Tesis - KI142502

PERBAIKANMETODE REKOMENDASI DISKUSI PEMROGRAMAN DENGAN NORMALISASI *IDENTIFIER* MENGGUNAKAN LINGUA::IDSPLITTER

Nanang Fakhrur Rozi
5112201045

DOSEN PEMBIMBING
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Fajar Baskoro, S.Kom., M.T.

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

Thesis - KI142502

IMPROVEMENT OF PROGRAMMING DISCUSSION RECOMMENDER METHOD BASED ON LINGUA::IDSPLITTER IDENTIFIER NORMALIZATION

Nanang Fakhrur Rozi
5112201045

SUPERVISOR
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Fajar Baskoro, S.Kom., M.T.

MAGISTERPROGRAM
THE EXPERTISE FIELD OF SOFTWARE ENGINEERING
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Nanang Fakhrur Rozi
NRP. 5112201045

Dengan judul :
Perbaikan Metode Rekomendasi Diskusi Pemrograman dengan Normalisasi *Identifier* Menggunakan Lingua::IdSplitter

Tanggal Ujian : 24-6-2016
Periode Wisuda : 2015 Genap

Disetujui oleh:

Daniel Oranova Siahaan, S.Kom., M.Sc., PD.Eng.
NIP. 197411232006041001 (Pembimbing 1)

Fajar Baskoro, S.Kom., M.T.
NIP. 197404031999031002 (Pembimbing 2)

Dr. Ir. Siti Rochimah, M.T.
NIP. 196810021994032001 (Penguji 1)

Sarwosri, S.Kom., M.T.
NIP. 197608092001122001 (Penguji 2)

Rizky Januar Akbar, S.Kom., M.Eng.
NIP. 198701032014041001 (Penguji 3)

Direktur Program Pasca Sarjana,

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 196012021987011001

Halaman ini sengaja dikosongkan

PERBAIKAN METODE REKOMENDASI DISKUSI PEMROGRAMAN DENGAN NORMALISASI *IDENTIFIER* MENGGUNAKAN LINGUA::IDSPLITTER

Nama : Nanang Fakhrur Rozi
NRP : 5112201045
Pembimbing I : Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Pembimbing II : Fajar Baskoro, S.Kom., M.T.

ABSTRAK

Situs tanya-jawab Stack Overflow telah sering digunakan sebagai acuan oleh *programmer*. Informasi atau solusi dalam proses pengembangan perangkat lunak dapat dicari dengan bantuan mesin pencari pada situs. Namun, perbedaan dalam gaya penulisan, terutama pada penulisan *identifier* program, sering menyebabkan rekomendasi (pencarian) menjadi tidak sesuai dengan kebutuhan *programmer*. Beberapa *programmer* menulis *identifier* dalam bentuk singkatan sementara yang lain tidak sehingga menurunkankinerja rekomendasi. Penelitian ini mengadopsi Lingua::IdSplitter untuk menormalkan*identifier* pada data diskusi Stack Overflow. Proses normalisasi dilakukan dengan memisahkan *identifier* yang terdiri atas komposisi term serta memperluas singkatan yang ada pada*identifier*ke bentuk penuh. Hasil penelitian menunjukkan bahwa normalisasi *identifier* menggunakan Lingua::IdSplitter hanya mampumeningkatkan performa sistem rekomendasi ketika*identifier* dengan unsur singkatanmendominasi pada data diskusi.

Kata kunci: *sistem rekomendasi, identifier, normalisasi, stack overflow*

Halaman ini sengaja dikosongkan

IMPROVEMENT OF PROGRAMMING DISCUSSIONRECOMMENDER METHODBASED ONLINGUA::IDSPLITTER IDENTIFIER NORMALIZATION

Name : Nanang Fakhrur Rozi
NRP : 5112201045
Supervisor I : Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Supervisor II : Fajar Baskoro, S.Kom., M.T.

ABSTRACT

Stack Overflow as a question and answer (Q&A) site has often been used as reference by programmers. Information or solutions in the software development process can be searched with the help of search engine on the site. However, differences in writing style, especially on the program identifier writing, often causing the recommendation (search) become incompatible with the programmers need. Some programmers write identifier in abbreviated form while another are not so that it lowers the recommendation performance. This research adopted the Lingua::IdSplitter to normalize the identifier on Stack Overflow discussion. The normalization process is done by separating the identifier comprising the composition of the term as well as expand the existing abbreviation on the identifier to the full form. The results showed that the identifier normalization using Lingua::IdSplitter only able to improve the system performance when the abbreviated term within identifierdominate in the dataset.

Keywords:*recommender system, identifier, normalization, stack overflow*

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perusahaan perangkat lunak yang baik seharusnya dapat menyediakan *deliverable* dengan waktu yang singkat dan dengan kualitas yang baik pula. Untuk mencapai hal tersebut, dibutuhkan manajemen serta kerja sama yang baik dari setiap *stakeholder*,antara lain*programmer* yang menjadi tulang punggung terciptanya perangkat lunak.Dengan demikian, *programmer* dengan performa yang baik merupakan suatu keniscayaan dalam menyukseskan pengerjaan perangkat lunak.

Cegielskidan Hall (2006) menyebutkan bahwa performa*programmer* dapat diukur berdasarkan kepribadian, kemampuan kognitif, serta tingkat kepercayaan terhadap nilai teoretis *(theoretical value belief).Programmer* yang selalu mencari pembuktian kebenaran terhadap hasil kerjanya, tidak asal-asalan dalam menyediakan solusi dalam bentuk kode program,cenderung memiliki performayang baikdalam jangka panjang.Hal tersebut memaksa *programmer*, terutama *programmer* pemula,untuk menginvestasikan waktunyadalam meningkatkan kemampuan terkait pemahaman algoritma, cara memprogram, informasi-informasi pendukung, atau hal-hal lain yang berkaitan dengan perangkat lunak yang akan dibangun.Mendapatkan informasi dari anggota tim atau teman kerja menjadi salah satu caranya (LaToza dkk., 2006). Namun, hal itu akan membebani anggota lain untuk juga meluangkan waktunya yang pada akhirnya akanberimbas pada perlambatan waktu *delivery*.

Guna menghindari hal tersebut, cara lain pun ditempuh. Seiring perkembangan web 2.0, informasi di internet pun semakin beragam, informasi-informasi bergunaterkait proses pengembangan perangkat lunak yang ditenagai oleh komunitas pun tersedia, antara lain yang berbentuk situs tanya-jawab (Anderson dkk., 2012).Media tersebut pada akhirnya banyak digunakan oleh para pengembang perangkat lunakuntuk mendapatkan solusi terhadap masalah-masalah

yang sedang dihadapi.Media tersebut dapat berupa sebagian dari *platform* media sosial seperti grup di Facebook atausitus khusus seperti StackOverflow.

Stack Overflow merupakan situs tanya-jawab populer yang berfokus pada bidangpemrograman. Diskusi yang terjadi pada Stack Overflow adalah diskusi yang sangat aktif dan terpercaya.Anderson dkk. (2012) menyebutkan bahwa lebih dari 90% pertanyaan yang dilontarkan oleh penanya dijawab tidak lebih dari12 menit oleh pengguna lainnya, penanya relatif mendapatkan kepuasan atas jawaban tersebut, serta kualitas jawaban yang juga selalu mendapat perhatiandari membernya.

Programmer dapat mengakses rekomendasi informasi tersebut melalui kolom pencarian yang tersedia pada situs.Mengingat mesin pencari pada Stack Overflow juga diadopsi dari mesin pencarian teksyang umum digunakan, maka berbagai penyesuaian pun perlu dilakukan, sehingga hasil pencarian yang diberikan juga belum tentu sesuai dengan rekomendasi yang dibutuhkan pengguna. Hal tersebut umumnya diakibatkan oleh sulitnya merumuskan *query* yang tepat sebagai masukan sistem pencarian, inkonsistensi penggunaan terminologi, serta berbagai kesulitan lainnya(Manning, 2008). Oleh karena itu, penelitian-penelitian tetap dilakukan untuk menyempurnakan hasil pencarian tersebut.

Cordeiro dkk. (2012) melakukan penelitian tentang sistem rekomendasi berbasis konteks. Penelitian tersebut dilatarbelakangi oleh ketidakmampuan mesin pencari secara umum untuk memberikan rekomendasi diskusi Stack Overflow yang baik apabila*query* yang diberikan berupa informasi *stack trace*.Informasi tersebutbiasanya muncul ketika terjadi eksepsi saat menjalankan program Java. Informasi *stack trace* tersebut diasumsikan oleh peneliti sebagai konteks permasalahan *programmer*.Dalam memberikan rekomendasi,sistem akan mengekstrak *stack trace* yang muncul untuk kemudian dicari strukturnyaberdasarkan nama eksepsi, nama *method*, beserta referensinya. Struktur tersebut selanjutnya akan dijadikan sebagaiacuan*query* yang akan dibangkitkan.Rekomendasiyang dihasilkan memiliki performa yang lebih baik dibandingkan dengan hasil rekomendasi dengan *query* yang berbasis kata kunci maupun hasil rekomendasi dari mesin pencaripada situs Stack Overflow sendiri.

Namun, penelitian ini hanya akan memiliki hasil yang maksimal apabila data diskusi yang dimaksud mengandung informasi *stack trace* pula.

Penelitian lain yang telah dilakukan adalah Seahawk (Ponzanelli dkk., 2013).Seahawk merupakan sistem rekomendasidata diskusi StackOverflow yang mampu memberikan hasil rekomendasi dengan *query* yang dibangkitkan secara otomatis berdasarkan kode program yang sedang dibuka pada editor kode dalam IDEEclipse. Kode program tersebut diasumsikan sebagai teks bahasa alami sehingga *query*yang dibangkitkan akan menghasilkankumpulan kata kunci yang diperolehdari proses tokenisasi dan beberapa praproses lain terhadap kode program.Pemeringkatan data diskusi dilakukan dengan memanfaatkan Apache Solr dengan model pembobotan TF-IDF. Pengujian dilakukan menggunakan kode program Java yang telah disiapkan sebelumnya. Hasilnya, Seahawk secara umum dapat merekomendasikan diskusi yang berguna bagi *programmer*.Namun, penelitian ini masih memiliki kekurangan.Hasilrekomendasi menjadi buruk ketika kata kunci bangkitan mengandung unsur singkatan seperti `strCmp`, `usrId`, atau lainnya.Padahal, pembentukan term yang baik merupakan kunci sukses untuk mendapatkan informasi yang tepat dari literatur yang tersimpan (Krauthammer dan Nenadic, 2004).

Penelitian lain terkait sistem rekomendasi yaitu penelitian Arwan dkk. (2015) yang menghasilkanJECO (Java Example Code). Penelitian ini mengusulkan sistem rekomendasi kode program pada data diskusi Stack Overflow dengan pemodelan topik LDA (Latent Dirichlet Allocation).*Query*yang digunakan berupa kata kunci yang dimasukkan secara manual oleh pengguna. Data diskusi dan *query* kemudian dicari topik beserta proporsinyamenggunakan LDA. Berdasarkan topik tersebut, *query* dan data diskusi dihitung kesamaannya sehingga didapatkan hasil rekomendasi dengan memanfaatkan *Cosine Similarity*.Hasilnya, JECO mampu merekomendasikan kode program Java ke *programmer* dengan rata-rata *precision* sebesar 48% dan rata-rata *recall* sebesar 58%. Namun, term singkatan juga tidak diperhatikan pada penelitian ini baik pada *query* maupun data diskusi. Hal tersebut tentu akan mempengaruhi *recall* pada proses rekomendasi.

Unsur singkatan biasa muncul pada *identifier* kode program. Seorang *programmer*terkadang lebih suka menuliskan `string` dengan `str` saja pada*identifier*-nya. Sementara *programmer* lain lebih suka menulis *identifier* dalam bentuk lengkap dibandingkandengan bentuk singkatan.Sebuah penelitian yang dilakukan olehCarvalho dkk. (2015) mengembangkanLingua::IdSplitter. Algoritma tersebut memilikikemampuan untuk memisah *identifier*yang umumnya terdiri atas komposisi term serta mengekspansi term singkatan menjadi term lengkap, baik yang menggunakan model penulisan *all lowercase* maupun *camel case*. Lingua::IdSplitter dikembangkan mengingat *identifier* merupakan salah satu sumber informasi yang cukup relevan untuk memahami sebuah program. Selain itu, *identifier* yang lengkap cenderung lebih memudahkan program untuk dipahami (Lawrie dkk., 2007).Cara pemisahan *identifier* yang dilakukan Lingua::IdSplitter dilakukandengan dua tahap, yakni *hard split* dan *soft split*. *Hard split* dilakukan untuk memisahkan *identifier* berdasarkan karakter tertentu seperti tanda garis bawah *(underscore)* atau berdasarkan tata penulisan *camel case*. Sedangkan*soft split* akan memisah *identifier* yang tidak terlalu terlihat pemisahnya. Hal ini dilakukan dengan bantuan kamus kata untuk mendapatkan himpunan kandidat katanya. Kandidat kata tersebut selanjutnya akan diperingkat berdasarkan otomata yang dibangun berdasarkan himpunan kandidat kata. Himpunan kata dengan skor tertinggi akan menjadi keluaran Lingua::IdSplitter.Penelitian ini membandingkan algoritma yang diusulkan dengan algoritma-algoritma lain yang sejenis. Hasilnya, algoritma ini mampu menghasilkan term yang telah dipisah dan diekspansi dengan tingkat*f-measure* sebesar 90%, tidak jauh berbeda dengan hasil algoritma pembanding lainnya.

Penelitian ini mengadopsi Lingua::IdSplitter sebagai algoritma untuk menormalisasi *identifier* padasistem rekomendasidata diskusi dari Stack Overflowyang dibangun dengan masukan berupa kode program Java. Dengan term yang baik, maka hasil rekomendasi akan menjadi lebih baik pula. Selain itu, model data yang digunakan adalah model data Bag-of-Words(BOW) yang telah diboboti menggunakan algoritma TF-IDF.Model tersebut dipilih mengingat penggunaan LDA untuk memodelkan topik masih memiliki kekurangan karena

tidak adanya proses seleksi atau pemilihan term penting (Hu, 2009) sehingga akan mengakibatkan ketidaksesuaianproporsi topik terhadap dokumen.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana membangun sistem rekomendasi diskusi pemrograman dengan masukan berupa kode program?
2. Bagaimana hasil metode rekomendasi diskusi pemrograman dapat diperbaiki dengan normalisasi *identifier*menggunakan Lingua::IdSplitter?

1.3 Batasan Masalah

Agar penelitian yang dilakukan dapat lebih terarah dan sesuai dengan tujuan, maka perlu dibatasi ruang lingkup dalam penelitian ini. Adapun batasan masalah dalam penelitian ini yaitu:

1. Masukan yang dapat diproses adalah kode program Java dengan *identifier*yang dibentuk dari kata-kata bahasa Inggris.
2. Data diskusi yang diambil adalah data diskusi yang memiliki label Java saja mengingat data masukan pada proses rekomendasi juga berupa kode program dengan bahasa Java.

1.4 Tujuan Penelitian

Tujuan penelitian ini yaitumembangun dan memperbaikisistem rekomendasi diskusi pemrograman dengan melakukan normalisasi *identifier* menggunakan Lingua::IdSplitter.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini yaitu dihasilkannya sistem rekomendasi diskusi pemrograman yang relevan yang mampu mendukung para *programmer* dalam menemukan solusi-solusi dari Stack Overflow terkait permasalahan pengembangan perangkat lunak.

1.6 Kontribusi Penelitian

Kontribusi penelitian ini adalah perbaikan term indeks dan *query* pada sistem rekomendasi diskusi pemrograman dengan melakukan normalisasi *identifier* menggunakan Lingua::IdSplitter.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Sistem Rekomendasi

Sistem rekomendasi merupakan metode yang mampu memberikan rekomendasi dengan memprediksi nilai sebuah item bagipengguna untuk kemudian mempresentasikan item dengan nilai prediksi tertinggi sebagai hasil rekomendasi. Sistem ini awalnya merupakan sebuah mekanisme *information filtering*, yakni bertujuan menyaring informasi sebagai akibat membludaknya informasi di internet. Terdapat tiga pendekatan yang bisa digunakan dalam membangun sistem rekomendasi, yaitu rekomendasi berbasis konten, rekomendasi berbasis metode kolaboratif, dan rekomendasi berbasis metode hibrida (Adomavicius dan Tuzhilin, 2005).

2.1.1 Rekomendasi Berbasis Konten

Jenis sistem rekomendasi ini memiliki akar keilmuan yang berasal dari ilmu temu kembali informasi *(information retrieval)*. Pada rekomendasi berbasis konten, pengguna memasukkan beberapa informasi ke dalam sistem, dalam hal ini disebut *query*, misalnyapermasalahan yang sedang dihadapi. Sistem kemudian mengubah *query* tersebut ke model tertentuuntuk kemudian digunakan oleh sistem dalamkomputasi untuk menyaring begitu banyak informasi yang tersediadan akan menghasilkan rekomendasi untuk disajikan kepadapengguna.

2.1.2 Rekomendasi Berbasis Metode Kolaboratif

Jenis sistem rekomendasi ini juga sering disebut *collaborative filtering system*. Model rekomendasi ini memanfaatkan data yang dimiliki oleh pengguna lain untuk membangun model. Artinya preferensi dari pengguna lain yang memiliki relasi ke pengguna yang ingin mendapat rekomendasi akan mempengaruhi hasil rekomendasinya.

2.1.3 Rekomendasi Berbasis Metode Hibrida

Jenis sistem rekomendasi ini menggabungkan karakteristik yang ada pada sistem rekomendasi berbasis konten dengan sistem rekomendasi berbasis metode kolaboratif. Model rekomendasi ini dikembangkan guna membantu menghindari batasan-batasan yang ada pada kedua jenis sistem rekomendasi tersebut.

Pada penelitian ini, model sistem rekomendasi yang digunakan adalah model rekomendasi berbasis konten. Model kolaboratif tidak digunakan mengingat model kolaboratif membutuhkan informasi dari pengguna lain yang dalam hal ini tidak tersedia. Selain itu, pemilihan model juga didasarkan pada penelitian-penelitian terdahulu yang digunakan sebagai acuan peneliti.

2.2 *Identifier*pada Kode Program

MenurutGosling dkk. (2015), *identifier*dalam ranah pemrograman, khususnya bahasa pemrograman Java, adalah rangkaian karakter Java dan digit Java dengan panjang tak hingga.Syaratnya, karakter pertama pada *identifier* harus berupa karakter Java. Karakter Java sendiri merupakan karakter alfabet (non-simbol), yakni karakter-karakteryang biasa digunakan untuk merangkai kata, baik yang terdapat dalam kumpulan karakter latin maupun non-latin. Contoh karakter non-latindapat ditemukanpada tulisan-tulisan berbahasaArab, Cina, Jepang, Korea, atau lainnya.Sementara itu, digit Java merupakan angka 0–9, baik dalam bentuk karakter latin maupun non-latin.Sebagai tambahan, karakter-karakter yang digunakan oleh Java didasarkan pada standar pengodean *unicode*.

*Identifier*pada java biasa digunakan untuk menamai elemen-elemen di dalam kode program, seperti *class*, variabel, atau *method* (Vohra dkk., 2015).Pada umumnya, penulisannya mengacu pada konvensi yang dibuat oleh Sun Microsystems (1997), yakni menggunakan aturan *camel-case* atau *underscore*seperti pada penulisan variabel `myWidth` dan `MAX_WIDTH`. Selain itu, penulisan *identifier* bersifat *case-sensitive*, yakni membedakan huruf besar dan kecil, misalnya membedakan antara variabel`var2a` dan `var2A`. Namun, dalam penelitian ini, konsep sensitivitas karakter tersebut diabaikan. Langkah tersebut

diambil mengingat perbedaan *case* tidak akan membedakan makna atau konsep*identifier*.

Mengingat *identifier*umumnya berupa gabungan kata yang mengakibatkan panjangnya *identifier*, maka ada*programmer*yangmenggunakan unsur singkatan dalam membentuk*identifier*. Singkatan-singkatan tersebut antara lain singkatan-singkatan yang umum digunakan seperti HTML, URL, atau singkatan-singkatan dalam bahasa tertentu.

2.3 Sistem Rekomendasi dalam Rekayasa Perangkat Lunak

Sistem rekomendasi dalam rekayasa perangkat lunak atau sering disebut *Recommendation Systems for Software Engineering* (RSSE) difungsikan untuk membantu pengembang dalam berbagai aktivitas, mulai dari sugesti informasi, penggunaan ulang kode program, hingga penulisan laporan bug yang efektif (Robillard dkk., 2010).Dengan semakin kompleksnya proses pengembangan perangkat lunak, maka sistem semacam ini sangat dibutuhkan oleh para pengembang, terlebih untuk pengembang dengan tingkat pengalaman yang masih rendah. Guna mendukung hal tersebut, berbagai penelitian telah dilakukan.

Salah satu topik RSSE yang telah diteliti yaitu sistem rekomendasi informasi, khususnya berupa data forum diskusi (tanya-jawab). Penelitian terhadapdata diskusi dilakukan mengingat informasi tersebut merupakan salah satu sumber informasi yang penting dan layak untuk dipertimbangkan sebagai sumber rekomendasi.Kelayakan tersebut didasarkan pada penelitian Anderson dkk. (2012) yang menyatakan bahwa berbagai situs tanya-jawab yang ditenagai oleh komunitas saat ini dapat dijadikan sebagai sumber informasi mengingat mekanisme reputasi ataupun mekanisme-mekanisme lain yang mendukung kekuatan dan validitas situs tanya-jawab tersebut. Studi kasus yang digunakan pada penelitian tersebut adalah situs tanya-jawab Stack Overflow.

Penelitian ini disusun dengan berlandaskan pada penelitian-penelitian terdahulu yang sudah pernah dilakukan. Penelitian-penelitian tersebut adalah penelitian yang berkaitan dengan topik sistem rekomendasi data diskusi pemrograman yang berasal dari Stack Overflow yang memanfaatkan model rekomendasi berbasis konten. Sistem rekomendasi tersebut dirancang untuk

membantu para pengembang, khususnya *programmer*, dalam menyelesaikan permasalahan-permasalahanketika mengembangkan program.

Cordeiro dkk. (2012) telah melakukan penelitian tentang sistem rekomendasi berbasis konteks. Konteks permasalahan programdilihat dari informasi *stack trace* yang umumnya muncul saat terjadi eksepsi ketika program dijalankan. Informasi tersebut merupakan informasi yang penting karena berkaitan erat dengan permasalahan yang sedang terjadi. Akan tetapi, mesin pencari yang umum digunakan belum mampu menghasilkan rekomendasi solusi atas permasalahan program jika masukannya berupa informasi *stack trace*. Dalam mendapatkan hasil rekomendasi, informasi *stack trace* yang muncul akan diekstrak sehingga didapatkan informasi eksepsi beserta referensinya yang dibentuk dalam sebuah struktur. Struktur tersebut selanjutnya akan diubah menjadi representasi leksikal berdasarkan nama eksepsi dan nama referensi menggunakan aturan penulisan *camel case*. Sebelum diubah ke dalam representasi leksikal, informasi referensi tersebut akan diboboti terlebih dahulu guna mengetahui referensi-referensi yang paling relevan terhadap eksepsi. Lima term dengan bobot tertinggi akan digunakan sebagai *query* pada sistem rekomendasi. Hasil penelitian menunjukkan bahwa pembentukan *query* yang berasal dari informasi *stack trace* ini mampu menghasilkan rekomendasi yang lebih baik dibandingkan dengan mesin pencarian yang umum digunakan serta fasilitas pencarian yang telah disediakan oleh Stack Overflow dengan masukan berupa informasi *stack trace*. Adapun proses pengindeksannya, penelitian ini memanfaatkan Apache Lucene. Lebih jauh, penelitian ini masih memiliki kelemahan, yakni hasil rekomendasi hanya terbatas pada data-data diskusi yang mengandung informasi *stack trace*saja. Dengan demikian, informasi yang direkomendasikan juga akan menjadi sangat sempit, padahal terkadang permasalahan-permasalahan yang terjadi membutuhkan dukungan pengetahuan yang lebih luas. Selain itu, karena proses representasi leksikalnya menggunakan aturan *camel case*, maka performa rekomendasi juga akan menurun apabila ditemukan *method* atau referensi yang tidak menggunakan aturan *camel case*.

Penelitian berikutnya adalah penelitian yang dilakukan oleh Ponzanelli dkk. (2013). Penelitian tersebut mengembangkan sebuah sistem rekomendasi

dengan konteks permasalahan yang didasarkan pada kode program yang sedang terbuka pada IDE Eclipse. Sintaks-sintaks pada program tersebut selanjutnya akan diekstrak sehingga didapatkan kumpulan term. Kumpulan term yang telah didapatkan direpresentasikan dalam bentuk vektor terbobot berdasarkan pembobotan TF-IDF. Sebelum dibentuk representasi vektornya, akan dilakukan praproses pada term-term yang telah diperoleh. Beberapa praproses yang dilakukan antara lain penghapusan *stop word*, *stemming*, filter sinonim, dan pengubahan term ke bentuk*lower case*.Sepuluh term dengan frekuensi tertinggi akan dijadikan sebagai *query* untuk menghasilkan rekomendasi.Penelitian ini mampu memberikan rekomendasi terutama untuk informasi-informasi yang bersifat spesifik. Namun, penelitian ini memiliki kekurangan. Jika term hasil ekstraksi mengandung unsur singkatan, hasil rekomendasinya menjadi buruk. Kelemahan lainnya yaitu ketidakmampuan sistem untuk merekomendasikan informasi yang relevan ketika kode program tidak menggunakan standar-standar yang sering digunakan seperti *framework* atau implementasi buruk dari pengembang.

Penelitian lainnya yaitu penelitian yang dilakukan oleh Arwan dkk. (2015). Masukan dalam sistem rekomendasi pada penelitian ini berupa *query*yang dimasukkan pengguna secara manual. Model data yang digunakan berbeda dengan kedua penelitian sebelumnya. Penelitian ini memanfaatkanpemodelan topik untuk merepresentasikan data diskusi yang akan direkomendasikan.Algoritma pemodelan topik yang digunakan adalah Latent Dirichlet Allocation (LDA).Setiap dokumen memungkinkan untuk memiliki lebih dari satu topik bahasan. Tiap topik yang melekat pada dokumen akan memiliki proporsi yang berbeda, satu topik bisa memiliki porsi yang lebih besar dibandingkan topik lainnya. Porsi topik yang melekat pada dokumen tersebut yang pada akhirnya akan dijadikan sebagai fitur dokumen yang juga dirupakan dalam model ruang vektor sebagaimana dua penelitian sebelumnya.Penentuan porsi topik tersebut juga akan dilakukan pada *query* pengguna, sehingga pada akhirnya dapat ditentukan kedekatan *query* dengan data diskusi yang ada menggunakan perhitungan Cosine Similarity. Hasil rekomendasi mampu menghasilkan rata-rata *precision* sebesar 48% dan rata-rata *recall* sebesar 58%.

Pemodelan topik digunakan pada penelitian ini untuk merepresentasikan dokumen dengan tujuan menangani permasalahan kesamaan topik antara satu term dengan term lain yang terkadang tidak dapat diidentifikasi oleh model vektor yang berbasis TF-IDF.Akan tetapi, pemodelan topik juga masih memiliki kekurangan antara lain tidak adanya pemilihan term penting yang mengakibatkan ketidaktepatan proporsi topik terhadap dokumen. Untuk memperjelas,Tabel 2.1 akan memberikan ringkasan singkat tentang ketiga penelitian yang telah diambil sebagai dasar dalam penelitian ini.

Tabel 2.1 Ringkasan Perbandingan Penelitian yang Diacu

Komponen Penelitian	**Penulis**		
	Cordeiro dkk. (2012)	**Ponzanelli dkk. (2013)**	**Arwan dkk. (2015)**
Judul	*Context-Based Recommendation to Support Problem Solving in Software Development*	*Leveraging Crowd Knowledge for Software Comprehension and Development*	*Source Code Retrieval on StackOverflow Using LDA*
Model Data	Model vektor (bobot TF-IDF)	Model vektor (bobot TF-IDF)	Model vektor (proporsi topik LDA)
Praproses	Praproses bawaan Apache Lucene	Praproses bawaan Apache Solr	Praproses umum mesin pencaridan *identifier splitting*
Simpulan	Hasil lebih baik dibandingkan dengan mesin pencari umum dan fasilitas pencarian Stack Overflow	Hasil rekomendasi secara umum baik	Rata-rata *precision* 48% dan rata-rata *recall* 58%

Ketiga penelitian tersebut belum ada yang melakukan normalisasi *identifier* atau secara khusus mengekspansi *identifier* baik pada data maupun *query* yang akan mengakibatkan turunnya*recall* hasil rekomendasi. Hal tersebut juga mengingat akan pentingnya pembentukan term yang baik (Krauthammer dan Nenadic, 2004).

2.4 Praproses pada Sistem Rekomendasi

Praproses merupakan salah satu hal yang lumrah dilakukan sebelum sebuah data diproses lebih lanjut. Praproses ini digunakan untuk menghapus informasi-informasi yang tidak berguna ataupun menormalisasi data agar data menjadi bersih atau sesuai format serta siap untuk diproses. Mengingat data yang

akan dibahas mengandung kode program, maka praproses yang dilakukan salah satunya harus mempertimbangkan pola penulisan kode program (Mahmoud dan Niu, 2011).

2.4.1 Tokenisasi, Filter Simbol, dan *Case Folding*

Jika diberikan kumpulan karakter dan pendefinisian unit dokumen, tokenisasi adalah proses pemotongan kumpulan karakter tersebut menjadi potongan-potongan yang disebut token.Gambar 2.1merupakan contohdari tokenisasi. Lebih dari itu, duplikasi token yang telah ditiadakan (token unik) sering juga disebut sebagai term pada bidang ilmu temu kembali informasi (Manning dkk., 2008). Mengiringi tokenisasi, penghapusan simbol atau data non-tekstual biasanya juga akan dilakukan. Pada kode program, simbol ini dapat berupa tanda plus (+), minus (-), modulus (%), atau simbol lain seperti @ yang biasa digunakan untuk memberi anotasi pada kode program (Mahmoud dan Niu, 2011). Sebagai tambahan, pengubahan huruf besar ke huruf kecil *(case folding)* juga biasa dilakukan untuk menyamaratakan karakter ke bentuk huruf kecil saja.

Masukan
even though backquote (') is not a standard quote character in
Keluaran
Even though backquote is not a standard quote character in

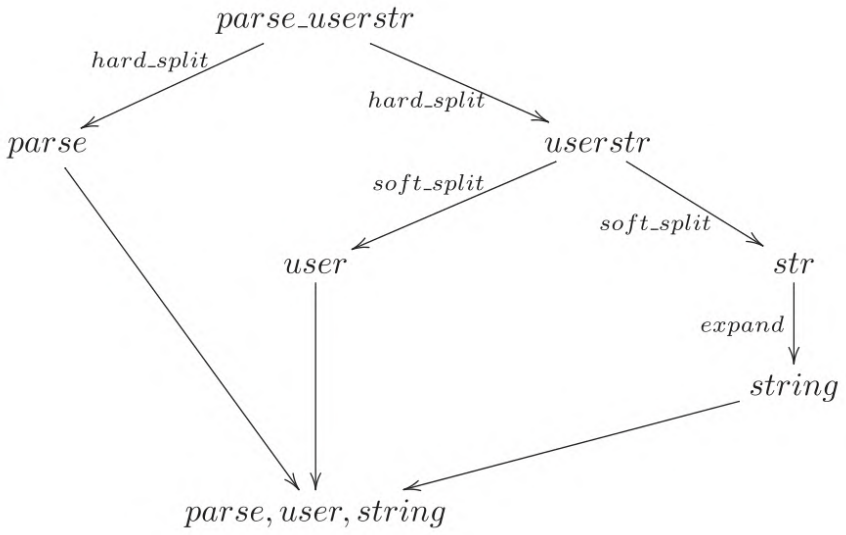
Gambar 2.1Hasil Tokenisasi, Filter Simbol, dan Case Folding pada Data Teks

2.4.2 Normalisasi dengan Lingua::IdSplitter

Lingua::IdSplitter merupakan sebuah algoritma yang dikembangkan oleh Carvalho dkk. (2015). Algoritma tersebut memiliki kemampuan untuk

memisahkan *identifier* kode program yang umumnya terdiri atas komposisi term serta mengekspansi term singkatan menjadi term lengkap, baik yang menggunakan model penulisan *all lowercase* maupun *camel case*. Lingua::IdSplitter dikembangkan mengingat *identifier* merupakan salah satu sumber informasi yang cukup relevan dalam memahami sebuah program. Selain itu, *identifier* yang lengkap cenderung lebih memudahkan program untuk dimengerti (Lawrie dkk., 2007).

Cara pemisahan *identifier* pada Lingua::IdSplitter dilakukan melalui dua tahap, yakni *hard split* dan *soft split*. *Hard split* dilakukan untuk memisahkan *identifier* berdasarkan karakter tertentu seperti tanda garis bawah *(underscore)* atau berdasarkan aturan penulisan*camel case*. Sementara itu, *soft split* akan memisahkan *identifier* yang tidak terlalu terlihat (tanda) pemisahnya. Ilustrasi proses pemisahan dan ekspansi ini dapat dilihat pada Gambar 2.2.



Gambar 2.2Mekanisme Pemisahan dan Ekspansi *Identifier* (Carvalho dkk., 2015)

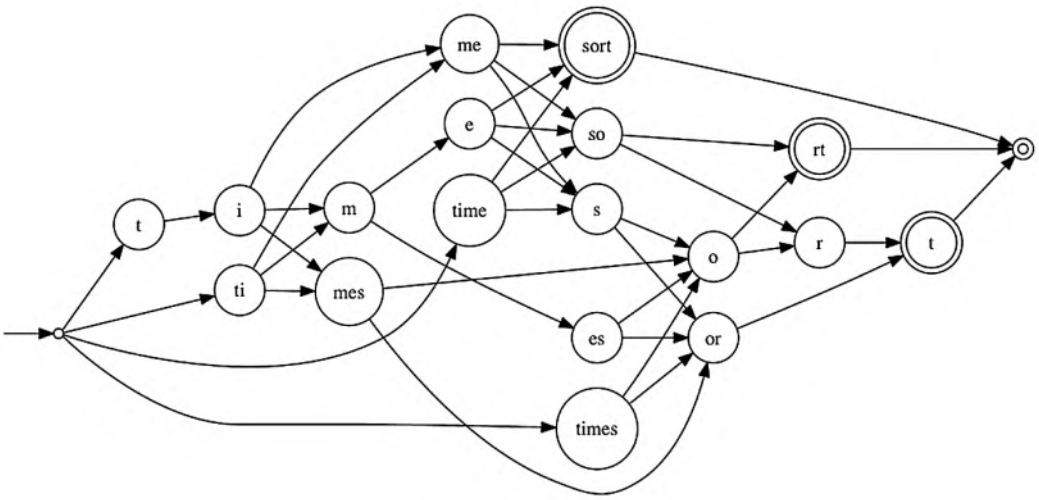
Soft split dilakukan salah satunya dengan bantuan kamus kata (akronim dan singkatan) untuk mendapatkan himpunan kandidat katanya. Kamus kata yang digunakan dapat dilihat pada Lampiran 1. Apabila kandidat kata tersebut tidak ditemukan dalam kamus, maka Lingua::IdSplitter akan memotong identifier dengan jumlah karakter tertentu untuk kemudian dicek validitas katanya. Kandidat kata tersebut selanjutnya akan diperingkat berdasarkan otomata yang dibangun

berdasarkan himpunan kandidat kata. Himpunan kata dengan skor tertinggi akan menjadi keluaran Lingua::IdSplitter. Ilustrasi hasil penskoran himpunan kandidat kata dapat dilihat pada Tabel 2.2. Sedangkan otomata yang dibangun dapat dilihat padaGambar 2.3.

Tabel 2.2Himpunan Kandidat Hasil Split Berdasarkan Skor Tertinggi

Kandidat *Split*	**Skor**
{time, sort}	1,4400
{ti, me, sort}	0,1920
{time, so, rt}	0,1920
{times, o, rt}	0,1500
{…}	…

Sumber: Carvalho dkk. (2015)



Gambar 2.3Otomata untuk Identifier *timesort* (Carvalho dkk., 2015)

Otomata padaGambar 2.3 digunakan untuk menghitung skor kandidat hasil pemisahan. Berdasarkan penghitungan skor kandidat kata pada Tabel 2.2, didapatkan bahwa kandidat term *time+sort* adalah term yang paling baik untuk menjadi hasil pemisahan atas term *timesort*.

Adapun normalisasi Lingua::IdSplitteryang digunakan dalam penelitian ini merupakan program berbahasa Perl yang telah tersedia secara bebas. Program ini dapat difungsikan sebagai pustaka pemrograman (Perl) atau dijalankan sebagai program eksternal melalui antarmuka baris perintah *(command line interface)* yang tersedia.

2.4.3 Penghapusan Stop Word

Penghapusan*stopword* adalahproses penghapusan term yang tidak memiliki arti, tidak relevan, atau term yang sangat umum. Termyang diperoleh dari tahap tokenisasi dicek dalam suatu *stoplist,*apabila sebuah kata masuk di dalam *stop list* maka kata tersebuttidak akan diproses lebih lanjut. Sebaliknya, apabila kata tidaktermasuk di dalam *stop list* maka kata tersebut akan masuk pada prosesberikutnya. *Stop list* tersimpan dalam suatu dokumen yang dimuat saat pemrosesan term.*Stop list* yang digunakan diambil dari MySQL (2006) yang memuat sebanyak 571 kata bahasa Inggris. Adapun daftar lengkapnya dapat dilihat pada Lampiran 2.

2.4.4 Stemming

Stemming dilakukan atas asumsi bahwa kata-kata yang memiliki stem yang sama memiliki makna yang serupa sehingga pengguna tidak keberatan untuk memperoleh dokumen-dokumen yang di dalamnya terdapat kata-kata dengan stem yang sama dengan *query*-nya. Teknik-teknik *stemming* dapat dikategorikan menjadi tiga, yaitu berdasarkan aturan bahasa tertentu, berdasarkan kamus, dan berdasarkan kemunculan bersama.

Proses ini memiliki dua tujuan. Dalam hal efisiensi, *stemming* mengurangi jumlah kata-kata unik dalam indeks sehingga mengurangi kebutuhan ruang penyimpanan untuk indeks dan mempercepat proses pencarian. Dalam hal keefektivan, *stemming* meningkatkan *recall* dengan mengurangi bentuk-bentuk kata menjadi bentuk dasarnya atau *stem*-nya. Sehingga, dokumen-dokumen yang menyertakan suatu kata dalam berbagai bentuknya memiliki kecenderungan yang sama untuk ditemukembalikan. Hal tersebut tidak akan diperoleh jika tiap bentuk kata disimpan secara terpisah dalam indeks. Adapun *stemmer* yang paling umum digunakan terhadap dokumen berbahasa Inggris adalah Porter Stemmer yang berbasis aturan bahasa Inggris (Manning, 2008). Meski masih memiliki kekurangan, namun metode *stemming* ini adalah metode yang cukup ringan dengan hasil yang cenderung lebih baik dibanding metode lain yang sejenis. Sedangkan hasil *stemming* berdasarkan kemunculan bersama akan sangat dipengaruhi oleh koleksi kata yang didapatkan dari dokumen. Sehingga, kata yang

tidak pernah muncul pada dokumen tidak akan bisa diperoleh *stem*-nya (Jivani, 2011). *Stemming* berdasarkan kamus tidak digunakan mengingat hasil berupa stem sudah mencukupi pada studi kasus yang diambil tanpa mengubah stem ke bentuk lema (kata valid pada kamus).

2.5 Perekomendasian Berdasarkan Similaritas Data

Perekomendasian baru bisa dilakukan setelah praproses dilakukan. Dengan masukan berupa daftar term, kemudian term tersebut diboboti dengan metode pembobotan umum TF-IDF. Algoritma inimenggabungkan dua konsep penghitungan bobot, yaitu frekuensikemunculan kata di dalam sebuah dokumen tertentu atau biasa disebut TF*(term frequency)*dan inversfrekuensi dokumen, yaitu frekuensi dokumen yang mengandung kata tersebut atau biasa disebut IDF*(inverse document frequency)*.Bobot TF dan IDF kemudian digabungkan dengan cara dikalikan sebagaimana pada rumus (2.1) untuk mendapatkan bobot komposit berdasarkan pertimbangan kedua kriteria tersebut (Manning, 2008).

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \text{IDF}_t \tag{2.1}$$

dengan:

$\text{TF}_{t,d}$ = frekuensi kemunculan term *t* pada dokumen *d*

IDF_t = invers frekuensi dokumen untuk term *t* yang dihitung menggunakan rumus (2.2)

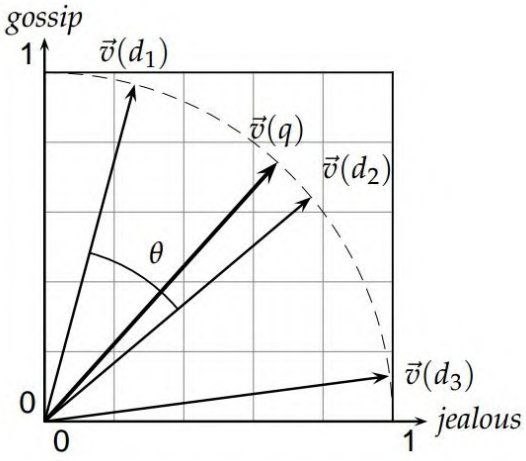
$$\text{IDF}_t = 1 + \log\frac{\text{N}}{\text{DF}_t + 1} \tag{2.2}$$

dengan:

N = total dokumen pada koleksi/korpus

DF_t = jumlah dokumen yang mengandung term *t*

Pembobotan setiap term yang telah dilakukanakandirepresentasikan dalam ruang vektor *(vector space model)*dalam bentuk matriks term-dokumen sehinggasetiap dokumen akan berbentuk sebagai sebuah fitur vektor. Vektor-vektor tersebut yang kemudian akan digunakan untuk menghitung kemiripan antardokumen dengan pendekatan *cosine similarity*. Ilustrasi *cosine similarity* dapat dilihat pada Gambar 2.4.



Gambar 2.4Penentuan Kesamaan Dokumen (Vektor) Berdasarkan Sudut yang Terbentuk (Manning, 2008)

Tiap anak panah pada Gambar 2.4mengilustrasikan dokumen yang direpresentasikan dalam bentuk vektor. Semakin kecil sudut yang terbentuk antara dua vektor, maka semakin mirip kedua dokumen tersebut. Adapun rumus yang digunakan adalah:

$$sim(d_1, d_2) = \frac{\overrightarrow{d_1} \cdot \overrightarrow{d_2}}{\left|\overrightarrow{d_1}\right|\left|\overrightarrow{d_2}\right|} \tag{2.3}$$

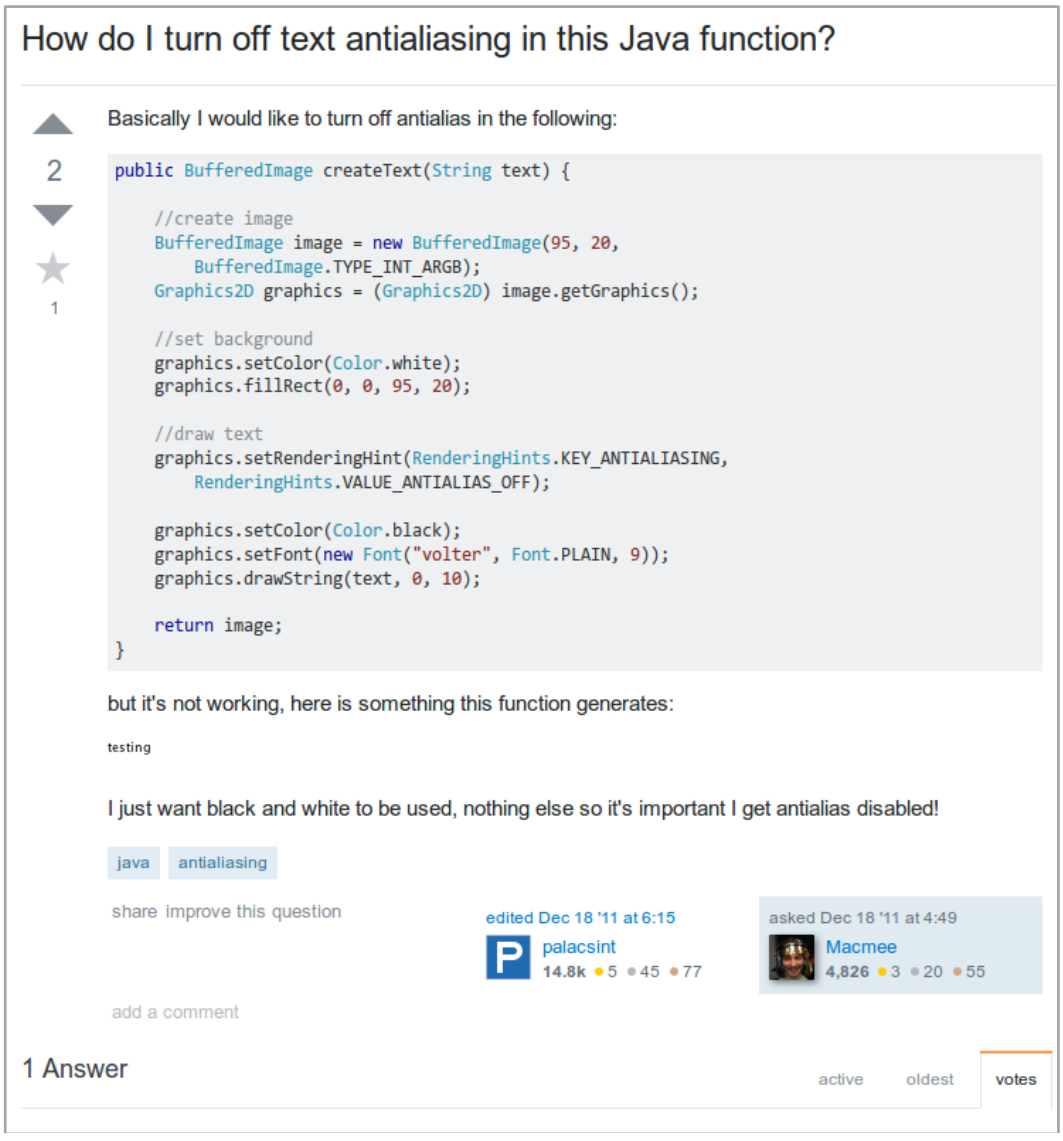
dengan pembilang berupa operasi *dot product* antara vektor pertama dan vektor kedua. Sedangkan penyebut berupa jarak Euclidian antara kedua vektor.

2.6 Stack Overflow Sebagai Sumber Rekomendasi

Stack Overflow merupakan situs tanya-jawab populer yang berfokus pada bidang pemrograman. Diskusi yang terjadi pada Stack Overflow adalah diskusi yang sangat aktif dan terpercaya. Anderson dkk. (2012) menyebutkan bahwa lebih dari 90% pertanyaan yang dilontarkan oleh penanya dijawab tidak lebih dari 12 menit oleh pengguna lainnya, penanya relatif mendapatkan kepuasan atas jawaban tersebut, serta kualitas jawaban yang didapat juga selalu mendapat perhatian dari membernya. Ilustrasi diskusi pada Stack Overflow dapat dilihat pada Gambar 2.5.

Diskusi pada Stack Overflow umumnya berupa data teks. Selain itu, pengguna umumnya mencantumkan potongan kode program pada pertanyaan

ataupun jawaban manakala dibutuhkan. Dan terkadang, pengguna juga mencantumkan gambar yang mampu mendukung pemahaman terhadap pertanyaan ataupun jawaban pengguna. Selain bertanya dan menjawab, pengguna Stack Overflow diberi fasilitas untuk melakukan *vote* terhadap pertanyaan maupun jawaban. Pertanyaan yang mendapat banyak *vote* dapat dikategorikan sebagai pertanyaan yang penting dan merupakan permasalahan yang sering terjadi. Sedangkan jawaban dengan *vote* yang banyak menandakan jawaban tersebut merupakan jawaban yang sangat bisa dipertimbangkan sebagai solusi akhir yang dapat diterima oleh penanya. *Vote* pada Gambar 2.5 dirupakan dengan ikon segitiga menghadap ke atas *(vote up)* dan ke bawah *(vote down)*.



Gambar 2.5Ilustrasi Diskusi pada Stack Overflow (Zorychta, 2011)

2.7 Evaluasi Sistem Rekomendasi

Karena model sistem rekomendasi yang diteliti adalah rekomendasi berbasis konten, maka sistem dapat dievaluasi dengan menggunakan perhitungan *precision* dan *recall*(Manning, 2008) berdasarkan *query* yang dimasukkan.*Precision*merupakan rasio jumlah dokumen relevan yang dikembalikan terhadapjumlah seluruh hasil yang dikembalikan. Rumusnya sebagaimana berikut.

$$\text{Precision} = \frac{\#(\text{dokumen relevan yang dikembalikan})}{\#(\text{dokumen yang dikembalikan})} \quad (2.4)$$

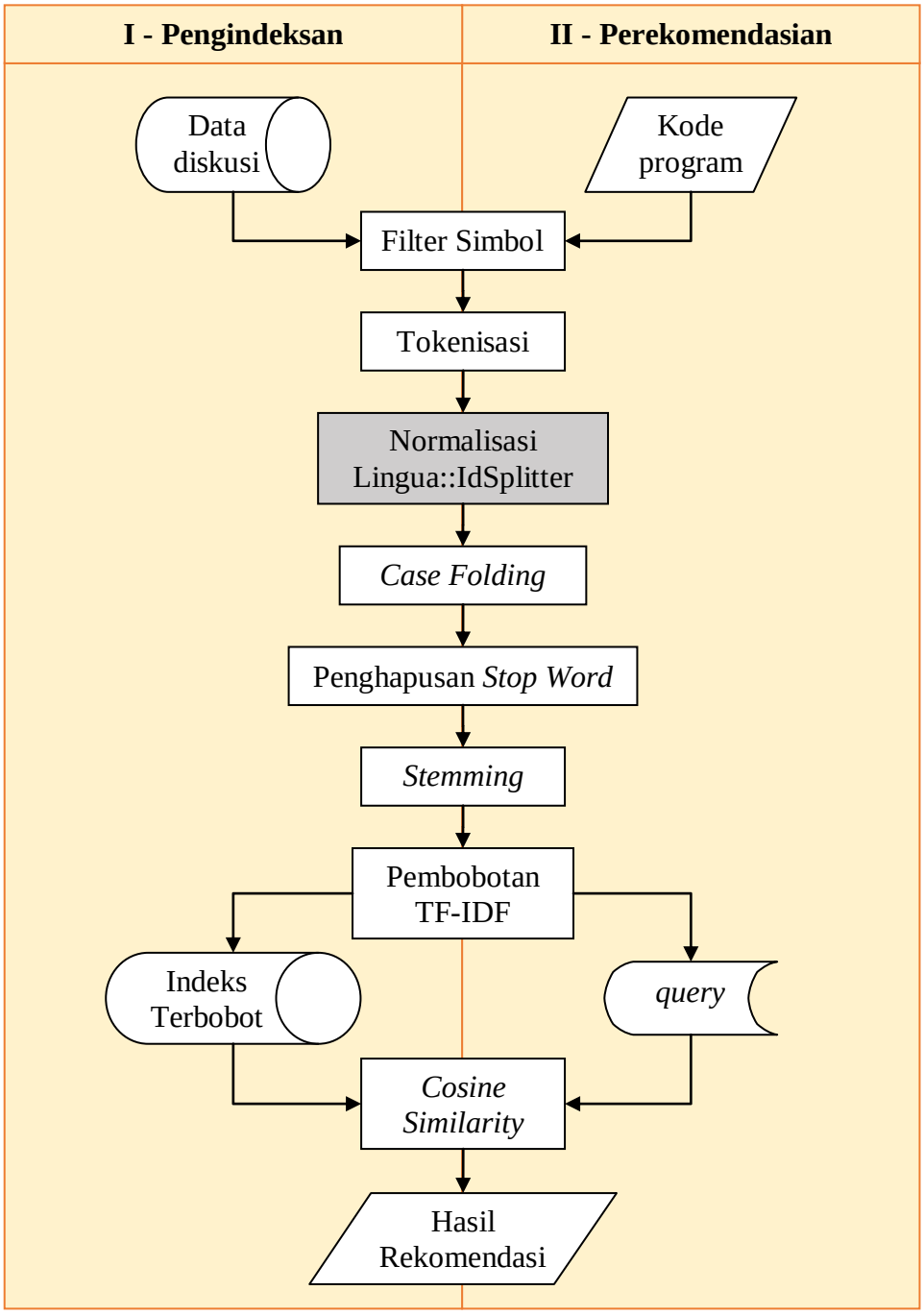
Sedangkan*Recall* merupakan rasio jumlah dokumen relevan yang dikembalikan terhadap jumlah seluruh dokumen yang relevan. Rumusnya adalah sebagai berikut.

$$\text{Recall} = \frac{\#(\text{dokumen relevan yang dikembalikan})}{\#(\text{dokumen yang relevan})} \quad (2.5)$$

Kedua pengukuran tersebut akan memiliki jangkauan nilai mulai 0 hingga 1*(floating point)* atau sering pula dirupakan dalam bentuk persentase dengan jangkauan nilai mulai 0 hingga 100.

BAB 3
METODE PENELITIAN

Penelitian ini menggunakan kerangka proses sebagaimana yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Desain Sistem Rekomendasi dengan Normalisasi Lingua::IdSplitter

Sebagaimana tampak pada Gambar 3.1, hasil rekomendasi didapatkan dari dua tahapyaitu tahap pengindeksan yang hanya dijalankan sebanyak satu kalidan tahap perekomendasian data diskusi. Tahap pengindeksan harus dilakukan terlebih dahulu agar sistem rekomendasi dapat dijalankan.

3.1 Data Masukan

Masukan pada penelitian ini terbagi menjadi dua jenis, yakni masukan untuk tahap pengindeksan dan masukan untuk tahap perekomendasian. Masukan untuk tahap pengindeksan berupateks diskusi (pertanyaan dan jawaban) yang diambil dari Stack Overflow.Data diambil dengan bantuan Stack Exchange Data Explorer berdasarkan *query* SQL berikut.

```
SELECT * FROM Posts WHERE Id IN (PostIds) ORDER BY Id
```

Gambar 3.2*Query* untuk Mengambil Pertanyaan dari Stack Overflow

PostIds yang menjadi nilaioperator*IN* pada *query*Gambar 3.2 didapatkan dari kumpulan *id post* yang digunakan pada penelitian Arwan dkk. (2015).Kumpulan *id post* tersebut merupakan datapertanyaan dengan label *(tag)* Java yang berjumlah 153 data.Data tersebut terdiri atas 34 data tentang topik *sorting*, 34 data tentang topik *database*, 32 data tentang topik *text file*, 34 data tentang topik *graphics*, dan 19 data tentang topik *thread*. Selain data pertanyaan, jawaban atas pertanyaan tersebut juga diambil. *Query* yang digunakan adalah sebagai berikut.

```
SELECT * FROM Posts WHERE ParentId IN (PostIds) ORDER BY Id
```

Gambar 3.3*Query* untuk Mengambil Jawaban dari Stack Overflow

*Query*padaGambar 3.3mirip dengan *query* pada Gambar 3.2.Yang membedakan keduanya hanya padakolomyang digunakan untuk memfilter data. Kolom yang digunakan yakni *ParentId*, sementara untuk *query* data pertanyaan menggunakan kolom*Id*. Data yang telah dihasilkan dari *query* tersebut kemudian diunduh dalam bentuk *file* CSV untuk kemudian diimpor secara lokal ke basis data MySQL.

Data yang digunakan hanya data pada kolom *Body* yang berisi pertanyaan atau jawaban dari pengguna. Ilustrasinya dapat dilihat pada Gambar 3.4.

***Body* (Pertanyaan), ID *Post*: 8549891**

```
How do I turn off text antialiasing in this Java function?

<p>Basically I would like to turn off antialias in the
following:</p>

<pre><code>public BufferedImage createText(String text) {

    //create image
    BufferedImage image = new BufferedImage(95, 20,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D graphics = (Graphics2D) image.getGraphics();

    //set background
    graphics.setColor(Color.white);
    graphics.fillRect(0, 0, 95, 20);

    //draw text
    graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_OFF);

    graphics.setColor(Color.black);
    graphics.setFont(new Font("volter", Font.PLAIN, 9));
    graphics.drawString(text, 0, 10);

    return image;
}
</code></pre>

<p>but it's not working, here is something this function
generates:</p>

<p><img src="http://i.stack.imgur.com/yWxw0.png" alt="enter
image description here"></p>

<p>I just want black and white to be used, nothing else so it's
important I get antialias disabled!</p>
```

***Body* (Jawaban), ID *Post*: 8549970, ParentID: 8549891**

```
<p>Try this instead:</p>

<pre><code>graphics.setRenderingHint(RenderingHints.KEY_TEXT_ANT
IALIASING,

RenderingHints.VALUE_TEXT_ANTIALIAS_OFF);
</code></pre>
```

Gambar 3.4 Data Masukan Ketika Dibaca Oleh Sistem

Ilustrasi yang ditunjukkan pada Gambar 3.4merupakan data masukan pada tahap pengindeksan. Kedua jenis informasi tersebut (pertanyaan dan jawaban-jawaban) akan digabung menjadi satu untuk mewakili satu data diskusi. Dapat dilihat bahwa informasi yang akan diindeks terdiri atas tag-tag HTML, hal tersebut

mengingat data diskusi di Stack Overflow berupa dokumen HTML. Gambar tersebut merupakan representasi fisik web Stack Overflow yang telah dijelaskan pada subbab2.6. Dapat dilihat pula bahwa data tidak hanya berupa kata-kata (bahasa natural) dari pengguna, namun juga berupa potongan kode program yang biasanya diapit dalam tag `<code>`.

Adapun masukan untuk tahap perekomendasian merupakan potongan kode program yang dimasukkan oleh pengguna. Ilustrasinya dapat dilihat pada Gambar 3.3.

```
...
public void paint(Graphics g) {

    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

    g.drawString("This is my string", 200, 200);
}
...
```

Gambar 3.5*Query* Sistem Berupa Kode Program Java

Potongan kode program yang dimaksud pada Gambar 3.5 merupakan potongan kode program dalam bahasa Java `(*.java)` yang dijadikan masukan pada tahap perekomendasian.

3.2 Filter Simbol

Data masukan yang mengandung simbol-simbol non-alfabet seperti titik koma (;), tanda tanya (?), dan lain-lain akan dihapus sehingga yang tersisa hanya karakter alfabet.Sedangkan tanda titik (.) akan diubah menjadi spasi karena tanda titik digunakan oleh Java untuk memisahkan objek atau *class*dengan membernya. Selain itu, karena data diskusi berupa data HTML, maka *tag-tag* HTML pada diskusi juga akan dihilangkan.Selain mengandung *tag* HTML, dokumen web terkadang juga mengandung simbol yang dirupakan dalam bentuk entitas HTML seperti `&lt;` yang ketika ditampilkan di peramban web akan muncul sebagai simbol *kurang dari* (<). Entitas HTML tersebut juga akan dihapus sebelum data diproses lebih lanjut. Gambar yang disertakan pada diskusi juga akan dihapus mengingat tag `<img>` pada data hanya memuat lokasi gambar saja pada server. Ilustrasi data yang telah difilter dapat dilihat pada Gambar 3.4.

```
How do I turn off text antialiasing in this Java function

Basically I would like to turn off antialias in the following

public BufferedImage createTextString text

    create image
    BufferedImage image new BufferedImage
        BufferedImageTYPE_INT_ARGB
    Graphics2D graphics Graphics2D imagegetGraphics

    set background
    graphicssetColorColorwhite
    graphicsfillRect

    draw text
    graphicssetRenderingHintRenderingHintsKEY_ANTIALIASING
        RenderingHintsVALUE_ANTIALIAS_OFF

    graphicssetColorColorblack
    graphicssetFontnew FontvolterFontPLAIN
    graphicsdrawStringtext

    return image

but its not working here is something this function generates

I just want black and white to be usednothing else so its
important I get antialias disabled
```

```
Try this instead

GraphicssetRenderingHintRenderingHints KEY_TEXT_ANTIALIASING
                          RenderingHintsVALUE_TEXT_ANTIALIAS_OFF
```

Gambar 3.6 Hasil Filter Simbol

Dapat dilihat pada Gambar 3.6, yang akan tersisa pada data hanya karakter-karakter alfabet saja. Semua tag HTML sudah tidak ada. Dengan kata lain, dokumen HTML akan diubah bentuknya menjadi teks *plain*.

3.3 Tokenisasi

Datayang telahdifilter kemudian ditokenisasi sehingga akan didapatkan daftar term.Model tokenisasi yang digunakan adalah *word token*, yakni pemisahan term berdasarkan karakter spasi *(white space)* sebagaimana tampak padaGambar 3.7.

```
...              create          Graphics2D      setColor
```

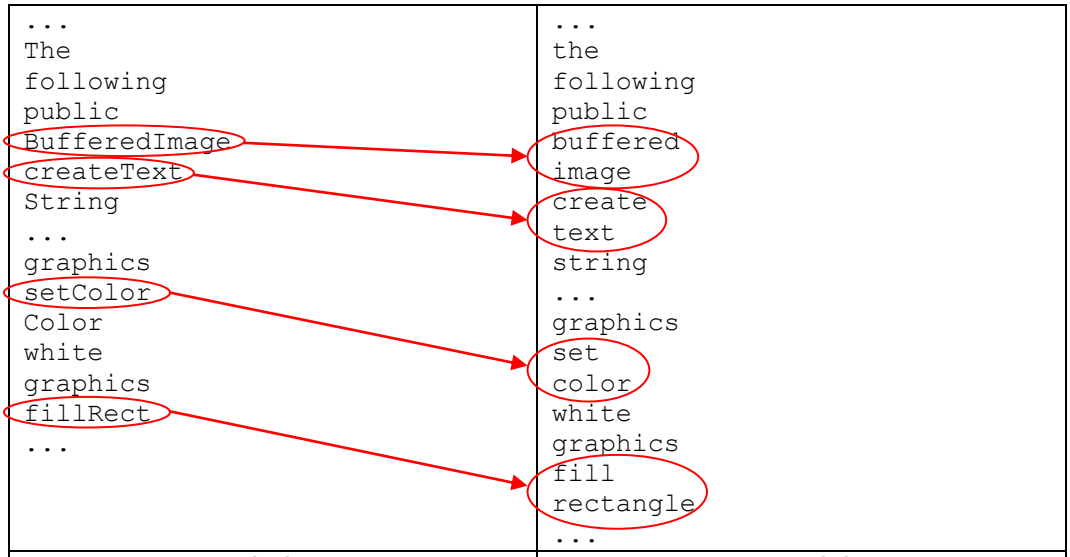
The	image	graphics	Color
following	BufferedImage	Graphics2D	white
public	image	image	graphics
BufferedImage	new	getGraphics	fillRect
createText	BufferedImage	set	...
String	BufferedImage	background	
Text	TYPE_INT_ARGB	graphics	

Gambar 3.7 Hasil Tokenisasi

Dapat dilihat padaGambar 3.7 tersebut bahwa setelah proses tokenisasi dilakukan, akan didapatkan daftar term yang dapat digunakan untuk proses berikutnya.

3.4 Normalisasi Term dengan Lingua::IdSplitter dan *Case Folding*

Term-termhasil tokenisasi akan dinormalisasi dengan Lingua::IdSplitter. Normalisasi ini akan mengubah *identifier* yang umumnya mengandung gabungan kata serta mengekspansi singkatan menjadi kata lengkap.Term yang memiliki variasi huruf besar dan huruf kecil selanjutnya akan diubah semuanya ke huruf kecil. Hal ini dimaksudkan untuk menyamaratakan semua karakter ke bentuk huruf kecil.Ilustrasinya dapat dilihat pada Gambar 3.8.



Gambar 3.8 Hasil Sebelum dan Sesudah Pemrosesan dengan Lingua::IdSplitter

Dapat dilihat pada Gambar 3.8bahwa term `BufferedImage`akan dapat berubah menjadi `buffered` dan `image`,`createText` dapat berubah menjadi `create`dan`text`, `setColor` dapat berubah menjadi `set` dan `color`, serta term `fillRect`juga dapat berubah menjadi `fill` dan `rectangle`.

Mengingat Lingua::IdSplitter merupakan modul Perl, maka pada penelitian ini, Lingua::IdSplitter akan difungsikan sebagai program eksternal yang akan dieksekusi pada aplikasi berbasis Java yang dikembangkan dengan bantuan objek`Runtime`dan `Process`. Adapun format perintahnya dapat dilihat padaGambar 3.9.

```
perl id-splitter termTarget
```

Gambar 3.9 Format Eksekusi Lingua::IdSplitter

Sebagaimana tampak pada Gambar 3.9,term yang akan dinormalisasi (`termTarget`) menjadi parameter Lingua::IdSplitter(`id-splitter`) yang pada akhirnya akanmenghasilkan ≥ 1 term hasil normalisasi.

3.5 Penghapusan *Stop Word*

Daftar term yang telah diperoleh selanjutnya difilter kembali untuk menghilangkan term-term yang merupakan term umum *(stop word)* seperti term *the*, *a*, *or*, dan lain-lain sebagaimana tampak pada Gambar 3.10.

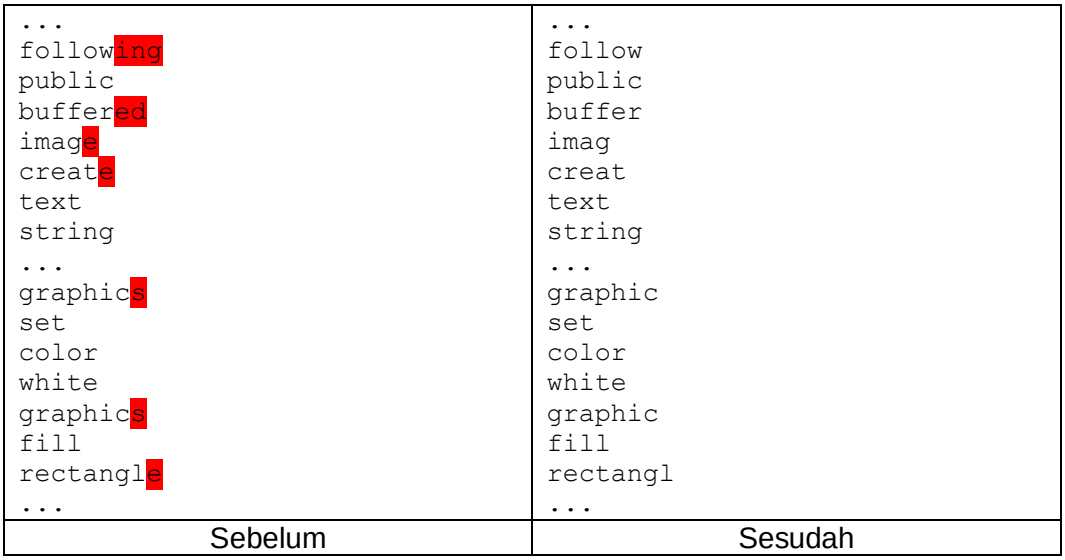
... the following public buffered image create text string ... graphics set color graphics fill rectangle ...	... following public buffered image create text string ... graphics set color graphics fill rectangle ...
Sebelum	Sesudah

Gambar 3.10 Hasil Sebelum dan Sesudah Penghapusan *Stop Word*

Sebagaimana tampak pada Gambar 3.10, term *the* akan dihapus dari daftar term karena termasuk salah satu term dalam *stop list* sebagaimana tercantum dalam Lampiran 2.

3.6 Stemming

Daftar term yang telah difilter dengan *stop word* selanjutnya akan diubah lagi bentuknya dengan proses *stemming* untuk mengurangi variasi kata yang memiliki bentuk dasar (stem).Ilustrasi proses *stemming* dapat dilihat pada Gambar 3.11.



Gambar 3.11 Hasil Sebelum dan Sesudah *Stemming*

Karakter yang disorotdengan warna merah pada Gambar 3.11 adalah karakter yang akan dihapus ketika dilakukan*stemming*.

3.7 Pembobotan TF-IDF

Term yang telah didapatkan selanjutnya dihitung frekuensi kemunculannya serta diboboti dengan menggunakan TF-IDF berdasarkanrumus (2.1). Dengan demikian, akan didapatkan daftar term unik beserta bobotnya terhadap data diskusi. Hasil pembobotan ini akan menghasilkan fitur vektor untuk tiap-tiap data diskusi. Adapun ilustrasinya dapat dilihat pada Gambar 3.12.D_1, D_2, D_3yang ditunjukkan pada Gambar 3.12merepresentasikan data diskusi dengan ID 1, 2, 3. Term *color*, *white*, dan seterusnya merupakan term unik yang berasal dari hasil *stemming*. Sedangkan nilai yang berada pada tabel merupakan pemetaan bobot term terhadap data diskusi.

	D_1	D_2	D_3	...
...	...	...	...	...

color	0,0006	0,0001	0,0003	...
white	0	0	0	...
graphic	1,0344	1,0344	13,3255	...
fill	0	0	0	...
rectangl	0,0003	0	0	...
...	...	...	...	...

Gambar 3.12 Contoh Daftar Term Beserta Bobotnya dalam Bentuk Matriks Term-Dokumen

3.8 Cosine *Similarity*

Penghitungan Cosine*similarity* digunakan untuk membandingkan antara *query* yang dimasukkan dengan data diskusi yang sudah tersedia dalam bentuk indeks terbobot (vektor). Selanjutnya hasil pembandingan akan diperingkat berdasarkan tingkat kemiripannyamenggunakanrumus (2.3).

3.9 Evaluasi Hasil

Evaluasi pada penelitian ini dilakukan dengan tujuan untuk mengetahui kualitas perbaikan metode rekomendasi yang memanfaatkan proses normalisasi *identifier* menggunakan Lingua::IdSplitter melaluipenghitungannilai *precision* dan*recall*berdasarkan rumus(2.4) dan(2.5).Nilai-nilai tersebut akan dibandingkan dengannilai*precision* maupun*recall* yang dihasilkan oleh sistem rekomendasi yang tidak memanfaatkan proses normalisasi *identifier*.Mengingat data yang digunakan berasal dari penelitian Arwan dkk. (2015), maka hasil penelitian ini juga akan dilihat keselarasannya dengan penelitian yang telah dilakukan oleh Arwan dkk. tersebut.Selain itu, guna mengetahui kemampuan Lingua::IdSplitter dalam menormalisasi *identifier*, akan digunakan pula data diskusi alternatifsejumlah 80 buah (20 diskusi/topik) dengan *identifier* singkatan yang lebih mendominasi dibandingkan dengan singkatan pada data diskusi yang digunakan pada penelitian Arwan dkk. (2015). Dominasi *identifier* singkatan tersebut muncul terutama sebagai*identifier-identifier* yang merupakan *identifier* atau term kunci topik.

Data uji *(query)*masukan pada sistem ini adalah kode program Java yang berjumlah 32 buah.Kode program tersebut diambil secara acak dari *repository* GitHub dengan spesifikasi sebagaimana tampak padaTabel 3.1.

Tabel 3.1 Spesifikasi Data Uji

Topik Kode Program	**Hasil Diskusi yang Diharapkan**	**Jml. *Query***
Thread synchronization	Diskusi yang membahas tentang sinkronisasi *thread*	8
*Accessing*MySQL	Diskusi yang membahas tentang pengaksesan data pada MySQL	8
Bubble sort	Diskusi yang membahas tentang metode*bubble sort*	8
Draw rectangle	Diskusi yang membahas tentang cara menggambar persegi panjang	8
	Total*Query*	**32**

Setiap kode program yang dimasukkan,diharapkan mampu menghasilkan rekomendasi diskusi sesuai topik dari kode program sebagaimana yang telah disajikan padaTabel 3.1.Detail kode-kode program yang dimaksud dapat dilihat pada Lampiran 3. Sedangkan untuk hasil rekomendasi yang akan dihitung performanya adalah 15 data dengan nilai bobot dokumen tertinggi pertama sebagaimana yang dilakukan oleh Ponzanelli (2013).

BAB 4
HASIL DAN PEMBAHASAN

4.1 Proses Rekomendasi

Proses rekomendasi dilakukan dengan cara melakukan *query* terhadap sistem menggunakan kode program yang telah didapatkan. *Query* yang dimaksud adalah kumpulan kata kunci yang diperoleh dari kode program yang telah diproses sebelumnya sesuai metode yang digunakan.Gambar 4.1merupakan contoh kode program terkait topik *graphics*yang digunakan saat proses rekomendasi.

```
/*
* File: DrawRectangle.java
* ------------------------
* This program allows users to create rectangles on the canvas
* by clicking and dragging with the mouse.
*/

import java.awt.event.*;
import acm.graphics.*;
import acm.program.*;

/** This class allows users to drag rectangles on the canvas */
public class DrawRectangle extends GraphicsProgram {

      /** Runs the program */
      public void run() {
            addMouseListeners();
      }

/** Called on mouse press to record the starting coordinates */
      public void mousePressed(MouseEvent e) {
            startX = e.getX();
            startY = e.getY();
            lastX = e.getX();
            lastY = e.getY();
            gobj = getElementAt(startX, startY);
            if (gobj == null) {
                  currentRect = new GRect(startX, startY, 0, 0);
                  currentRect.setFilled(true);
                  add(currentRect);
            }
      }

/** Called on mouse drag to reshape the current rectangle */
      public void mouseDragged(MouseEvent e) {
            if (gobj != null) {
                  gobj.move(e.getX() - lastX, e.getY() - lastY);
                  lastX = e.getX();
```

```
                    lastY = e.getY();
                } else {
                double x = Math.min(e.getX(), startX);
                double y = Math.min(e.getY(), startY);
                double width = Math.abs(e.getX() - startX);
                double height = Math.abs(e.getY() - startY);
                currentRect.setBounds(x, y, width, height);
                }
        }

/* Private state */
        private GRect currentRect; /* The current rectangle */
        private double startX; /* The initial mouse X position */
        private double startY; /* The initial mouse Y position */
        private GObject gobj;
        private double lastX;
        private double lastY;
}
```

Gambar 4.1 Contoh Kode Program Sebelum Diekstrak Kata Kuncinya

Contoh kode program yang tampak pada Gambar 4.1merupakan kode program Java yang mengandung berbagai unsur seperti *class*, objek, variabel, *method*, komentar, dan lainnya.Kode program tersebut kemudian diekstrak untuk mendapatkan kata-kata kunci dengan menggunakan metode yang sama dengan metode yang digunakan ketika proses pengindeksan. Hasil ekstraksi kata kunci, baik yang memanfaatkan normalisasi *identifier* dan yang tidak, dapat dilihat pada Gambar 4.2dan Gambar 4.3.

```
file draw rectangl java program user creat rectangl canva click
drag mous import java awt event import acm graphic import acm
program class user drag rectangl canva public class draw
rectangl extend graphic program run program public void run add
mous listen call mous press record start coordin public void
mous press mous event start start object element start start
object null current rectangl rectangl start start current
rectangl set fill true add current rectangl call mous drag
reshap current rectangl public void mous drag mous event object
null object move doubl math min start doubl math min start doubl
width math ab start doubl height math ab start current rectangl
set bound width height privat state privat rectangl current
rectangl current rectangl privat doubl start initi mous posit
privat doubl start initi mous posit privat object object privat
doubl privat doubl
```

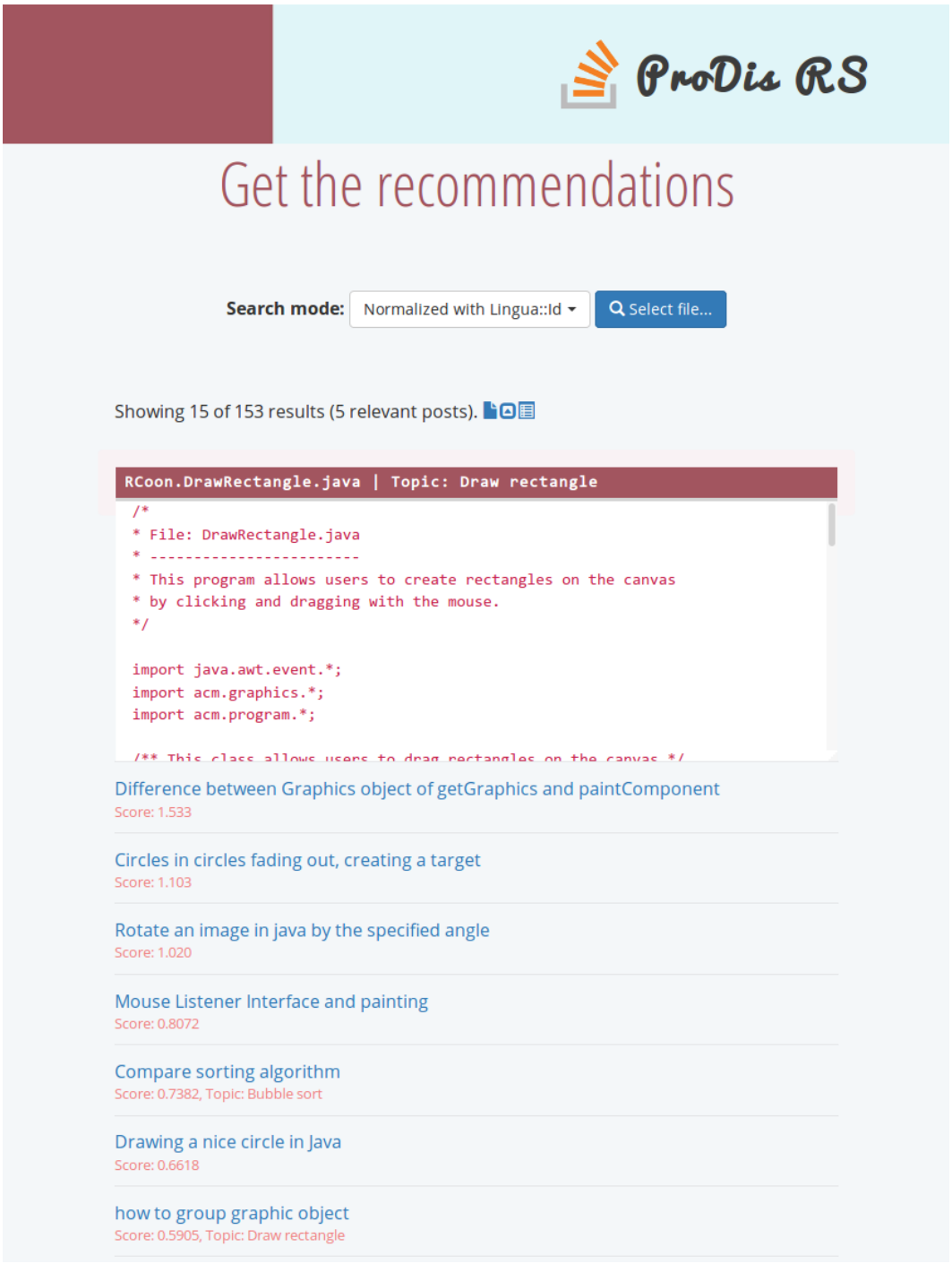
Gambar 4.2 Kata Kunci Hasil Ekstraksi dengan Memanfaatkan Normalisasi *Identifier*

```
file drawrectangl java program user creat rectangl canva click
drag mous import java awt event import acm graphic import acm
program class user drag rectangl canva public class drawrectangl
extend graphicsprogram run program public void run
addmouselisten call mous press record start coordin public void
mousepress mouseev startx getx starti geti lastx getx lasti geti
gobj getelementat startx starti gobj null currentrect grect
startx starti currentrect setfil true add currentrect call mous
drag reshap current rectangl public void mousedrag mouseev gobj
null gobj move getx lastx geti lasti lastx getx lasti geti doubl
math min getx startx doubl math min geti starti doubl width math
ab getx startx doubl height math ab geti starti currentrect
setbound width height privat state privat grect currentrect
current rectangl privat doubl startx initi mous posit privat
doubl starti initi mous posit privat gobject gobj privat doubl
lastx privat doubl lasti
```

Gambar 4.3 Kata Kunci Hasil Ekstraksi Tanpa Normalisasi *Identifier*

Dapat dilihat pada Gambar 4.2, kata kunci yang dihasilkan melalui proses normalisasi *identifier* yakni sebanyak 142 term (48 term unik). Sedangkan yang tampak pada Gambar 4.3, kata kunci hasil ekstraksi tanpa normalisasi sebanyak 146 term (60 term unik). Dapat dilihat pula, kumpulan kata kunci yang dihasilkan melalui proses normalisasi cenderung lebih mirip bahasa alami dibandingkan dengan kumpulan kata kunci tanpa normalisasi. Hal tersebutdisebabkan karena*identifier* yang umumnya terdiri atas gabungan kata sudah terpisah menjadi kata-kata tunggal. Selain itu, term singkatan juga dapat diubah ke bentuk term lengkapnya. Kata kunci yang telah didapat selanjutnya akan digunakan sebagai *query* untuk mendapatkan data-data diskusi yang relevan dengan kode program tersebut.

Adapun proses rekomendasinya dapat dilihat padaGambar 4.4.Terlihat pada gambar bahwa proses rekomendasi dapat dilakukan dengan atau tanpa normalisasi dengan memilih mode pada kolom pilihan*Search mode* di aplikasi yang dikembangkan. Kode program yang akan dijadikan sebagai *query* dapat dimasukkan dengan menekan tombol *Select file*. Sistem akan memberikan rekomendasi maksimum sebanyak 15 diskusi. Antara kode program dan hasil rekomendasi dapat diketahui kecocokan topiknya berdasarkan informasi yang ada pada bagian kode program dan keterangan pada tiap-tiap judul diskusi. Selain itu, dapat diketahui pula skor similaritas tiap-tiap diskusi dengan kode program yang dimasukkan.



Gambar 4.4 Proses Rekomendasi

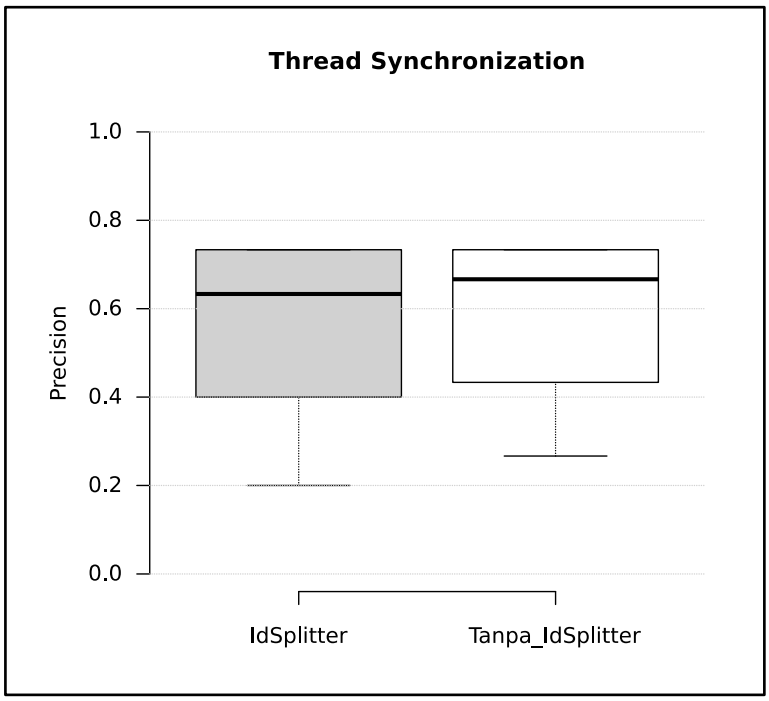
Berdasarkan contoh hasil rekomendasi pada Gambar 4.4, diketahui bahwa diskusi dengan skor tertinggi ternyata tidak terkait dengan topik kode program*(draw rectangle)*. Diskusi yang terkait dengan topik tersebut berada pada urutan ke-7 dengan skor 0,5905. Tampak pada gambar pula bahwa diskusi relevan yang direkomendasikan adalah sejumlah 5 diskusi, sementara 10 lainnya tidak relevan.

4.2 Hasil Pengujian

Sebagaimana telah disebutkan pada subbab 3.9, pengujian akan dilakukan dengan melakukan *query* terhadap sistem menggunakan kode sumber program Java dengan topik yang telah ditentukan. Detail hasil pengujian dapat dilihat pada Lampiran 4 dan Lampiran 5.

4.2.1 Pengujian dengan Kode ProgramTerkait *Thread Synchronization*

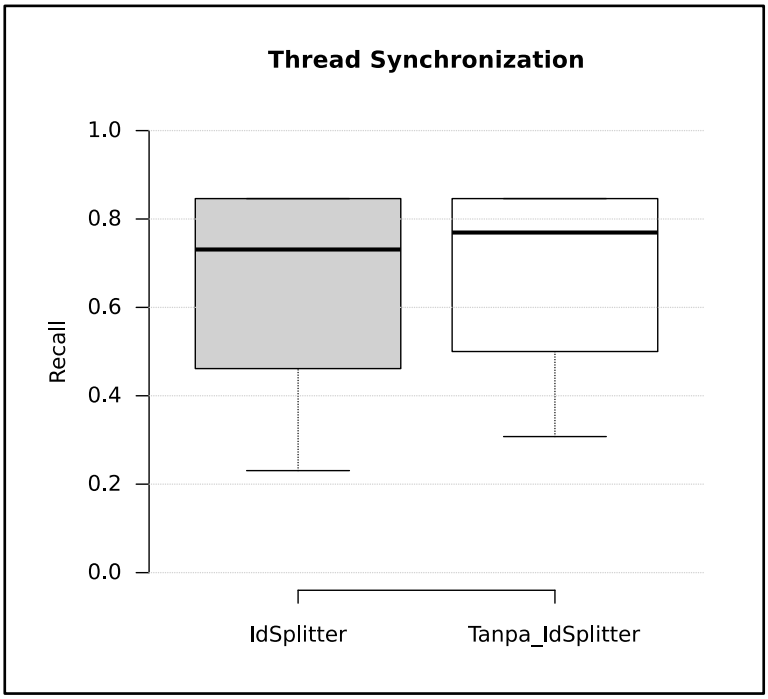
Pengujian terhadap 8 kode program yang terkait dengan topik*thread synchronization* dapat dilihat padaGambar 4.5 dan Gambar 4.6.



Gambar 4.5Perbandingan*Precision* untuk Topik *Thread Synchronization*

Berdasarkan grafik pada Gambar 4.5, dapat dilihat bahwa nilai *precision* untuk data yang dinormalisasi dengan Lingua::IdSplitter memiliki jangkauan antara 0,2 hingga 0,73 dengan median 0,63 sementara data yang tidak dinormalisasi memiliki jangkauan antara 0,27 hingga 0,73 dengan median 0,67. Sedangkan apabila dilihat dari nilai *recall*,sebagaimana tampak pada Gambar 4.6, diketahui bahwa *recall* yang diperoleh ketika data dinormalisasi yakni antara 0,23 hingga 0,85 dengan median 0,73 sementara data yang tidak dinormalisasi yakni antara 0,31 hingga 0,85 dengan median 0,77. Dengan demikian, dapat disimpulkan

bahwa proses normalisasi tidak dapat meningkatkan nilai*precision* maupun *recall* untuk *query* dengan topik *thread synchronization*.

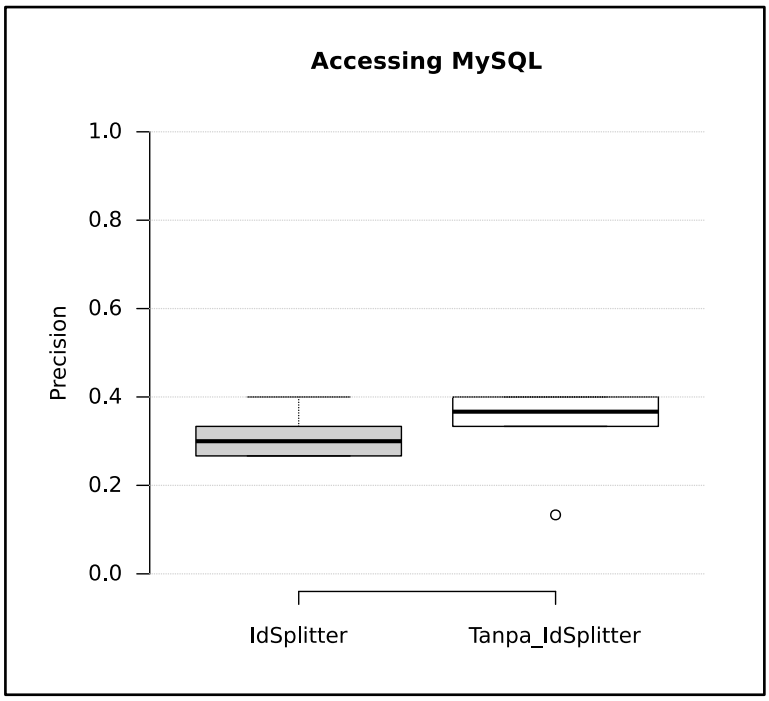


Gambar 4.6 Perbandingan *Recall* untuk Topik *Thread Synchronization*

Hal ini disebabkan oleh tidak ditemukannya*identifier* `synchronize` yang disingkat menjadi `sync`pada data diskusi sebagaimana yang lumrah digunakan oleh *programmer* dalam menyingkat *identifier*. Oleh karena itu, normalisasi (ekspansi) term tidak berdampak terhadap kasus *query* dengan topik *thread synchronization* ini. Selain itu, sinkronisasi *thread* juga umum menggunakan *exception* dengan jenis `InterruptedException`. Pemisahan *identifier* tersebut menjadi `Interrupted`dan `Exception`justru menurunkan skor dokumen terkait *thread synchronization* karena term `exception` lumrah ditemukan pada topik-topik selainnya.Sementara itu, ekspansi term dalam hal ini tidak terlalu berpengaruh mengingat pada kode program dan data sampel diskusi yang digunakakan tidak ada yang menggunakan term `sync` yang biasanya digunakan untuk mewakili term `synchronize` yang merupakan term penting pada topik *thread synchronization*.

4.2.2 Pengujian dengan Kode ProgramTerkait *Accessing*MySQL

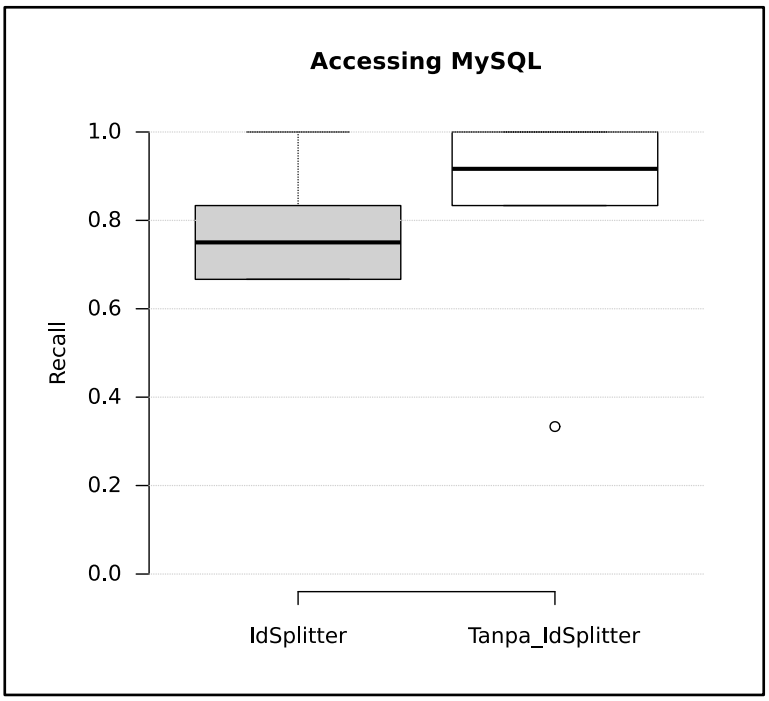
Pengujian terhadap 8 kode program yang terkait dengan topik *accessing* MySQLdapat dilihat padaGambar 4.7 dan Gambar 4.8.



Gambar 4.7Perbandingan*Precision* untuk Topik *Accessing* MySQL

Berdasarkan grafik pada Gambar 4.7, dapat dilihat bahwa nilai *precision* untuk data yang dinormalisasi dengan Lingua::IdSplitter memiliki jangkauan antara 0,27 hingga 0,4 dengan median 0,3 sementara data yang tidak dinormalisasi memiliki jangkauan antara 0,33 hingga 0,4 dengan median 0,37. Sementara apabila dilihat dari nilai *recall*,sebagaimana tampak pada Gambar 4.8, diketahui bahwa *recall* yang diperoleh ketika data dinormalisasi yakni antara 0,23 hingga 1 dengan median 0,75 sementara data yang tidak dinormalisasi yakni antara 0,83 hingga 1 dengan median 0,92. Dengan demikian, dapat disimpulkan bahwa proses normalisasi juga tidak dapat meningkatkan nilai *precision* maupun *recall* untuk *query* dengan topik *accessing* MySQL. Hal ini antara lain disebabkan oleh hasil ekspansi *identifier* `jdbc` menjadi `j`, `database`, dan `c`. Perluasan tersebut justru membuat direkomendasikannya data-data diskusi lain yang mengandung term `database`meski tidak memiliki term `jdbc` yang menjadi ciri umum bahasan tentang pengaksesan MySQL. Selain itu, pemisahan *identifier* `mysql`menjadi `my`

dan `sql` juga turut menurunkan performa rekomendasi. Pemisahan tersebut justru mengakibatkan hilangnya term penting `mysql` dalam mewakili data diskusi terkait *accessing* MySQL.

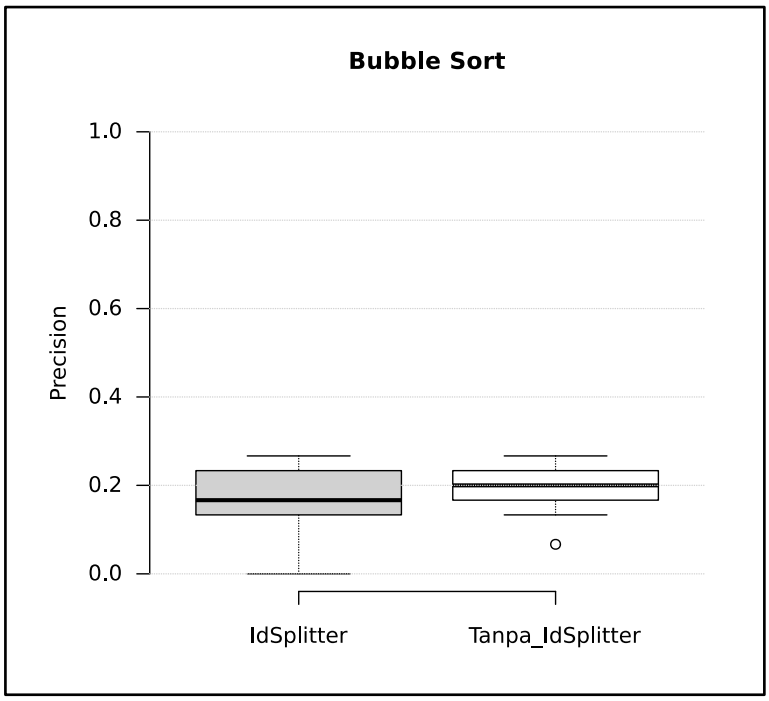


Gambar 4.8 Perbandingan *Recall* untuk Topik *Accessing* MySQL

Normalisasi (pemisahan) dapat memberikan peningkatan hasil rekomendasi untuk *query* kode program `org.embulk.output.MySQLOutputPlugin.java`yang berisi tentang kode program untuk melakukan *insert* ke basis data MySQL secara massal. Tanpa normalisasi, sistem menjadikan *identifier* seperti`batchInsert`, `mysqlBatchInsert`, dan `newBatchInsert` sebagai term penting *query*. Secara logika, term-term tersebut memang tepat mewakili *query*. Namun, hal tersebut tidak didukung oleh sampel data diskusi, tidak ada satupun topik data diskusi yang membahas tentang proses *insert* secara massal.Jika demikian, setidaknyasistem bisa merekomendasikan data diskusi tentang proses *insert* pada MySQL secara umum. Akan tetapi, kenyataannya tidak.Efek tersebut dapat dilihat pada pencilan nilai *precision* pada titik 0,13 dan *recall* pada titik 0,33 untuk proses rekomendasi yang tidak menggunakan normalisasi Lingua::IdSplitter.

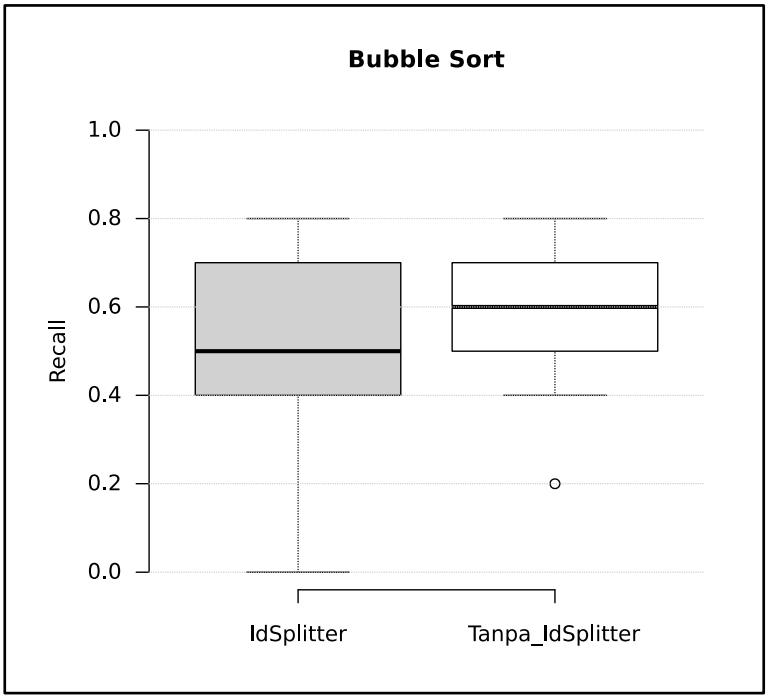
4.2.3 Pengujian dengan Kode Program Terkait *Bubble Sort*

Pengujian terhadap 8 kode program yang terkait dengan topik *bubble sort* dapat dilihat pada Gambar 4.9 dan Gambar 4.10.



Gambar 4.9Perbandingan*Precision* untuk Topik *Bubble Sort*

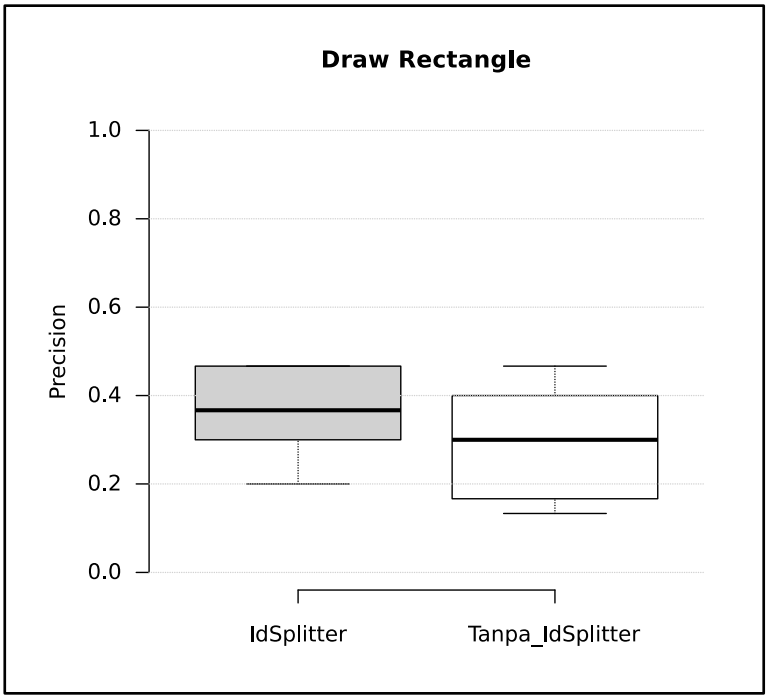
Grafik pada Gambar 4.9 menunjukkan bahwa nilai *precision* untuk data yang dinormalisasi dengan Lingua::IdSplitter memiliki jangkauan antara 0 hingga 0,27 dengan median 0,17 sementara data yang tidak dinormalisasi memiliki jangkauan antara 0,13 hingga 0,27 dengan median 0,2. Sementara apabila dilihat dari nilai *recall*,sebagaimana tampak pada Gambar 4.10, diketahui bahwa *recall* yang diperoleh ketika data dinormalisasi yakni antara 0 hingga 0,8 dengan median 0,5 sementara data yang tidak dinormalisasi yakni antara 0,4 hingga 0,8 dengan median 0,6. Dapat disimpulkan bahwa proses normalisasi juga tidak dapat meningkatkan nilai *precision* maupun *recall* untuk *query* dengan topik *bubble sort*. Hal ini antara lain disebabkan oleh hasil pemisahan *identifier* `BubbleSort` menjadi `bubble` dan `sort`. Pemisahan tersebut justru membuat meningkatnya skor data diskusi tentang *sorting* yang lain, yang menggunakan metode selain *bubble sort*. Ekspansi pada topik ini juga tidak terlalu berpengaruh mengingat *identifier*`bubble` maupun `sort` jarang dirupakan dalam bentuk singkat.



Gambar 4.10 Perbandingan *Recall* untuk Topik *Bubble Sort*

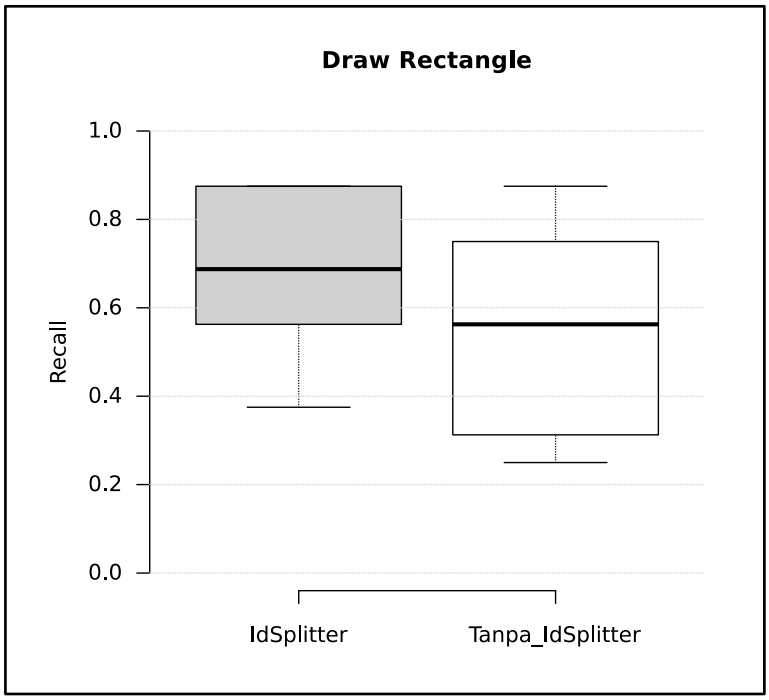
4.2.4 Pengujian dengan Kode Program Terkait *Draw Rectangle*

Pengujian terhadap 8 kode program yang terkait dengan topik *draw rectangle* dapat dilihat padaGambar 4.11 dan Gambar 4.12.



Gambar 4.11 Perbandingan *Precision* untuk Topik *Draw Rectangle*

Berdasarkan grafik padaGambar 4.11, dapat dilihat bahwa nilai *precision* untuk data yang dinormalisasi dengan Lingua::IdSplitter memiliki jangkauan antara 0,2 hingga 0,47 dengan median 0,37 sementara data yang tidak dinormalisasi memiliki jangkauan antara 0,13 hingga 0,47 dengan median 0,3.



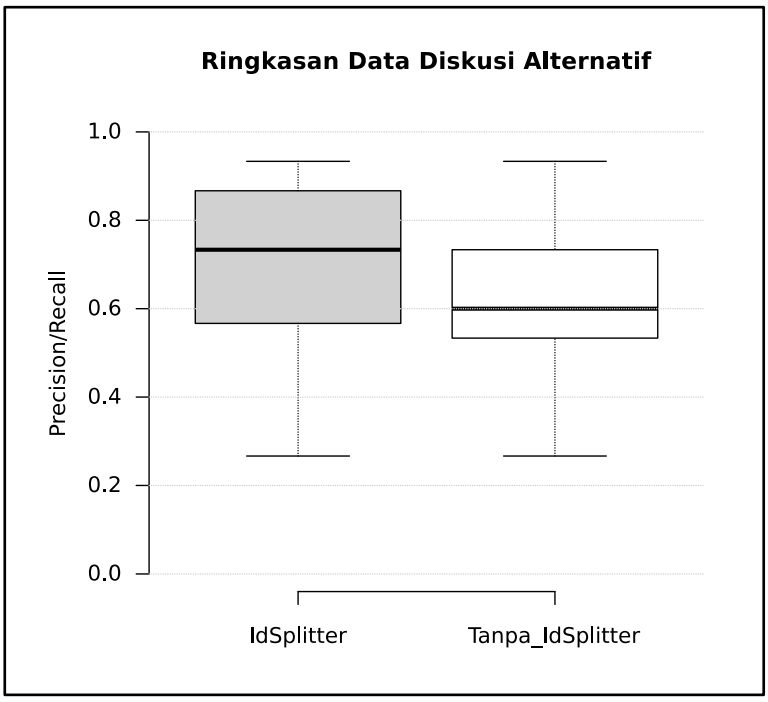
Gambar 4.12Perbandingan *Recall* untuk Topik *Draw Rectangle*

Sementara apabila dilihat dari nilai *recall*,sebagaimana tampak pada Gambar 4.12, diketahui bahwa *recall* yang diperoleh ketika data dinormalisasi yakni antara 0,38 hingga 0,88 dengan median 0,69 sementara data yang tidak dinormalisasi yakni antara 0,25 hingga 0,88 dengan median 0,56. Dengan demikian, dapat disimpulkan bahwa proses normalisasi dapat meningkatkan nilai median*precision* maupun *recall*masing-masing sebesar 0,07 dan 0,13 untuk *query* dengan topik *draw rectangle*. Hal ini antara lain disebabkan oleh pemisahan *identifier*`DrawFeature` menjadi `Draw` dan `Feature`, `DrawRectangle` menjadi `Draw` dan `Rectangle`, serta `DrawRectangleOptions` menjadi `Draw`, `Rectangle`, dan `Options`.*Identifier* dalam bentuk terpisah lebih banyak ditemui pada data diskusi. Oleh karena itu, proses normalisasi menjadi penting. Selain itu, pemisahan dan ekspansi *identifier*`fillRect` menjadi `fill` dan `Rectangle` serta `drawRect` menjadi `draw` dan `Rectangle` memberikan dampak positif terhadap hasil

rekomendasi, mengingat teks diskusi yang ada lebih banyak menggunakan kata lengkap dibandingkan singkatan.

4.3 Uji Coba Menggunakan Data Alternatif

Hasil uji coba menggunakan data alternatif dengan dominasi *identifier* yang mengandung singkatan dapat dilihat padaGambar 4.13. Pengujian ini menghasilkan nilai-nilai *precision* dan *recall* yang sama, sehingga representasinya hanya ditampilkan dalam satu grafik saja.



Gambar 4.13 Ringkasan Perbandingan Uji Coba Data Alternatif

Berdasarkan grafik pada Gambar 4.13 dapat diketahui bahwa nilai *precision* dan *recall* yang dapat dihasilkan sama, yakni berada pada jangkauan 0,27 hingga 0,93. Namun, median nilai *precision* dan *recall* hasil rekomendasi yang memanfaatkan normalisasi Lingua::IdSplitter lebih tinggi 0,13 dibandingkan dengan tanpa memanfaatkan proses normalisasi.

4.4 Ringkasan Hasil Uji Coba

Berdasarkan hasil pengujian yang telah dijelaskan untuk tiap-tiap topik pada subbab 4.2.1 sampai dengan 4.2.4, diketahui bahwa proses normalisasi dengan Lingua::IdSplitter hanya berpengaruh positif terhadap rekomendasi *query* dengan topik *draw rectangle*. Hal tersebut disebabkan oleh keberadaan *identifier*`rectangle`pada data diskusi maupun *query*, baik berupa singkatan maupun term lengkap. Sedangkan berdasarkan uji coba terhadap data alternatif yang telah dipilih, diketahui bahwa normalisasi *identifier* menggunakan Lingua::IdSplitter juga berpengaruh positif terhadap hasil rekomendasi dilihat dari nilai median yang dihasilkan. Hal tersebut juga dipengaruhi oleh keberadaan *identifier* pada data diskusiyang ditulis dalam bentuk singkatan.

Adapun apabila dilihat keselarasannya dengan penelitian Arwan dkk. (2015) yang menggunakan model topik dalam merekomendasikan data diskusi, penggunaan model bobot TF-IDF masih cukup bisa diandalkan dengan melihat nilai median*recall* secara umum yang bisa dihasilkan oleh sistem berbasis TF-IDF pada data uji yang sama, yakni sebesar 0,67.Namun demikian, perbaikan masih perlu dilakukan mengingat nilai median *precision* umum yang dihasilkan hanya mampu mencapai 0,33. Hal ini diakibatkan oleh ketidaktepatan pemilihan*identifier-identifier* yang layak untuk dinormalisasi, baik terkait kasus ketidaktepatan pemisahan komposisi term maupun pengekspansian term penuh dari term singkatan sebagaimana yang telah dijelaskan pada subbab-subbab sebelumnya.

Halaman ini sengaja dikosongkan

Lampiran 1

Kamus Kata pada Lingua::IdSplitter

A. Kamus Akronim

acl ---------- acl
aes ---------- aes
ansi ---------- ansi
api ---------- api
arf ---------- arf
arm ---------- arm
asn ---------- asn
avp ---------- avp
bfd ---------- bfd
bhcp ---------- bhcp
bpf ---------- bpf
bsd ---------- bsd
bsd ---------- bsd
bsd ---------- bsd
bzip ---------- bzip
cdb ---------- cdb
cdid ---------- cdid
cdrom ---------- cdrom
cgi ---------- cgi
cgm ---------- cgm
codec ---------- codec
conexant ---------- conexant
cpu ---------- cpu
csi ---------- csi
css ---------- css
cst ---------- cst
cvs ---------- cvs
dbsm ---------- dbsm
dejagnu ---------- dejagnu
dll ---------- dll
dma ---------- dma
dom ---------- dom
dpb ---------- dpb
dsa ---------- dsa
dsp ---------- dsp
ebp ---------- ebp
elf ---------- elf
eoa ---------- eoa
eob ---------- eob
evt ---------- evt
fdc ---------- fdc
fdt ---------- fdt
frv ---------- frv
ftp ---------- ftp
gdk ---------- gdk
gid ---------- gid
gmp ---------- gmp
gnu ---------- gnu
gnu ---------- gnu
gtk ---------- gtk
hci ---------- hci
hdr ---------- hdr
href ---------- href
html ---------- html
http ---------- http
https ---------- https
ieee ---------- ieee
ifp ---------- ifp
inode ---------- inode
insn ---------- insn
ipc ---------- ipc
irq ---------- irq
iso ---------- iso
jit ---------- jit
jpeg ---------- jpeg
js ---------- js
kld ---------- kld
mav ---------- mav
mfi ---------- mfi
mips ---------- mips
mmx ---------- mmx
mpn ---------- mpn
mutex ---------- mutex
ncp ---------- ncp
nntp ---------- nntp
nss ---------- nss
pci ---------- pci
pcm ---------- pcm
pcr ---------- pcr
pkix ---------- pkix
plt ---------- plt
png ---------- png
pop ---------- pop
psp ---------- psp
psppire ---------- psppire
pwd ---------- pwd
rgb ---------- rgb
rhp ---------- rhp
rrt ---------- rrt
rtc ---------- rtc
rtx ---------- rtx
scsi ---------- scsi
sdp ---------- sdp
sgc ---------- sgc
sha ---------- sha
smob ---------- smob
smr ---------- smr
sql ---------- sql
ssl ---------- ssl
stderr ---------- stderr
stdout ---------- stdout
stt ---------- stt
stv ---------- stv
svm ---------- svm
swf ---------- swf
tcl ---------- tcl
tcp ---------- tcp
tipc ---------- tipc
tlv ---------- tlv
tms ---------- tms
toc ---------- toc
tty ---------- tty
udp ---------- udp
uid ---------- uid
uni ---------- uni
uri ---------- uri
url ---------- url
usb ---------- usb
utf ---------- utf
vma ---------- vma
vms ---------- vms
wais ---------- wais
wsa ---------- wsa
www ---------- www
xdr ---------- xdr
xfrm ---------- xfrm
xml ---------- xml
xpt ---------- xpt
zip ---------- zip

Jumlah: 129 kata

B. Kamus Singkatan (Umum)

abbr -------- abbreviation
abbrev ----- abbreviation
abs --------- absolute
addr -------- address
adr ---------- address
algo -------- algorithm
alloc -------- allocate
alog -------- analog
alt ---------- alternative
arg ---------- argument
args -------- arguments
attr ---------- attribute
auth -------- authenticate
aux --------- auxiliary
avg --------- average
blk ---------- block
bool -------- boolean
buf ---------- buffer
buff --------- buffer
byts --------- bytes
cal ---------- calculator
ch ---------- change
char -------- character
chg --------- change
chr ---------- character
clck --------- clock
clr ---------- clear
cmd -------- command
cmp -------- compare
cntr --------- counter
col ---------- column
comm ------ communication
comp ------- compare
config ------ configuration
conn -------- connection
connect ---- connection
const ------- constant
constr ------ constraint
cont -------- continuation
cpy --------- copy
ctls ---------- controls
ctr ---------- control
ctrl ---------- control
cur ---------- current
curr --------- current
cust --------- custom
db ---------- database
dec --------- declaration
del ---------- delete
dest -------- destination
dev --------- deviation
dev --------- device
dict --------- dictionnary
diff ---------- difference
dir ---------- directory
dpcy -------- dependency
dyn --------- dynamic
edt ---------- editing
elem -------- element
elt ---------- element
enum ------- enumeration
env --------- environment
envp -------- envelope
ers ---------- erase
eval -------- evaluate
evt ---------- event
exec -------- execute
exec -------- execution
exp --------- exponent
expr -------- expression
expt -------- exponent
ext ---------- extend
fail ---------- failure
flg ---------- flag
fmt ---------- format
frag --------- fragment
frm ---------- frame
frmt --------- format
func -------- function
glbl --------- global
glob -------- global
graph ------ graphic
grp ---------- group
hdl ---------- handle
hdr ---------- header
help -------- helper
hex --------- hexadecimal
hist --------- history
id ---------- identifier
idx ---------- index
imag ------- image
img --------- image
indx -------- index
info --------- information
inform ------ information
init ---------- initialization
int ---------- integer
interf ------- interface
intern ------ internal
iter ---------- iterator
kernl ------- kernel
lang -------- lang
lbl ---------- label
len ---------- length
lib ---------- library
lim ---------- limit
ln ---------- link
lng ---------- long
lnk ---------- link
lst ---------- list
mapp ------ mappings
matr -------- matrix
max -------- maximum
mem ------- memory
met --------- method
meth ------- method
mltpy ------- multiply
mnt --------- mount
mod -------- modify
mod -------- module
nbr ---------- number
net ---------- network
nod --------- node
notif -------- notification
num -------- number
obj ---------- object
ok ---------- ok
oper -------- operator
opt --------- option
opts -------- options
params ---- parameters
paren ------ parenthesis
perm ------- permission
pg ---------- page
pic ---------- picture
pict --------- picture
pntr -------- pointer
pos --------- position
prcl --------- protocol
priv --------- private
proc -------- procedure
proc -------- process
procs ------ process
prods ------ products
ptr ---------- pointer
rand -------- random
rect --------- rectangle
ref ---------- reference
reg --------- register
rem -------- remote
rem -------- remove
req --------- request
req --------- requirement
rng --------- range
sched ------ scheduler
sect -------- sector
seg --------- segment
sel ---------- select
sem -------- semaphored
sem -------- semaphore
seq --------- sequence
sfx ---------- suffix
siz ---------- size
spec ------- specification
src ---------- source
stat --------- statistic
std ---------- standard
str ---------- string
struct ------ structure
succ ------- success
svc --------- service
symbl ------ symbol
sys --------- system
sz ---------- size
tab --------- table
targ -------- target
tbl ---------- table
term -------- terminal
tmp --------- temporary
trans ------- transparency
trc ---------- trace
unlck ------- unlock
updt -------- update
usr --------- user
val ---------- value
var --------- variable
vec --------- vector
vect -------- vector
win --------- window
wind ------- windows
wrd --------- word

Jumlah: 191 kata

C. Kamus Singkatan *(Programming)*

abs ------- absolute
addr ------ address
ansi ------ ansi
arg-------- argument
args ------ arguments
ascii ------ ascii
avg ------- average
boolean - boolean
buf-------- buffer
char ------ character
charset -- charset
chr-------- character
chr-------- character
chrs ------ characters
chrs ------ characters
clr--------- clear
cmd ------ command
cmp ------ compare
cobol----- cobol
const----- constant
cpy ------- copy
cwd------- cwd
del-------- delete
descr----- description
dev ------- device
dir--------- directory
dirs ------- directories
doc ------- document
elem ----- element
eof-------- eof
eol-------- eol
eor-------- eor
eps ------- eps
eps ------- eps
err -------- error
errs------- errors
exec ----- execute
fd--------- fd
fifo ------- fifo
fig -------- figure
figs------- figures
flg -------- flag
flgs------- flags
func------ function
funcs ---- functions
gid ------- gid
gnu ------ gnu
hdr ------- header
hdr ------- header
html------ html
html------ html
ibm------- ibm
init-------- initialize
inode ---- inode
inode ---- inode
int -------- integer
iso-------- iso
iso-------- iso
len ------- length
leng------ length
lib -------- library
linux ----- linux
login ----- login
malloc--- malloc
mem----- memory
misc ----- miscellaneous
msg------ message
msgs ---- messages
multi ----- multi
num------ number
obj ------- object
param --- parameter
params --parameters
pdf --------pdf
ptr---------pointer
rect -------rectangle
regexp ---regexp
sem ------semantic
snd -------sound
src --------source
str---------string
strs -------strings
struct-----structure
structs ---structures
sym-------symbol
syms -----symbols
tbl---------table
tex --------tex
tmp -------temporary/tmp
todo ------todo
tooltip ----tooltip
tty---------tty
tuple------tuple
txt---------text
uid --------uid
unix-------unix
url---------url
utf8 -------utf8
val --------value
vals-------values
vec -------vector
ver--------version
vers ------versions
vga -------vga
vga -------vga
vol --------volume
win--------window

Jumlah: 129 kata

Halaman ini sengaja dikosongkan

Lampiran2

Stop List

a	available	contains	five	himself
a's	away	corresponding	followed	his
able	awfully	could	following	hither
about	b	couldn't	follows	hopefully
above	be	course	for	how
according	became	currently	former	howbeit
accordingly	because	d	formerly	however
across	become	definitely	forth	i
actually	becomes	described	four	i'd
after	becoming	despite	from	i'll
afterwards	been	did	further	i'm
again	before	didn't	furthermore	i've
against	beforehand	different	g	ie
ain't	behind	do	get	if
all	being	does	gets	ignored
allow	believe	doesn't	getting	immediate
allows	below	doing	given	in
almost	beside	don't	gives	inasmuch
alone	besides	done	go	inc
along	best	down	goes	indeed
already	better	downwards	going	indicate
also	between	during	gone	indicated
although	beyond	e	got	indicates
always	both	each	gotten	inner
am	brief	edu	greetings	insofar
among	but	eg	h	instead
amongst	by	eight	had	into
an	c	either	hadn't	inward
and	c'mon	else	happens	is
another	c's	elsewhere	hardly	isn't
any	came	enough	has	it
anybody	can	entirely	hasn't	it'd
anyhow	can't	especially	have	it'll
anyone	cannot	et	haven't	it's
anything	cant	etc	having	its
anyway	cause	even	he	itself
anyways	causes	ever	he's	j
anywhere	certain	every	hello	just
apart	certainly	everybody	help	k
appear	changes	everyone	hence	keep
appreciate	clearly	everything	her	keeps
appropriate	co	everywhere	here	kept
are	com	ex	here's	know
aren't	come	exactly	hereafter	knows
around	comes	example	hereby	known
as	concerning	except	herein	l
aside	consequently	f	hereupon	last
ask	consider	far	hers	lately
asking	considering	few	herself	later
associated	contain	fifth	hi	latter
at	containing	first	him	latterly

least
less
lest
let
let's
like
liked
likely
little
look
looking
looks
ltd
m
mainly
many
may
maybe
me
mean
meanwhile
merely
might
more
moreover
most
mostly
much
must
my
myself
n
name
namely
nd
near
nearly
necessary
need
needs
neither
never
nevertheless
new
next
nine
no
nobody
non
none
noone
nor
normally
not
nothing
novel
now
nowhere
o
obviously
of
off
often
oh
ok
okay
old
on
once
one
ones
only
onto
or
other
others
otherwise
ought
our
ours
ourselves
out
outside
over
overall
own
p
particular
particularly
per
perhaps
placed
please
plus
possible
presumably
probably
provides
q
que
quite
qv
r
rather
rd
re
really
reasonably
regarding
regardless
regards
relatively
respectively
right
s
said
same
saw
say
saying
says
second
secondly
see
seeing
seem
seemed
seeming
seems
seen
self
selves
sensible
sent
serious
seriously
seven
several
shall
she
should
shouldn't
since
six
so
some
somebody
somehow
someone
something
sometime
sometimes
somewhat
somewhere
soon
sorry
specified
specify
specifying
still
sub
such
sup
sure
t
t's
take
taken
tell
tends
th
than
thank
thanks
thanx
that
that's
thats
the
their
theirs
them
themselves
then
thence
there
there's
thereafter
thereby
therefore
therein
theres
thereupon
these
they
they'd
they'll
they're
they've
think
third
this
thorough
thoroughly
those
though
three
through
throughout
thru
thus
to
together
too
took
toward
towards
tried
tries
truly
try
trying
twice
two
u
un
under
unfortunately
unless
unlikely
until
unto
up
upon
us
use
used
useful
uses
using
usually
uucp
v
value
various
very
via
viz
vs
w
want
wants
was
wasn't
way
we
we'd
we'll
we're
we've
welcome
well
went
were
weren't
what
what's
whatever
when
whence
whenever
where
where's
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
who's

whoever	wish	would	yet	your
whole	with	would	you	yours
whom	within	wouldn't	you'd	yourself
whose	without	x	you'll	yourselves
why	won't	y	you're	z
will	wonder	yes	you've	zero
willing				

Halaman ini sengaja dikosongkan

Lampiran 3

Detail Data Uji

Topik	No.	Nama *File*	URL Sumber
Thread synchro-nization	1	com.in28minutes.java.threads.ThreadDeadlock.java	https://goo.gl/kUv3PT
	2	PingPong.java	https://goo.gl/pzPrhJ
	3	IncorrectSynchronizedIncrement.java	https://goo.gl/jL4tct
	4	sleep.practice.com.sleepimpl.Threa.↵ sleep%20Implementation.java	https://goo.gl/kpBjcc
	5	classes.test.Synchronized.java	https://goo.gl/lVYHxx
	6	BlockingQueue.java	https://goo.gl/7b3UYj
	7	java.util.concurrent.Semaphore.TwistedPairVer2.java	https://goo.gl/s8RjQY
	8	Collision.java	https://goo.gl/lQ2nvM
Accessing MySQL	9	ucf.mysql.jdbc.DBEngine.java	https://goo.gl/Bs2ZD5
	10	io.github.web.data.mysql.MySqlJdbcInsert.java	https://goo.gl/LO1dUX
	11	org.embulk.output.MySQLOutputPlugin.java	https://goo.gl/ei0IuI
	12	JdbcMysql.java	https://goo.gl/iQKBEj
	13	com.github.kristofa.brave.mysql.↵ MySQLStatementInterceptor.java	https://goo.gl/c2UkH3
	14	org.ops4j.pax.jdbc.mysql.impl.↵ MysqlDataSourceFactory.java	https://goo.gl/gVcejs
	15	com.gmail.johanringmann.blockcounter.MySQL.java	https://goo.gl/8VPqdV
	16	com.mysql.jdbc.integration.c3p0.↵ MysqlConnectionTester.java	https://goo.gl/9h5LlX
Bubble sort	17	com.github.pedrovgs.problem74.BubbleSort.java	https://goo.gl/oJYuE4
	18	LinkedList.java	https://goo.gl/5azHkx
	19	bubbleSort.java	https://goo.gl/WoZHyv
	20	spagy.BubbleSort.java	https://goo.gl/zzbJQ2
	21	sorting.algorithms.BubbleSort.java	https://goo.gl/W682RB
	22	quincy.BubbleSort.java	https://goo.gl/tYVM7d
	23	jeffheaton.BubbleSort.java	https://goo.gl/D7Euc4
	24	JakenHerman.Bubblesort.java	https://goo.gl/j5i1ow
Draw rectangle	25	com.googlecode.lanterna.examples.DrawRectangle.java	https://goo.gl/C473VJ
	26	org.peimari.gleaflet.client.draw.DrawRectangle.java	https://goo.gl/s9lUce
	27	q2.DrawRectangle.java	https://goo.gl/whiqC4
	28	com.mlakhia.draw.shapes.Rectangle.java	https://goo.gl/7uYWv6
	29	RCoon.DrawRectangle.java	https://goo.gl/cnkj5f
	30	shapes.rectangle.java	https://goo.gl/M3B6wa
	31	mrquatsch.Rectangle.java	https://goo.gl/xUqdJ6
	32	dcstimm.Rectangle.java	https://goo.gl/Q6dj69

Halaman ini sengaja dikosongkan

Lampiran 4

Hasil Uji Coba Menggunakan Normalisasi Lingua::IdSplitter

Topik	***#Query***	**#Relevan**	***Precision***	***Recall***
Thread Synchronization	1	11	0.7334	0.8462
	2	11	0.7334	0.8462
	3	11	0.7334	0.8462
	4	6	0.4	0.4616
	5	11	0.7334	0.8462
	6	6	0.4	0.4616
	7	3	0.2	0.2308
	8	10	0.6667	0.7693
Accessing MySQL	9	5	0.3334	0.8334
	10	5	0.3334	0.8334
	11	3	0.2	0.5
	12	4	0.2667	0.6667
	13	4	0.2667	0.6667
	14	4	0.2667	0.6667
	15	6	0.4	1
	16	4	0.2667	0.6667
Bubble Sort	17	2	0.1334	0.4
	18	2	0.1334	0.4
	19	0	0	0
	20	4	0.2667	0.8
	21	3	0.2	0.6
	22	3	0.2	0.6
	23	4	0.2667	0.8
	24	2	0.1334	0.4
Draw Rectangle	25	3	0.2	0.375
	26	7	0.4667	0.875
	27	4	0.2667	0.5
	28	5	0.3334	0.625
	29	5	0.3334	0.625
	30	6	0.4	0.75
	31	7	0.4667	0.875
	32	7	0.4667	0.875

Keterangan: Nomor query dapat dilihat pada Lampiran 3.

Halaman ini sengaja dikosongkan

Lampiran 5

Hasil Uji Coba Tanpa Normalisasi Lingua::IdSplitter

Topik	***#Query***	**#Relevan**	***Precision***	***Recall***
Thread Synchronization	1	11	0.7334	0.8462
	2	11	0.7334	0.8462
	3	11	0.7334	0.8462
	4	6	0.4	0.4616
	5	11	0.7334	0.8462
	6	6	0.4	0.4616
	7	3	0.2	0.2308
	8	10	0.6667	0.7693
Accessing MySQL	9	5	0.3334	0.8334
	10	5	0.3334	0.8334
	11	3	0.2	0.5
	12	4	0.2667	0.6667
	13	4	0.2667	0.6667
	14	4	0.2667	0.6667
	15	6	0.4	1
	16	4	0.2667	0.6667
Bubble Sort	17	2	0.1334	0.4
	18	2	0.1334	0.4
	19	0	0	0
	20	4	0.2667	0.8
	21	3	0.2	0.6
	22	3	0.2	0.6
	23	4	0.2667	0.8
	24	2	0.1334	0.4
Draw Rectangle	25	3	0.2	0.375
	26	7	0.4667	0.875
	27	4	0.2667	0.5
	28	5	0.3334	0.625
	29	5	0.3334	0.625
	30	6	0.4	0.75
	31	7	0.4667	0.875
	32	7	0.4667	0.875

Keterangan: Nomor query dapat dilihat pada Lampiran 3.

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil uji coba dan penjelasan yang telah dipaparkan, dapat disimpulkan bahwa:

1. Proses normalisasi *identifier*menggunakan Lingua::IdSplitter hanya mampu meningkatkan performasistem rekomendasi manakaladata-data diskusi yang ada mengandung *identifier*dengankomposisi term singkatan.
2. Ketidaktepatan pemilihan *identifier* yang layak untuk dinormalisasi menjadi penyebab utama turunnya performa sistem rekomendasi.
3. Pemanfaatan model TF-IDF masih layak digunakan dalam membangun sistem rekomendasi berdasarkan nilai median *recall* umum yang dihasilkan, yakni sebesar 0,67.

5.2 Saran

Guna perbaikanproses normalisasi *identifier* pada sistem rekomendasi, perlu dipertimbangkan adanya proses pemilihan *identifier* yang layak untuk dinormalisasi sebelum dijadikan masukan pada proses normalisasi *identifier*.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

Adomavicius, G. dan Tuzhilin, A., (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *Knowledge and Data Engineering, IEEE Transaction, 17*(6), pp.734-749.

Anderson, A., Huttenlocher, D., Kleinberg, J.,dan Leskovec, J., (2012). Discovering Value from Community Activity on Focused Question Answering Sites: ACase Study of Stack Overflow. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 850-858). ACM.

Arwan, A., Rochimah, S.,dan Akbar, R.J., (2015). Source Code Retrieval on StackOverflow Using LDA. *Information and Communication Technology (ICoICT), 2015 3rd International Conference* (pp. 295-299). IEEE.

Carvalho, N.R., Almeida, J.J., Henriques, P.R.,dan Varanda, M.J., (2015). From Source Code Identifiers to Natural Language Terms. *Journal of Systems and Software, 100*, pp.117-128.

Cegielski, C.G. dan Hall, D.J., (2006). What Makes A Good Programmer?. *Communications of the ACM, 49*(10), pp.73-75.

Cordeiro, J., Antunes, B.,dan Gomes, P., (2012). Context-Based Recommendation to Support Problem Solving in Software Development. *Recommendation Systems for Software Engineering (RSSE), 2012 Third International Workshop* (pp. 85-89). IEEE.

Gosling, J., Joy, B., Steele, G., Bracha, G., dan Buckley, A. (2015). Chapter 3. Lexical Structure,*(online)*,(https://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html, diakses 20 Juli 2016).

Hu, D.J., (2009). Latent Dirichlet Allocation for Text, Images, and Music. *University of California, San Diego*.

Jivani, A.G., (2011). A Comparative Study of Stemming Algorithms. *Int. J. Comp. Tech. Appl, 2*(6), pp.1930-1938.

Krauthammer, M. dan Nenadic, G., (2004). Term identification in the biomedical literature. *Journal of biomedical informatics, 37*(6), pp.512-526.

LaToza, T.D., Venolia, G.,dan DeLine, R., (2006). Maintaining Mental Models: A Study of Developer Work Habits. *Proceedings of the 28th international conference on Software engineering* (pp. 492-501). ACM.

Lawrie, D., Morrell, C., Feild, H.,dan Binkley, D., (2007). Effective identifier names for comprehension and memory. *Innovations in Systems and Software Engineering, 3*(4),pp.303-318.

Mahmoud, A. dan Niu, N., (2011). Source Code Indexing for Automated Tracing. *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 3-9). ACM.

Manning, C.D., Raghavan, P.,dan Schütze, H., (2008). *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

MySQL. (2006). *MySQL stopword list, (online)*, (http://dev.mysql.com/doc/refman/5.0/en/fulltext-stopwords.html, diakses 2Agustus 2016).

Ponzanelli, L., Bacchelli, A.,dan Lanza, M., (2013). Leveraging Crowd Knowledge for Software Comprehension and Development. *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference* (pp. 57-66). IEEE.

Robillard, M.P., Walker, R.J.,dan Zimmermann, T., (2010). Recommendation Systems for Software Engineering. *Software, IEEE, 27*(4), pp.80-86.

Sun Microsystems, (1997). *Java Code Conventions*. Sun Microsystems, Inc.

Vohra, D., Baesens, B., Backiel, A. danVanden Broucke, S., (2015). *Beginning Java Programming: The Object-oriented Approach*. John Wiley & Sons.

Zorychta, D. (18Desember2011). *How do I turn off text antialiasing in this Java function?* Stack Overflow, *(online)*, (http://stackoverflow.com/questions/8549891/how-do-i-turn-off-text-antialiasing-in-this-java-function, diakses 11 Mei 2016).

BIODATA PENULIS

Nanang Fakhrur Rozi, lahir di Gresik, 9 April1988. Merupakan anak ke-3 dari 5 bersaudara.Berasal dari desa Kalirejo kecamatan Dukunkabupaten Gresik. Pendidikan yang telah ditempuh antara lain Sekolah Dasar Negeri KalirejoGresik, Madrasah TsanawiyahYKUI Maskumambang Gresik, dan lulus Madrasah Aliyah YKUI Maskumambang Gresik pada tahun 2005, kemudian melanjutkanpendidikan tinggi padaprogram Diploma III Jurusan Teknologi Informasi di Politeknik Elektronika Negeri Surabaya(PENS) hingga lulus pada tahun 2008. Tahun 2009 kembali melanjutkan studi pada program Diploma IV (Lanjut Jenjang) Jurusan Teknik Informatika di Politeknik Elektronika Negeri Surabayadan lulus pada tahun 2011. Ketika laporan penelitian ini disusun, penulis sedang menempuh studi akhirnya padaProgram Magister Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) Surabaya.Penulis memiliki minat pada bidang rekayasa perangkat lunak serta keterkaitannya dengan bidang kecerdasan buatan.

Informasi Kontak:

E-mail : nfrozy@gmail.com

Situs Web : http://nfrozi.web.id

LinkedIn : http://linkedin.com/in/nfrozi