhitungan antara *query-score* dan *static-score* seperti yang telah dijelaskan pada rumus 3.1. Hasil *final-score* akan menentukan urutan dari sebuah entitas pada hasil pencarian tertentu. Berikut ini kode 5.9 detail dari eksekusi hingga didapatkannya *final-score* :

Kode 5.9: Kode untuk mendapatkan skor final

```
//mendapatkan jumlah panjang dokumen
double docLeng = termLength(totalterms);
//mengambil spread dari term yang terdapat pada
    kueri
Double spread = dependentRanking.getSpreadValue(
    uniqueTerms, fields);
float tfief = (float) (hits[i].score/docLeng);
//menghitung queryscore
Double queryScore = tfief * spread;
//mengambil dan menghitung staticscore dari
    database mysql
Double staticScore = independentRanking.
    getIndependentRanking(entityType,
    entityIdforDb);
//menghitung nilai final-score dari dokumen
Double finalScore = finalScoreRanking.
    getFinalScore(staticScore, queryScore);
```

Setelah diketahui *final-score* dari masing-masing dokumen, selanjutnya ditambahkan masing-masing filed dan konten dari dokumen-dokumen tersebut kedalam sebuah model yang akan disajikan dalam bentuk JSON sehingga dapat digunakan kembali oleh berbagai aplikasi. Sebelum ditampilkan dalam bentuk JSON dibuat *Plain Java Old Object* untuk menyimpan objek yang selanjutnya akan dikonversi menjadi JSON. Kelas Entity yang merupakan model dari entitas akan mewakili data dari setiap entitas yang ada pada hasil

pencarian.

Selanjutnya membuat objek entitas dan memasukkannya kedalam *List Entity* yang akan menyimpan keseluruhan data entitas dari hasil pencarian. Berikut ini adalah detail kode 5.10 yang membuat entitas dan menambahkannya kedalam *List Entity* :

Kode 5.10: Kode untuk mengatur ranking dokumen

```
//membuat objek list entity
List<Entity> entityList = new ArrayList<>();
Entity entity = new Entity();
entity.setLabel(doc.get("label"));
//mengatur skor untuk setiap entitas
entity.setScore(finalScore);
//mengatur atribute untuk masing-masing entitas
entity.setAtribute(atribute);
//menambahkan entitas ke dalam list untuk
    diurutkan
entityList.add(entity);
//mengurutkan entitas dari skor tertinggi ke
    rendah
entityList.sort(Comparator.comparingDouble(
    Entity::getScore).reversed());
```

Setelah dokumen diurutkan berdasarkan *final-score* dalam bentuk list objek, data akan ditampilkan dalam bentuk JSON menggunakan library GSON. GSON akan mengubah list objek yang masih berbentuk objek Java menjadi JSON. List objek java ditambahkan terlebih dahulu ke objek kelas ResultEntity yang berfungsi untuk dikonversi menjadi JSON. Objek kelas ResultEntity akan dikonversi oleh kelas GsonHelper dengan method convertToJSON yang memiliki parameter kelas ResultEntity. Berikut ini detail dari method convertToJSON. Proses konversi objek ResultEntity menjadi JSON dilakukan melalui eksekusi method convertToJSON pada objek Re-

sultEntity seperti kode 5.11 berikut ini :

Kode 5.11: Kode untuk menampilkan output dari list hasil pencarian

```
//membuat objek ResultEntity
ResultEntity resultEntity = new ResultEntity(
    messageSearch, entityList);
//menampilkan hasil konversi JSON
System.out.println(new GsonHelper().
    convertToJSON(resultEntity));
//keluar dari aplikasi
System.exit(1);
```

Tampilan JSON yang akan ditampilkan seperti kode 5.12 berikut ini :

Kode 5.12: Output hasil pencarian dengan kueri ”monosodium”

```
//hasil pencarian menggunakan keyword "
    monosodium"

{
  "message": "Only␣results␣7␣total␣matching␣
     documents␣collected.",
  "entityData": [
    {
      "score": 1.9522362301613012,
      "label": "Monosodium␣glutamate",
      "atribute": {
        "path": "/var/www/halal/public/resources
            /ingredients/Monosodium_glutamate.ttl
            ",
        "rank": "283",
        "comment": "E621",
        "label": "Monosodium␣glutamate",
        "sameAs": "Monosodium␣glutamate"
      }
    }, ..
```

```
    ]
}
```

Setelah JSON berhasil ditampilkan, selanjutnya hasil luaran ini dapat dimanfaatkan untuk ditampilkan pada aplikasi website maupun android. Proses penyajian data melalui aplikasi website maupun android. Penyajian pada aplikasi android akan dijelaskan pada sub bab selanjutnya.

5.2.7 Penyajian Data pada Aplikasi

Pada proses penyajian data pada aplikasi dibagi menjadi dua tahap yaitu pembuatan *end-point API (Application programming Interface)* dan pembuatan aplikasi android yang menampilkan data luaran dari hasil *API* yang telah dibuat.

Pembuatan *End-point API*

Data JSON yang ditampilkan oleh library *Halal Ranker* selanjutnya ditampilkan melalui aplikasi website Halal Nutrition Food. Hasil luaran tampilan akan ditampilkan melalui URL *end-point API* salah satunya yaitu */search?q=keyword*. *End-point API* ini selanjutnya akan menjadi *end-point* request yang dilakukan melalui aplikasi android. Berikut ini detail kode proses menampilkan hasil luaran library *Halal Ranker* dan Halal Nutrition Food ke dalam *end-point* API seperti kode **??** berikut ini :

Kode 5.13: End-point pada aplikasi

```
//setting file routes.php pada laravel
Route::get('/search','RankingController@index');
```

```
Route::get('/related', '
    RankingController@releated');
Route::get('/inglist', '
    RankingController@inglist');
Route::get('/addlist', '
    RankingController@addlist');

//method index untuk menampilkan data hasil
    pencarian dengan keyword tertentu
public function index() {
    if (Input::has('q')) {
        $q = Input::get('q');
        // file Halal.jar merupakan library \
            textit{Halal Ranker} yang dibuat pada
             tugas akhir ini
        $eksekusi = 'java -jar ./Halal.jar -
            query "'.$q.'"';
        //proses eksekusi Halal.jar melalui php
        $a = shell_exec($eksekusi);
        echo $a;
    }
}

//method releated untuk menampilkan data produk
    terkait
public function releated() {
    if (Input::has('id')) {
        $id = Input::get('id');
        //query untuk similarity
        $similarity = DB::select('SELECT id,
            fName, cosine, euclidean
        FROM foodproducts
        INNER JOIN similarity
        ON foodproducts.id = similarity.docId
        OR foodproducts.id = similarity.docId2
        WHERE (
        similarity.docId = :id1
        OR similarity.docId2 = :id2
```

```
        )
        AND foodproducts.id != :id3
        ORDER BY similarity.cosine DESC
        LIMIT 5', ['id1' => $id,'id2' => $id,'
    id3' => $id]);
    //menampilkan dalam bentuk JSON
        return response()->json($similarity);
    }
}

//method menampilkan daftar komposisi produk
public function inglist() {
    if (Input::has('id')) {
                $id = Input::get('id');
                $foodProduct = FoodProduct::find
                    ($id);
                if (!empty($foodProduct)) {
                        $ingredients =
                            $foodProduct->
                            ingredient->all();
                        $inglist = array();
            foreach ($ingredients as $ing) {
                if($ing->iType==0){
                    $inglist[] = array('name' =>
                        $ing->iName, 'id' =>
                        $ing->id, 'type' => '
                        Ingredient');
                }
            }
        }
    //menampilkan dalam bentuk JSON
    return response()->json($inglist);
}

//method menampilkan daftar zat additif produk
public function addlist() {
    if (Input::has('id')) {
        $id = Input::get('id');
```

```
        //melakukan wuery select dengan id $id
        $foodProduct = FoodProduct::find($id);
        if (!empty($foodProduct)) {
                $ingredients = $foodProduct->
                    ingredient->all();
                $addlist = array();
            foreach ($ingredients as $ing) {
                if($ing->iType!=0) {
                    z$addlist[] = array('name'
                        => $ing->iName, 'id' =>
                        $ing->id, 'type' => '
                        Additive');
                }
            }
        }
    //menampilkan dalam bentuk JSON
        return response()->json($addlist);
    }
}
```

Pembuatan Aplikasi Android

Pembuatan aplikasi android memiliki fungsional utama untuk menampilkan data-data yang telah dibuat pada *API*. Aplikasi android dibangun diatas 6 package utama yaitu activity, adapter, api, fragment, model dan utils. Setiap directory package berisi kelas-kelas yang memiliki fungsi masing-masing. Berikut ini adalah fungsi kelas dari masing-masing pakcage :

1. Activity
   (a) MainActivity
   Pada *activity* ini berfungsi untuk memberikan alat kepada pengguna untuk melakukan pencarian terhadap produk atau komposisi produk. Selanjutnya dari masuk-

an pengguna, *activity* ini berfungsi untuk menampilkan respon dari *end-point API search* yang telah dibuat, akan dijelaskan pada poin berikutnya pada package *Api*. Berikut ini merupakan method 5.14 untuk memanggil *end-point API /search* :

Kode 5.14: Memanggil data dari server dan menampilkannya ke aplikasi android

```
//inisiasi objek
List<EntityDatum> listEntity;
RecyclerView recyclerView;
EntitysAdapter adapter;
ResultEntity resultEntity;
private ProgressDialog dialog;
//method untuk memanggil end-point API
private void getRetrofitSearch(String
    query) {
     dialog.show();
     Retrofit retrofit = new Retrofit.
         Builder()
          .baseUrl(Params.BASE_URL)
          .addConverterFactory(
              GsonConverterFactory.create
              ())
          .build();

          RetrofitHalalAPI service =
              retrofit.create(
              RetrofitHalalAPI.class);

     Call<ResultEntity> call = service.
         getSearchEntity(query);

     call.enqueue(new Callback<
         ResultEntity>() {
          @Override
```

```
        public void onResponse(Call<
            ResultEntity> call, Response
            <ResultEntity> response) {
            resultEntity = new
                ResultEntity();
            resultEntity.setMessage(
                response.body().
                getMessage());
            resultEntity.setEntityData(
                response.body().
                getEntityData());

            listEntity = response.body()
                .getEntityData();
            if (response.body().
                getEntityData().size()
                == 0) {
                    Toast.makeText(
                        MainActivity.
                        this, "Entity
                        not found",
                        Toast.
                        LENGTH_SHORT).
                        show();
            }


            adapter = new EntitysAdapter
                (getBaseContext(),
                listEntity);
            recyclerView.setAdapter(
                adapter);

                dialog.dismiss();
        }

        @Override
        public void onFailure(Call<
```

```
                ResultEntity> call,
                Throwable t) {
                 Toast.makeText(MainActivity.
                     this, "Error occured",
                     Toast.LENGTH_SHORT).show
                     ();

            }
        });

}
```

(b) EntityActivity

Pada *activity* ini berfungsi untuk memberikan informasi kepada pengguna terhadap item produk yang dipilih oleh pengguna dari hasil pencarian yang telah dilakukan. *Activity* ini menampilkan informasi seperti label, kode makanan, komposisi produk, nutrition facts dan produk terkait. Berikut ini detail kode 5.15 *EntityActivity* :

Kode 5.15: Kode untuk mengatur tampilan *activity* pada informasi produk

```
//method onCreate
Intent intent = getIntent();
atribute = intent.getParcelableExtra("
    atribute");

//jika entitas berupa produk maka
    tampilkan tablayout viewpager
if (atribute.getContainsIngredient() !=
    null) {
     setViewPager();
//jika entitas berupa komposisi produk
    tampilan viewpager disembunyikan
} else {
```

```
    hideViewPager();
    setEntityView();
}
```

2. Adapter
   (a) EntitysAdapter
   Adapter ini berfungsi untuk menampilkan data dari objek list Entity pada *RecyclerView* dan *layout* yang sudah ditentukan. Berikut ini detail kode 5.16 yang menampilkan data dari entitas ke layout pada RecyclerView :

   Kode 5.16: Kode untuk mengatur tampilan *activity* pada informasi produk

```
@Override
public void onBindViewHolder(final
    EntitysAdapter.MyViewHolder holder,
    final int position) {
    //mengambil data entity dari list
        objek
    final EntityDatum entity =
        entityList.get(position);

    holder.tv_label.setText(entity.
        getLabel() != null ? entity.
        getLabel() : "Null");
    holder.tv_containsIngredient.setText
        (entity.getAtribute().
        getContainsIngredient() != null
        ?
                "Ingredients : "+entity.
                    getAtribute().
                    getContainsIngredient
                    ().toString() : "
                    Label : "+entity.
                    getAtribute().
                    getLabel().toString
                    ());
```

```
//jika berupa produk maka diatur
    backgroud dari item
if (entity.getAtribute().
    getContainsIngredient() != null)
    {
    holder.tv_Score.setText("Type : "
        Food Product - Score : "+
        entity.getScore().toString()
        );
    holder.layout_item.
        setBackgroundColor(mContext.
        getResources().getColor(R.
        color.entity_product));
//jika berupa komposisi produk maka
    diatur backgroud dari item
} else {
    holder.tv_Score.setText("Type : "
        Ingredient - Score : "+
        entity.getScore().toString()
        );
    holder.layout_item.
        setBackgroundColor(mContext.
        getResources().getColor(R.
        color.entity_ingredient));
}

holder.itemView.setOnClickListener(
    new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //mengirim objek atribute ke
            activity berikutnya
        Atribute atribute =
            entityList.get(position)
            .getAtribute();
        Bundle bundle = new Bundle()
            ;
        bundle.putParcelable("
```

```
                atribute", atribute);
            Intent intent = new Intent(
                mContext, EntityActivity
                .class);
            intent.putExtras(bundle);
            intent.setFlags(Intent.
                FLAG_ACTIVITY_NEW_TASK);
            mContext.startActivity(
                intent);
        }
    });
}
```

(b) IngAddsAdapter

Adapter ini berfungsi untuk memberikan aksi ketika item pada daftar ingredient produk diklik akan menjalankan *activity* entitas yang menampilkan komposisi produk. Berikut ini kode 5.17 detail dari IngAddsAdapter :

Kode 5.17: Kode untuk mendapatkan data komposisi produk dari server

```
@Override
public void onBindViewHolder(final
    IngAddsAdapter.MyViewHolder holder,
    final int position) {
     final IngredientAdditives
         ingredientAdditives = ingAddList
         .get(position);

     holder.tv_name.setText(
         ingredientAdditives.getName() !=
          null ? ingredientAdditives.
         getName() : "Null");
     holder.itemView.setOnClickListener(
         new View.OnClickListener() {
          @Override
          public void onClick(View v) {
```

```
                try {
                    //menjalankan activity
                        untuk menampilkan
                        komposisi produk
                    getRetrofitSearch("rank"
                        +ingredientAdditives
                        .getId());

                } catch (Exception e) {
                    Toast.makeText(mContext,
                        "Failed to connect
                        Server", Toast.
                        LENGTH_SHORT).show()
                        ;
                }
            }
        });
}
```

(c) ReleatedAdapter

Adapter ini berfungsi untuk memberikan aksi ketika item pada daftar produk terkait diklik akan menjalankan *activity* entitas yang menampilkan produk terkait. Berikut ini detail kode 5.18 dari ReleatedAdapter :

Kode 5.18: Kode untuk mengambil data produk terkait dari server

```
@Override
public void onBindViewHolder(final
    ReleatedAdapter.MyViewHolder holder,
     final int position) {
     final Related related =
         relatedProductList.get(position)
         ;

     holder.tv_fName.setText(related.
         getFName() != null ? related.
```

```
        getFName() : "Null");
    holder.tv_similarityScore.setText("
        Similarity : " +Double.
        parseDouble(related.getCosine())
        +"%");
    holder.itemView.setOnClickListener(
        new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             try {
                 //menjalankan activity
                     untuk menampilkan
                     detail produk
                 getRetrofitSearch("
                     foodproductId"+
                     related.getId());

             } catch (Exception e) {
                 Toast.makeText(mContext,
                      "Failed to connect
                     Server", Toast.
                     LENGTH_SHORT).show()
                     ;
             }
         }
    });
}
```

(d) ViewPagerAdapter

Adapter ini berfungsi untuk menampilkan *fragment* ketika *TabLayout* di klik oleh pengguna. Ketika salah satu item di klik juga akan disertakan parameter objek atribute untuk memberikan aksi pada saat *fragment* diinisiasi. Berikut ini detail kode yang menjalankan *fragment* ketika elemen dari *TabLayout* di klik :

Kode 5.19: Kode berpindah dari tab satu ke tab lainnya

```
//mendefinisikan objek atribute
    sementara
Atribute atribute;
public ViewPagerAdapter(FragmentManager
    fm, Atribute atributeP) {
     super(fm);
     this.atribute = atributeP;
}

@Override
public Fragment getItem(int position) {

     switch (position) {
          case 0:
               return DetailEntityFragment.
                   newInstance(atribute);
          case 1:
               return
                   NutritionFactsFragment.
                   newInstance(atribute);
          case 2:
               return
                   ReleatedProductsFragment
                   .newInstance(atribute);
     }
     return null;
}
```

3. Api
   (a) RetrofitHalalAPI
   Kelas ini berfungsi untuk mendefinisikan *end-point API* dan jenis respon data pada server. Berikut ini kode detail dari kode **??** RetrofitHalalAPI :

Kode 5.20: Konfigurasi retrofit

```
//end-point pencarian
@GET("search")
```

```
Call<ResultEntity> getSearchEntity(
    @Query("q") String keyword);
//end-point produk terkait
@GET("related")
Call<List<Related>> getRelatedProduct(
    @Query("id") String id);
//end-point daftar komposisi produk
@GET("inglist")
Call<List<IngredientAdditives>>
    getIngList(@Query("id") String id);
//end-point daftar bahan additive produk
@GET("addlist")
Call<List<IngredientAdditives>>
    getAddList(@Query("id") String id);
```

4. Fragment
   (a) DetailEntityFragment
   Fragment ini berfungsi untuk menampilkan informasi detail dari entitas baik berupa produk dan daftar komposisi produk. Berikut ini kode 5.21 detail dari DetailEntityFragment :

   Kode 5.21: Kode untuk mengambil data produk kemudian ditampilkan dengn melakukan load terhadap komposisi produk

```
//mengatur textview dengan data atribute
     produk
//pada method onCreate
tv_foodid.setText(atribute.getFoodCode()
    );
tv_label.setText(atribute.getLabel());
tv_ingredients.setText(atribute.
    getContainsIngredient());
try {
     tv_info.setText(fullness());
} catch (Exception e) {
```

```
}
//menampilkan komposisi yang terkandung
    pada produk
getRetrofitIngAddList(atribute.
    getFoodproductId());
```

(b) NutritionFactsFragment
Fragment ini berfungsi untuk menampilkan informasi *nutrition fatcs* produk. Berikut ini kode 5.22 detail dari NutritionFactsFragment :

Kode 5.22: Mengatur Nutrition Facts

```
//pada method onCreate
try {
    //mengatur elemen dari nutrition
        facts dengan nilai yang terdapat
         pada objek atribute produk
    setValues();
} catch (Exception e) {

}
```

(c) RelatedProductsFragment
Fragment ini berfungsi untuk menampilkan daftar produk terkait dari sebuah produk. Berikut ini kdoe 5.23 detail dari RelatedProductsFragment :

Kode 5.23: Mencari data produk terkait dari server

```
//menampilkan daftar produk terkait pada
    recyclerview
//pada method onCreate
getRetrofitReleatedProducts(atribute.
    getFoodproductId());
```

5. Model

(a) Atribute
Model Atribute berfungsi untuk menyimpan data atribute yang didapatkan dari *end-point API* dan menjadi parameter di method lain. Model Atribute mengekstend Parcelable untuk memudahkan pengiriman objek Atribute dari *activity* satu ke *activity* lain atau method satu ke method lain.

(b) EntityDatum
Model EntityDatum berfungsi untuk menyimpan data Entity yang didapatkan dari *end-point API*.

(c) IngredientAdditives
Model IngredientAdditive berfungsi untuk menyimpan data komposisi produk maupun additive yang didapatkan dari *end-point API*.

(d) Related
Model Related berfungsi untuk menyimpan data produk terkait dari produk yang didapatkan dari *end-point API*.

(e) ResultEntity
Model ResultEntity berfungsi untuk menyimpan data hasil pencarian yang didapatkan dari *end-point API* dengan keyword tertentu.

6. Utils

(a) Params
Kelas Params berisi variabel statis untuk memudahkan konfigurasi dalam memanggil variabel yang sering digunakan dan bersifat statis seperti variabel yang digunakan sebagai *base url* untuk retrofit dan judul fragment.

5.2.8 Pengujian Stopwords

Pengujian stopwords dilakukan pada saat indexing dan pencarian. Untuk proses indexing yaitu dengan memanfaatkan term yang paling muncul pada hasil index awal kemudian membuat path baru untuk menyimpan hasil index yang akan digunakan pada saat pencarian yang menggunakan stopwords dari index awal. Berikut ini kode dari implementasi stopwords pada saat indexing 5.24 dan pencarian 5.25.

Kode 5.24: Kode indexing menggunakan stopwords

```
//jika menggunakan stopwords
if (stopwords) {
///mengambil stopwords dari term yang paling
    sering muncul di index awal
    IndexReader readerDirectoryReader.open(
        FSDirectory.open(Paths.get(indexPath)));
    CharArraySet stopSetCharArraySet.copy(
        StandardAnalyzer.STOP_WORDS_SET);
    Comparator<TermStats> comparator =
        nHighFreqTerms.DocFreqComparator();
    HighFreqTerms freqTerms = new HighFreqTerms
        ();
    TermStats[] termsLabel = freqTerms.
        getHighFreqTerms(reader, 1"label",
        comparator);
    for(int i = 0; i < termsLabel.length; ++i) {
    //menambahkan term yang paling sering muncul
         ke daftar stopwords
         stopSet.add(termsLabel[i].termtext.
             utf8ToString());
    }
    dir = FSDirectory.open(Paths.get(
        indexPathStopwords));
    //menambahkan stopwords ke analyzer
    analyzer = new StandardAnalyzer(stopSet);
```

```
} else {
    analyzer = new StandardAnalyzer();
    dir = FSDirectory.open(Paths.get(indexPath))
        ;
}
```

Kode 5.25: Kode pencarian menggunakan stopwords

```
//menentukan path indexing yang menggunakan
    stopwords
IndexReader reader;
if (stopwords) {
    reader = DirectoryReader.open(FSDirectory.
        open(Paths.get(indexStopwords)));
} else {
    reader = DirectoryReader.open(FSDirectory.
        open(Paths.get(index)));
}
IndexSearcher searcher = new IndexSearcher(
    reader);
searcher.setSimilarity(new TFIEFSimilarity());
Analyzer analyzer;
if (stopwords) {
//mengambil stopwords dari term yang paling
    sering muncul di index awal
    CharArraySet stopSet CharArraySet.copy(
        StandardAnalyzer.STOP_WORDS_SET);
    Comparator<TermStats> comparator = new
        HighFreqTerms.DocFreqComparator();
    HighFreqTerms freqTerms = new HighFreqTerms
        ();
    TermStats[] termsLabel = freqTerms.
        getHighFreqTerms(reader, 10, "label",
        comparator);
    for(int i = 0; i < termsLabel.length; ++i) {
    //menambahkan term yang paling sering muncul
        ke daftar stopwords
        stopSet.add(termsLabel[i].termtext.
```

```
                utf8ToString());
        }
        //menambahkan stopwords ke analyzer
        analyzer = new StandardAnalyzer(stopSet);
    } else {
        analyzer = new StandardAnalyzer();
    }
```

BAB 6

HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil dan pembahasan dari proses pengujian aplikasi.

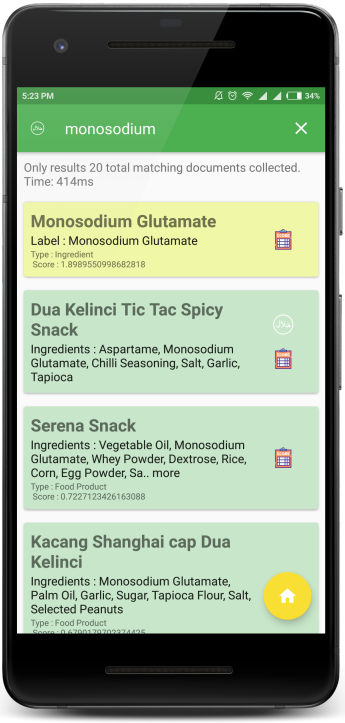
6.1 Hasil Pengujian

Pada bagian ini akan dijelaskan hasil pengujian aplikasi, baik pengujian algoritma BM25F maupun pengujian kinerja.

6.1.1 Pengujian Fungsional

Pengujian algoritma dilakukan menggunakan berbagai skenario penggunaan aplikasi dimana setiap skenario berfokus dalam menguji fungsional dan algoritma kombinasi Query-Dependent dan Query-Independent Ranking apakah menghasilkan hasil pencarian yang relevan dengan kueri yang dimasukkan pengguna serta hasil yang lebih baik dari fungsi pencarian yang dimiliki oleh aplikasi Halal Nutrition Food versi sebelumnya dinilai dengan penilaian responden melalui formulir (https://intip.in/kuesionerhalal). Untuk lebih jelasnya, berikut adalah skenario pengujian yang dilakukan:

1. Pengujian Dengan 1 Suku Kata
   Pada skenario pengujian ini dilakukan pencarian dalam aplikasi Halal Nutrition Food dengan nilai yang telah ditentukan yaitu dengan 1 suku kata. Pencarian dilakukan menggunakan kueri ”monosodium”.
   Hasil pencarian dapat dilihat seperti pada Gambar 6.1.



Gambar 6.1: Hasil pencarian uji coba algoritma query-qndependent dan query-qependent qanking dengan 1 suku kata

Penjelasan hasil pencarian teratas dari gambar tersebut adalah seperti tabel **??** sebagai berikut:

Tabel 6.1: Penjelasan nilai untuk *entity* hasil pencarian teratas kueri ”monosodium”

Dok.	Ss	Qs	Fs	Urutan	Relevansi
Monosodium Glutamate	24.0	2.14665	1.89895	1	76%
Dua Kelinci Tic Tac Spicy Snack	5.0	0.27104	0.73669	2	56%
Serena Snack	7.0	0.22379	0.72621	3	52%
Kacang Shanghai cap Dua Kelinci	5.0	0.23733	0.67901	4	52%

Penjelasan untuk entitas diatas digambarkan nilai yang terdapat pada masing-masing entitas pada tabel diatas, misalnya pada entitas ”monosodium glutamate” *static score (Ss)* diatas berasal dari jumlah produk yang memiliki komposisi ”monosodium glutamate”. Sedangkan *query-score (Qs)* berasal dari perkalian antara nilai TF-IEF yang dihitung oleh Apache Lucene dengan *normalisasi spread* yang dihitung oleh library Halal Ranker menggunakan rumus 2.3 dan 2.5. Selanjutnya *final score (Fs)* dihitung oleh library Halal Ranker menggunakan rumus 3.1. Relevansi merupakan hasil dari presentase jumlah responden yang menilai entitas dengan skor 5 (sangat sesuai) dari total jumlah responden.
Dengan hasil output JSON dari kueri tersebut seperti terlihat pada kode 6.3

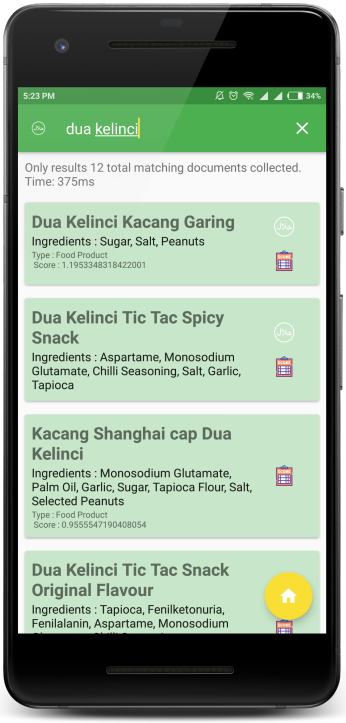
Kode 6.1: Hasil output JSON kueri satu term satu field

```
{
  "message": "Only results 20 total matching documents collected. Time:
      398ms",
  "entityData": [
    {
      "score": 1.8989550998682818,
      "label": "Monosodium Glutamate",
      "statsScore": "Ss : 24.0 - DocScore : 2.4533184 - NumbOfTerms : 2.0
          - TFIEF : 1.2266592 - Spread : 1.75 - Qs : 2.146653562784195
          - Fs : 1.8989550998682818",
      "atribute": {
        "path": "/var/www/halal/public/resources3/ingredients/
            Monosodium_Glutamate.ttl",
        "rank": "283",
        "halalSource": "283",
        "comment": "E621",
        "label": "Monosodium Glutamate",
        "sameAs": "Monosodium Glutamate"
      }
    },...
    }
  ]
}
```


Pada hasil diatas terlihat untuk hasil pencarian paling tinggi memiliki nilai *final score* 1.88816. Penjelasan nilai tersebut mengikuti rumus yang telah dijelaskan dalam rumus **??**, rumus 3.1 sebagai berikut:

$w = 1.8$
$k = 1$
$a = 0.6\, S_f = log(S_q) + w * \frac{S_s^a}{k^a + S_s^a}\; S_f = log(2.146653562784195) +$
$1.8 * \frac{22.0^{0.6}}{1^{0.6} + 22.0^{0.6}}$
$S_f = 1.888166651639039$

2. Pengujian dengan 2 Suku Kata dengan Fokus Nama Produk
   Pada skenario pengujian ini dilakukan pencarian dalam aplikasi Halal Nutrition Food dengan nilai yang telah ditentukan yaitu dengan 2 suku kata yang fokus pada nama produk. Pencarian dilakukan menggunakan kueri "dua kelinci".
   Hasil pencarian dapat dilihat seperti pada Gambar 6.2.



Gambar 6.2: Hasil pencarian uji coba algoritma query-independent dan query-dependent Ranking dengan 2 suku kata fokus pada nama produk

Penjelasan hasil pencarian teratas dari gambar tersebut adalah seperti tabel 6.3 sebagai berikut:

Tabel 6.3: Penjelasan nilai untuk *entity* hasil pencarian teratas kueri ”dua kelinci”

Dok.	Ss	Qs	Fs	Urutan	Relevansi
Dua Kelinci Kacang Garing	3.0	1.02094	1.19533	1	80%
Dua Kelinci Tic Tac Spicy Snack	5.0	0.51223	1.01312	2	76%
Kacang Shanghai cap Dua Kelinci	5.0	0.44863	0.95555	3	76%
Dua Kelinci Tic Tac Snack Original Flavour	5.0	0.42263	0.92961	4	84%

Dengan hasil output JSON dari kueri tersebut seperti terlihat pada kode 6.2

Kode 6.2: Hasil output JSON kueri dua term dengan fokus nama produk

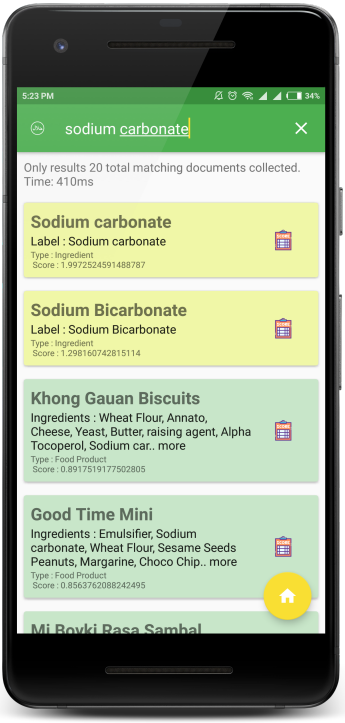
```
{
  "message": "Only results 12 total matching documents collected. Time:
      370ms",
  "entityData": [
    {
      "score": 1.1953348318422001,
      "label": "Dua Kelinci Kacang Garing",
      "statsScore": "Ss : 3.0 - DocScore : 4.9066367 - NumbOfTerms : 7.0
          - TFIEF : 0.7009481 - Spread : 1.4565217391304348 - Qs :
          1.0209461735642475 - Fs : 1.1953348318422001",
      "atribute": {
        "fiber": "3.00",
        "calcium": "0",
```

```
        "vitaminC": "0",
        "certificate": "15",
        "label": "Dua Kelinci Kacang Garing",
        "calories": "160",
        "type": "FoodProduct",
        "foodproductId": "17",
        "path": "/var/www/halal/public/resources3/foodproducts/
            Dua_Kelinci_Kacang_Garing.ttl",
        "manufacture": "17",
        "sodium": "99.90",
        "saturatedFat": "3.00",
        "netWeight": "25",
        "protein": "8.00",
        "containsIngredient": "Sugar, Salt, Peanuts",
        "fat": "12.00",
        "iron": "0",
        "vitaminA": "0",
        "sugar": "1.00",
        "foodCode": "011747233033"
      }
    }, ..
  ]
}
```


Pada hasil diatas terlihat untuk hasil pencarian paling tinggi memiliki nilai *final score* 1.19533. Penjelasan nilai tersebut mengikuti rumus yang telah dijelaskan dalam rumus **??**, rumus 3.1 sebagai berikut:

$$w = 1.8$$
$$k = 1$$
$$a = 0.6$$
$$S_f = log(S_q) + w * \frac{S_s^a}{k^a + S_s^a}$$
$$S_f = log(1.0209461735642475) + 1.8 * \frac{3.0^{0.6}}{1^{0.6} + 3.0^{0.6}}$$
$$S_f = 1.1953348318422001$$

3. Pengujian dengan 2 Suku Kata dengan Fokus Bahan-bahan Produk
   Pada skenario pengujian ini dilakukan pencarian dalam aplikasi Halal Nutrition Food dengan nilai yang telah ditentukan yaitu dengan 2 suku kata yang fokus pada bahan-bahan produk. Pencarian dilakukan menggunakan kueri ”sodium carbonate”.
   Hasil pencarian dapat dilihat seperti pada Gambar 6.3.



Gambar 6.3: Hasil pencarian uji coba algoritma query-independent dan query-dependent Ranking dengan 2 suku kata fokus pada bahan-bahan produk

Penjelasan hasil pencarian teratas dari gambar tersebut adalah seperti tabel 6.5 sebagai berikut:

Tabel 6.5: Penjelasan nilai untuk *entity* hasil pencarian teratas kueri ”sodium carbonate”

Dok.	Ss	Qs	Fs	Urutan	Relevansi
Sodium carbonate	12.0	3.37331	1.99725	1	84%
Sodium Bicarbonate	2.0	1.63554	1.29816	2	56%
Khong Gauan Biscuits	8.0	0.31141	0.89175	3	60%
Good Time Mini	8.0	0.28705	0.85637	4	60%

Dengan hasil output JSON dari kueri tersebut seperti terlihat pada kode 6.3

Kode 6.3: Hasil output JSON kueri satu term satu field

```
{
  "message": "Only results 20 total matching documents collected. Time:
       403ms",
  "entityData": [
    {
      "score": 1.9972524591488787,
      "label": "Sodium carbonate",
      "statsScore": "Ss : 12.0 - DocScore : 4.9066367 - NumbOfTerms : 2.0
           - TFIEF : 2.4533184 - Spread : 1.375 - Qs :
          3.3733127415180206 - Fs : 1.9972524591488787",
      "atribute": {
        "path": "/var/www/halal/public/resources3/ingredients/
             Sodium_carbonate.ttl",
        "rank": "230",
        "halalSource": "230",
        "comment": "E500",
        "label": "Sodium carbonate",
        "sameAs": "Sodium carbonate"
      }
    }, ..
  ]
}
```

Pada hasil diatas terlihat untuk hasil pencarian paling tinggi memiliki nilai *final score* 1.46205. Penjelasan nilai tersebut mengikuti rumus yang telah dijelaskan dalam rumus **??**, ru-

mus 3.1 sebagai berikut:

$w = 1.8$
$k = 1$
$a = 0.6$
$S_f = log(S_q) + w * \frac{S_s^a}{k^a + S_s^a}$
$S_f = log(3.3733127415180206) + 1.8 * \frac{12.0^{0.6}}{1^{0.6} + 12.0^{0.6}}$
$S_f = 1.9972524591488787$

6.1.2 Pengujian Penambahan Stopwords

Pengujian ini dilakukan untuk membandingkan hasil pencarian index yang tidak menggunakan stopwords dan index yang menggunakan stopwords seperti yang dijelaskan pada bagian 3.1.7. Berikut ini adalah datar stopwords yang berasal dari term yang paling sering muncul yang didapatkan dari index yang awal.

Tabel 6.7: Daftar term yang paling sering muncul pada index awal

Term	Field	Total Frequency	Document Frequency
rasa	label	33	33
flavour	label	33	33
pwder	label	30	30
flavor	label	24	24
milk	label	18	18
chocolate	label	18	18
identical	label	15	15
original	label	13	13
zona	label	12	12
wafer	label	12	12

Berikut adalah perbandingan hasil pencarian menggunakan antara index lama dengan index baru dengan kueri ”wafer susu” seperti tabel 6.9 dan 6.11.

Tabel 6.9: Jumlah pencarian menggunakan stopwords

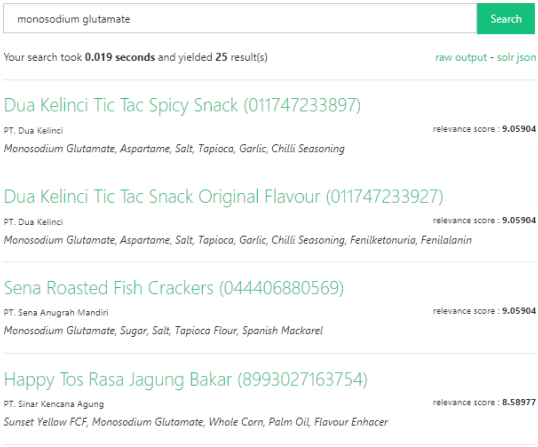
Tanpa Stopwords	Dengan Stopwords
17	6

Tabel 6.11: Perubahan skor pada hasil pencarian

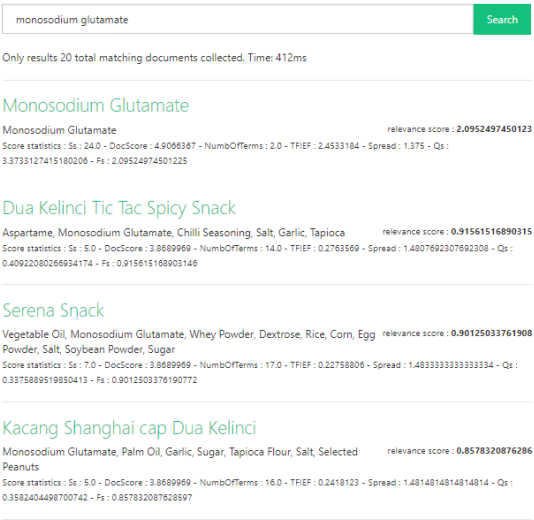
Entitas	Tanpa Stopwords	Dengan Stopwords
Tango Susu Vanilla	0.60920	0.60825
Majorico Wafer Roll Rasa Susu Vanilla	0.86430	0.56231
Susu Jahe Sidomuncul	0.50552	0.50457
Energen Susu Cereal Rasa Jahe	0.47207	0.47112
Ultra Milk Minuman Susu UHT Rasa Coklat	0.42456	0.42360
Dancow Susu Bubuk Full Cream	0.41837	0.41742

6.1.3 Pengujian Perbandingan dengan Sistem Pencarian Sebelumnya

Pengujian ini bertujuan untuk menunjukkan perbedaan yang terdapat pada hasil pencarian yang menggunakan algoritma OKAPI BM25F dengan algoritma kombinasi Query-Independent dan Query-Dependent Ranking. Berikut ini adalah hasil pencarian dengan menggunakan keyword ”monosodium glutamate” seperti gambar 6.4 dan 6.5.



Gambar 6.4: ”Hasil pencarian menggunakan OKAPI BM25F”



Gambar 6.5: Hasil pencarian menggunakan Kombinasi Query-Independent dan Query-Dependent Ranking

6.2 Pembahasan

Pada subbab ini akan dibahas dan disimpulkan hasil dari pengujian fungsional dan kinerja dari aplikasi android Halal Nutrition Food.

6.2.1 Pembahasan Uji Fungsional

Pada pengujian fungsional algoritma aplikasi telah dilakukan beberapa skenario, yakni skenario menggunakan varian kueri dengan jumlah term yang berbeda-beda yaitu pengujian pencarian kueri dengan satu term, pencarian kueri dua term dengan fokus pada nama produk dan pencarian kueri dua term dengan fokus pada bahan-bahan produk didapatkan hasil bahwa skor hasil pencarian dalam aplikasi sudah sesuai dengan kaidah algoritma *query-independent* dan *query-dependent ranking* dan menghasilkan hasil pencarian yang cocok dengan kueri yang diminta oleh pengguna. Pada pengujian pertama 6.1 dengan keyword ”monosodium”, dari 4 hasil pencarian teratas lebih dari 50% responden menyatakan bahwahasil pencarian sangat sesuai atau relevan. Pada pengujian kedua 6.3 dengan keyword ”dua kelinci”, dari 4 hasil pencarian teratas lebih dari 70% responden menyatakan bahwa hasil pencarian sangat sesuai atau relevan. Pada pengujian ketiga 6.5 dengan keyword ”sodium carbonate”, dari 4 hasil pencarian teratas lebih dari 50% mentatakan bahwa hasil dari pencarian sangat sesuai atau relevan.

Pada setiap skenario didapatkan skor tertinggi untuk entitas dengan relevansi yang tinggi sesuai dengan kueri yang diinputkan oleh pengguna. Kemudian diikuti oleh entitas entitas lain yang berkorelasi dengan masing-masing term yang ada pada queri dengan skor yang bervariasi.

Indikator terbesar yang menentukan relevansi pencarian ditentukan oleh **query-score** yang terdapat pada term itu sendiri, kemudian dilanjutkan dengan **static-score** berpengaruh besar terhadap ranking entitas. Semakin besar nilai *final-score* menunjukkan bahwa entitas pada hasil pencarian semakin relevan.

6.2.2 Pengujian Penambahan Stopwords

Pada pengujian penambahan stopwords terdapat perbedaan jumlah hasil pencarian. Pada hasil pencarian yang tanpa menggunakan stopwords terdapat 17 hasil pencarian, sedangkan hasil pencarian yang menggunakan stopwords terdapat 6 hasil pencarian. Dari hasil 6 pencarian ini mengalami penurunan skor dibandingkan skor tanpa menggunakan stopwords seperti pada tabel 6.11.

6.2.3 Pengujian Perbandingan dengan Sistem Pencarian Sebelumnya

Pada pengujian perbandingan dengan sistem pencarian menggunakan algoritma OKAPI BM25F menghasilkan 25 hasil pencarian, waktu pencarian 0,019s dan 4 hasil pencarian teratas berupa Dua Kelinci Tic Tac Spicy Snack, Dua Kelinci Tic Tac Snack Original Flavour, Sena Roasted Fish Crackers dan Happy Tos Rasa Jagung Bakar. Sedangkan sistem pencarian menggunakan algoritma kombinasi query-independent dan query-dependent ranking menghasilkan 20 hasil pencarian, waktu pencarian 0,412s dan 4 hasil pencarian teratas berupa Monosodium Glutamate, Dua Kelinci Tic Tac Spicy Snack, Serena Snack dan Kacang Shanghai cap Dua Kelinci.

Dari hasil perbedaan diatas terdapat perbedaan utama yaitu pada

algoritma OKAPI BM25F hanya menampilkan entitas produk saja, sedangkan pada algoritma query-independent dan query-dependent ranking menampilkan entitas komposisi produk dan produk dibuktikan dengan adanya ”monosodium glutamate” yang merupakan entitas komposisi produk yang menunjukkan relevansi pencarian terdapat kueri yang diinputkan oleh pengguna.

BAB 7

KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan dan saran dalam pengerjaan tugas akhir.

7.1 Kesimpulan

Berdasarkan dengan pengerjaan tugas akhir dengan judul ”Rancang Bangun Aplikasi Android Halal Nutrition Food Menggunakan Kombinasi *Query-Independent* dan *Query-Dependent Ranking*” yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut:

1. Aplikasi android Halal Nutrition Food dengan fitur pencarian produk halal yang metode skornya menggunakan algoritma *query-independent* dan *query-dependent ranking* untuk meningkatkan relevansi pencarian berhasil dibuat dan siap untuk digunakan oleh pengguna.
2. Alur pencarian dan penampilan data pada aplikasi android dilakukan ketika pengguna memasukkan kueri pencarian dan dilakukan penghitungan skor oleh library Halal Ranker pada dokumen hasil pencarian yang dihasilkan oleh Apache Lucene. Hasil pencarian akan ditampilkan melalui website berbentuk JSON dan diolah dengan aplikasi andorid untuk ditampilkan kembali kepada pengguna.
3. Dalam melakukan pencarian, aplikasi Halal Nutrition Food memiliki 4 entitas utama yaitu basis data MySQL, Apache Jena, Apache Lucene dan library Halal Ranker. Data produk halal berbentuk turtle RDF dibaca oleh Apache Jena, ke-

mudian diindex oleh Apache Lucene dan pengaturan ranking ditentukan oleh library Halal Ranker berdasarkan kueri pengguna dengan membaca skor dokumen hasil pencarian melalui index Apache Lucene berupa *query-score*. Kemudian skor ini dikombinasikan dengan skor ranking yang telah ditentukan pada basis data di MySQL berupa *static-score* yang menghasilkan *final-score*. Semakin besar nilai *final-score* pada entitas hasil pencarian menunjukkan semakin relevan entitas tersebut pada hasil pencarian.

7.2 Saran

Saran penulis untuk penelitian selanjutnya sebagai berikut:

1. Pada aplikasi android Halal Nutrition Food belum disertakan pencarian berdasarkan foodcode yang terdapat pada kemasan produk. Pencarian berdasarkan foodcode akan memudahkan pengguna dalam mencari informasi produk tertentu.
2. Pada aplikasi android Halal Nutrition Food sebaiknya disertakan pencarian produk menggunakan fitur tambahan scan barcode untuk foodcode sehingga memudahkan pengguna dalam mencari produk tanpa harus menginputkan nomor foodcode yang tertera pada kemasan.

DAFTAR PUSTAKA

[1] F. Andan, M. Rancang bangun sistem pencarian dalam aplikasi halal nutrition food menggunakan algoritma okapi bm25f.

[2] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 416–423, 2005.

[3] R. Delbru, N.A. Rakhmawati, and G. Tummarello. Sindice at semsearch 2010. *Proceedings of the 19th International World Wide Web Conference*, pages 1–3, 2010.

[4] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. Hierarchical link analysis for ranking web data. *Proceedings of the Extended Semantic Web Conference (ESWC 2010)*, 2010.

[5] Renaud Delbru. Searching web data: an entity retrieval model.

[6] Google. Gson, 2017. Dapat diakses di https://github.com/google/gson.

[7] Angga Indrawan. Inilah 10 negara dengan populasi muslim terbesar di dunia.

[8] F. Jauhar. Rancang bangun perangkat lunak linked open data halal dan gizi pada produk makanan dan minuman.

[9] Kelvin. Basic concepts lucene, 2017. Dapat diakses di http://www.lucenetutorial.com/basic-concepts.html.

[10] S. Latifi and M. Nematbakhsh. Query-independent learning to rank rdf entity results of sparql queries. *IEEE*, 2014.

[11] Guus Schreiber, VU University Amsterdam, Yves Raimond, and BBC. RDF 1.1 primer.

[12] Square. Retrofit, 2017. Dapat diakses di http://square.github.io/retrofit/.

[13] Techopedia. Semantic search, 2017. Dapat diakses di https://www.techopedia.com/definition/23731/semantic-search.

[14] Webopedia. What is semantic web? webopedia definition.

BIODATA PENULIS

Penulis lahir di Blitar pada tanggal 9 Desember 2009. Merupakan anak kedua dari 3 bersaudara dan telah menempuh pendidikan formal yaitu; MI Roudlotut Tholibin Ringinanom, MTsN Kunir Wonodadi Blitar, dan SMA Negeri 1 Srengat Blitar dan pendidikan non formal melalui Madrasah Diniyah di kampung halaman.

Pada tahun 2014 melanjutkan pendidikan di Jurusan Sistem Informasi FTIK - Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 5214100057. Selama menjadi mahasiswa penulis aktif mengikuti kegiatan ormawa Himpunan Mahasiswa Sistem Informasi dan Unit Kegiatan Mahasiswa Cinta Rebana ITS. Penulis meraih berbagai prestasi baik dalam bidang hardskill maupun softskill, antara lain Finalis Gemastik 9 UI pada cabang Pengembangan Perangkat Lunak, Juara 1 MTQ Cabang Desain Aplikasi Al-Qur'an tingkat ITS dan Penulis pernah menjadi ketua UKM terbesar di ITS yaitu UKM Cinta Rebana ITS periode 2016-2017.

Pada tahun keempat karena penulis tertarik dengan bidang desiminasi informasi, maka penulis mengambil bidang minat Laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis memiliki hobi ngoding, mengikuti mejlis dzikir dan sholawat. Penulis memiliki sebuah private project yaitu Ahmad Studio yang fokus mengembangkan berbagai aplikasi berbasis web dan android. Penulis memiliki website portofolio yang dapat diakses di http://najib.my.id dan dapat dihubungi melalui email ahmadchoirunnajib@gmail.com.