SISTEM REKOMENDASI TAG PADA POSTING BLOG BERBAHASA INGGRIS MENGGUNAKAN KOMBINASI *CONTENT BASED FILTERING* DENGAN *COLLABORATIVE FILTERING*

SKRIPSI

oleh :

YAUMIL AKHIR

0410960065-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011

UNIVERSITAS BRAWIJAYA

SISTEM REKOMENDASI TAG PADA POSTING BLOG BERBAHASA INGGRIS MENGGUNAKAN KOMBINASI *CONTENT BASED FILTERING* DENGAN *COLLABORATIVE FILTERING*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
dalam Bidang Ilmu Komputer

oleh :
YAUMIL AKHIR
0410960065-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

SISTEM REKOMENDASI TAG PADA POSTING BLOG BERBAHASA INGGRIS MENGGUNAKAN KOMBINASI *CONTENT BASED FILTERING* DENGAN *COLLABORATIVE FILTERING*

oleh:

YAUMIL AKHIR
0410960065-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 20 Januari 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Marji, MT	**Nurul Hidayat,SPd.,MSc**
NIP. 196708011992031001	**NIP. 196804302002121001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 196709071992031001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Yaumil Akhir**
NIM	**: 0410960065-96**
Jurusan	**: Matematika**
Penulis Skripsi berjudul	**: Sistem Rekomendasi Tag pada Posting Blog Berbahasa Inggris Menggunakan Kombinasi *Content Based Filtering* dengan *Collaborative Filtering***

Dengan ini menyatakan bahwa :

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 20 Januari 2011
Yang menyatakan,

(Yaumil Akhir)
NIM. 0410960065-96


UNIVERSITAS BRAWIJAYA

SISTEM REKOMENDASI TAG PADA POSTING BLOG BERBAHASA INGGRIS MENGGUNAKAN KOMBINASI *CONTENT BASED FILTERING* DENGAN *COLLABORATIVE FILTERING*

ABSTRAK

Aplikasi-aplikasi blog yang terkenal seperti Blogspot dan Wordpress telah menyediakan sistem rekomendasi tag pada posting blog namun hanya didasarkan dari tag populer maupun tag sosial saja yang tidak mencerminkan isi posting. Rekomendasi tag yang sesuai dengan isi posting dapat dihasilkan dengan menggunakan *Content Based Filtering*. Hasil rekomendasi dari *Content Based Filtering* dapat diperluas dengan mengkombinasikannya dengan *Collaborative Filtering*. Hasil uji coba yang telah dilakukan menunjukan kombinasi antara *Content Based Filtering* dengan *Collaborative Filtering* dapat menghasilkan presisi yang lebih baik daripada jika diaplikasikan secara sendiri-sendiri.

TAG RECOMMENDATION SYSTEM FOR ENGLISH BLOG POST USING COMBINATION OF CONTENT BASED FILTERING AND COLLABORATIVE FILTERING

ABSTRACT

Popular blog applications like Blogspot and Wordpress has provided a tags recommendation system on blog posts, but only refers to the popular tags or social tags that do not reflect content of the post. Relevant recommendation tags can be generated by using the Content-Based Filtering. The recommendations result of the Content-Based Filtering can be expanded by combining it with Collaborative Filtering. The experiment results show a combination of Content-Based Filtering with Collaborative Filtering can produce a better precision than when applied individually.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul: "**Sistem Rekomendasi Tag pada Posting Blog Berbahasa Inggris Menggunakan Kombinasi *Content Based Filtering* dengan *Collaborative Filtering***"

Skripsi ini diajukan sebagai syarat ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Fakultas MIPA, Program Studi Ilmu Komputer, Jurusan Matematika, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, Penulis mengucapkan terima kasih kepada:

1. Drs. Marji, MT. selaku Dosen Pembimbing skripsi dan selaku Ketua Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
2. Nurul Hidayat, SPd. MSc. selaku Dosen Pembimbing skripsi
3. Edy Santoso, S.Si, M.Kom, selaku Dosen Penasehat Akademik.
4. Dr. Abdul Rouf Al-Ghofari, M.Sc, selaku Ketua Jurusan Matematika FMIPA Universitas Brawijaya.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
7. Orang tua Penulis dan saudara-saudaraku atas segala dukungan materi dan doa restunya kepada Penulis.
8. Rekan-rekan Ilmu Komputer yang telah memberikan dukungannya kepada penulis.
9. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat kami sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini tentunya tidak terlepas dari berbagai kekurangan dan kesalahan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat Penulis harapkan dari berbagai pihak demi penyempurnaan penulisan skripsi ini.

Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Malang, 20 Januari 2011

Yaumil Akhir
Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL


UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tag (kata-kata dan frasa yang memberikan keterangan pada konten) secara luas digunakan dalam *web-log* (blog). Penulis blog (*Blogger*) dapat menggunakannya untuk kategorisasi posting atau identifikasi topik. Pembaca dapat menggunakannya untuk mendapatkan gagasan cepat dari isi dan orientasi dari sebuah blog, untuk pencarian posting, dan sistem blogging dapat menggunakannya pada tampilan blog, pengindeksan, dan layanan rekomendasi blog.

Meskipun pemberian tag (*tagging*) pada blog telah terkenal namun beberapa layanan blog yang sangat terkenal seperti Wordpress (Wordpress.com) hanya menyediakan rekomendasi tag berdasarkan tag populer yang mana tidak mencerminkan isi dari tulisan dan Blogspot (Blogger.com) yang mana dikembangkan oleh Google bahkan tidak menyediakan sistem rekomendasi tag sama sekali. Beberapa peneliti telah mengembangkan aplikasi untuk penyaranan tag tapi sebagian besar dari aplikasi ini terfokus pada prediksi tag sosial. Menurut Michael Hart dkk. pada tahun 2009 tag sosial adalah tag-tag yang paling menarik dan informatif yang diturunkan (melalui aggregasi, pemberian nilai dan penyaringan) dari semua tag yang diberikan oleh beberapa pengguna dari sebuah komunitas online (misalnya del.icio.us, StumbleUpon, Digg) untuk sebuah item konten. Tag-tag sosial memungkinkan anggota komunitas online untuk berbagi dan berinteraksi lebih dengan efektif. Namun, tag sosial ini tidak mencerminkan pola pikir dari penulis konten.

Untuk dapat menghasilkan rekomendasi tag yang sesuai dengan isi tulisan terdapat dua pendekatan yang umum. Pendekatan pertama yaitu dengan *Content Based Filtering* pada *Content Based Filtering* tag-tag rekomendasi didasarkan oleh karakteristik dari konten secara langsung, seperti teks pada konten, judul, tanggal pembuatan dan metadata lainnya. Kelebihan dari pendekatan ini yaitu implementasinya tergolong cukup sederhana dan cepat (Marko Balabanovic dan Yoav Shoham, 1997). Namun metode ini dirasa

kurang komprehensif karena tag-tag yang dihasilkan merupakan tag-tag yang dimuat dalam teks saja dan tidak dapat menemukan alternatif tag-tag lain yang mungkin relevan.

Disisi lain terdapat pendekatan kedua yaitu dengan *Collaborative Filtering* (CF). Pada *Collaborative Filtering* tag-tag rekomendasi dihasilkan secara kolaborasi dari kumpulan data selera pengguna yang jumlahnya sangat besar. Dari data yang sangat besar tersebut dibuat menjadi rangkuman selera rata-rata pengguna. Dasar yang digunakan pada pendekatan ini yaitu bahwa pengguna yang menyukai suatu hal pada waktu yang lampau akan menyukai hal yang sama dimasa depan. Kelebihan dari pendekatan ini yaitu dapat memberikan rekomendasi yang relevan kepada pengguna meski tidak terdapat pada profil dari pengguna itu sendiri. Kelebihan lain dari *Collaborative Filtering* yaitu dapat diaplikasikan pada domain dimana tidak terdapat konten atau konten sulit dianalisa oleh komputer. Sedangkan kekurangan dari pendekatan ini yaitu *sparsity*. *Sparsity* yaitu data yang jumlahnya besar namun kepadatannya rendah. Data seperti ini membutuhkan sumberdaya komputasi yang besar untuk dapat diolah. Selain itu pendekatan ini tidak dapat berjalan jika belum terdapat data selera dari pengguna atau jika jumlahnya terlalu sedikit, permasalahan ini disebut *first rater* atau *cold start*. Kekurangan lain dari pendekatan ini yaitu tidak dapat mengetahui konteks alasan mengapa sebuah item disukai (Reyn Nakamoto dkk., 2007).

Sangat jelas jika kedua pendekatan tersebut jika diaplikasikan secara terpisah tidak akan cukup untuk menghasilkan rekomendasi yang baik. Untuk itu kedua pendekatan tersebut dikombinasikan untuk dapat saling melengkapi (Prem Melville dkk., 2001). Permasalahan *first rater* pada *Collaborative Filtering* dapat diatasi dengan *Content Based Filtering* karena *Content Based Filtering* hanya membutuhkan konten yang dianalisa untuk menghasilkan tag rekomendasi. Selain itu tag-tag yang dihasilkan dari *Collaborative Filtering* dapat memperluas jangkauan dari rekomendasi dari tag yang dihasilkan dengan *Content Based filtering*.

Berdasarkan latar belakang yang dipaparkan maka skripsi ini diberi judul "**Sistem Rekomendasi Tag pada Posting Blog Berbahasa Inggris Menggunakan Kombinasi *Content Based Filtering* dengan *Collaborative Filtering***".

1.1 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana merancang sistem rekomendasi tag untuk posting blog dengan kombinasi algoritma *Content Based Filtering* dengan algoritma *Collaborative Filtering*.
2. Bagaimana melakukan implementasi rancangan sistem rekomendasi yang telah disusun pada sebuah sistem berbasis web.
3. Bagaimana melakukan uji coba dan menganalisa hasil uji coba dari sistem rekomendasi yang telah dibuat.

1.2 Batasan Masalah

Dari permasalahan, berikut ini diberikan beberapa batasan-batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Sistem rekomendasi yang dibangun ditujukan untuk posting blog.
2. Bahasa yang digunakan dalam posting blog yaitu bahasa Inggris.
3. Masukan dari sistem ini yaitu *plain text*, dimana tag-tag HTML akan diabaikan.
4. Sistem yang akan dibangun dibuat berbasis web.

1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan Tugas Akhir ini antara lain:

1. Merancang sistem rekomendasi tag untuk posting blog dengan kombinasi algoritma *Content Based Filtering* dengan algoritma *Collaborative Filtering*.
2. Melakukan implementasi rancangan sistem rekomendasi yang telah disusun pada sebuah sistem berbasis web.
3. Melakukan uji coba dan menganalisa hasil uji coba dari sistem rekomendasi yang telah dibuat.

1.4 Manfaat

1. Dapat memberikan rekomendasi tag kepada penulis blog saat membuat atau menyunting posting blog.
2. Menjaga konsistensi tag antara pengguna yang satu dengan yang lain maupun tag-tag yang ditulis oleh pengguna itu sendiri.

1.5 Sistematika Penulisan

Sistematika penulisan tugas akhir terbagi atas 5 bab, yaitu sebagai berikut:

BAB I PENDAHULUAN

Di dalam bab ini diuraikan mengenai latar belakang masalah yang akan dibahas, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan.

BAB II DASAR TEORI

Pada bab ini dijelaskan tentang teori-teori yang mendasari pembuatan sistem rekomendasi tag.

BAB III METODE DAN PERANCANGAN SISTEM

Bab ini menjelaskan metode-metode yang dipakai dan tahapan perancangan dalam membuat aplikasi ini.

BAB IV IMPLEMENTASI DAN UJI COBA

Bab ini membahas implementasi dari rancangan yang telah dibuat dan uji coba yang dilakukan.

BAB V PENUTUP

Bab ini merupakan bab terakhir yang berisi kesimpulan dan saran dari pelaksanaan penelitian.

BAB II
DASAR TEORI

2.1 Sistem Rekomendasi

Sistem rekomendasi adalah teknologi pemilihan informasi yang sifatnya personal yang digunakan untuk memprediksi apakah seorang pengguna tertentu akan tertarik dengan sesuatu (Han.dkk, 2005). Tujuan utama dari sistem rekomendasi adalah memperkirakan item atau bagian mana yang paling disukai atau yang paling berguna. Dengan melakukan tugas tersebut maka sistem ini bisa mengurangi informasi yang berlebihan yang membuat orang/pengguna bingung untuk menentukan pilihan.

Sistem rekomendasi dibangun berdasar pada perpaduan dari kecerdasan buatan, interaksi manusia dan komputer, sosiologi, *information retrieval* dan teknologi web, rekomendasi yang berasal dari teman, kolega, publikasi dan sumber lainnya untuk membantu menentukan pilihan.

Berbagai teknik *data mining* seperti *association rule* dan *market basket analysis* (Weiyang, 2000), hingga *clustering* telah digunakan dalam membangun sebuah sistem rekomendasi. Namun, secara garis besar teknik yang digunakan dalam sistem rekomendasi ada dua macam, yakni *Content-Based Filtering* dan *Collaborative Filtering* (Schafer, 2001).

2.2 Tag

Tag adalah sebuah kata kunci atau frasa pendek yang diberikan penulis pada artikel untuk mendeskripsikan atau mengidentifikasi konten, subjek, orang-orang yang terlibat, jenis artikel, tema yang dibahas (technorati.com, 2009). Manfaat dari pemberian tag yaitu untuk pembaca dapat menggunakannya untuk mendapatkan gagasan cepat dari isi dan orientasi dari sebuah konten, untuk pencarian dan untuk penyedia konten dapat digunakan sebagai tampilan dengan *tag cloud*, pengindeksan, dan layanan rekomendasi.

Berikut adalah jenis-jenis tag menurut Scott A. Golder dan Bernardo A. Huberman (2005):

1. Mengidentifikasi tentang apa (atau tentang siapa). Tag mengidentifikasi topik item yang di-*bookmark*. Item ini termasuk kata benda umum dari berbagai tingkat kekhususan, dan juga banyak kata benda, dalam hal membahas isi orang atau organisasi.
2. Mengidentifikasi apakah itu. Tag-tag dapat mengidentifikasi hal macam apa yang dari item yang di-*bookmark*, selain itu juga digunakan untuk apa, misalnya, artikel, blog dan buku.
3. Mengidentifikasi siapakah yang memiliki. Beberapa tag kadang dibuat menurut siapa yang memiliki atau menciptakan konten tersebut. Mengingat terdapatnya pengguna yang memiliki popularitas yang tinggi, mengidentifikasi kepemilikan konten menjadi sangat penting.
4. Menentukan kategori. Beberapa tag terkadang tidak berdiri sendiri dan membentuk kategori sendiri, memperbaiki atau memenuhi syarat kategori yang ada.
5. Mengidentifikasi kualitas atau karakteristik. Kata sifat seperti menakutkan, lucu, bodoh, tag inspirasional menurut pendapat Tagger dari isi.
6. Referensi diri. Tag-tag yang diawali dengan "*my*" seperti "*mycomments*" mengidentifikasi konten dalam hal hubungannya dengan *tagger*.
7. Pengorganisasian tugas. Ketika mengumpulkan informasi yang terkait dengan pelaksanaan sebuah tugas, informasi yang ditemukan harus ditandai sesuai dengan tugas tersebut, untuk mengelompokan informasi tersebut bersama-sama. Contohnya termasuk *toread*, *jobsearch*. Pengelompokan informasi yang berhubungan dengan tugas dapat menjadi bagian penting dari pengorganisasian dalam melakukan tugas.

2.3 Blog

Blog adalah publikasi di web pribadi dari pikiran dan pendapat yang frekuensinya sering, kronologis ditujukan untuk dibaca pengguna internet lainnya. Istilah blog diciptakan pada akhir 1990-an, berasal dari istilah "*Web log*". Ada jutaan blog di internet. Selain pikiran dan pendapat, banyak *blogger* (istilah untuk orang yang menulis blog) juga menggunakan blog untuk merekomendasikan buku-buku, musik, dan link ke situs lain di *World Wide Web* (WWW). Bentuk sebuah blog adalah sangat tergantung pada

individu yang menyimpannya. Kebanyakan blog adalah campuran dari apa yang terjadi dalam kehidupan seseorang dan apa yang mereka rasakan tentang hal-hal yang mereka lihat di web. Dalam hal ini, blog adalah semacam gabungan antara buku harian dan panduan (Microsoft Encarta Online Encyclopedia, 2009).

2.4 Notasi Struktur Data

Notasi-notasi penulisan struktur data pada tulisan ini menggunakan format JSON (*Javascript Object Notation*).

JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau larik asosiatif (*associative array)*.
2. Daftar nilai terurutkan. Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Bentuk-bentuk notasi pada JSON:

1. *Object*
   ```
   {}
   { members }
   ```
2. *members*
   ```
   pair
   pair , members
   ```
3. *pair*
   ```
   string : value
   ```
4. *array*
   ```
   []
   [ elements ]
   ```
5. *elements*
   ```
   value
   value , elements
   ```
6. *value*
   ```
   string
   number
   object
   array
   true
   false
   null
   ```

2.5 Content Based Filtering

Content Based filtering merupakan metode dalam sistem rekomendasi yang mana untuk menghasilkan item-item yang direkomendasikan didasarkan dari karakteristik yang melekat pada objek permasalahan. Misal pada sebuah dokumen, maka karakteristik yang digunakan sebagai acuan untuk menghasilkan rekomendasi bisa dari teks pada dokumen, meta data, nama berkas, nama pembuatnya dan karakteristik lain yang berkenaan langsung dokumen tersebut.

2.5.1 Kelebihan

Kelebihan dari Metode *Content Based filtering* adalah sebagai berikut :

1. Sederhana dan cepat (Marko Balabanovic dan Yoav Shoham, 1997)
2. Pengguna dapat mengetahui alasan mengapa item dianggap relevan terhadap mereka karena isi dari setiap item dapat diketahui dari representasinya (Peretz Shoval dkk., 2008).
3. Tidak dipengaruhi oleh masalah-masalah pada pemfilteran kolaboratif pada umumnya seperti *cold start* dan *sparsity* (Claypool dkk., 1999).

2.5.2 Kekurangan

Beberapa kekurangan dari metode *Content Based filtering* yaitu :

1. Sama sekali tidak memperhatikan aspek seni, semua informasi multimedia (termasuk text dalam gambar) dan faktor jaringan seperti waktu untuk memuat (Marko Balabanovic dan Yoav Shoham, 1997).
2. Tidak mampu menangkap relasi yang lebih kompleks pada level semantik yang lebih dalam, berdasarkan pada tipe atribut yang di asosiasikan pada objek terstruktur dari teks (Dai dan Mobasher, 2001). Konsekuensinya banyak item yang terlewat dan banyak item-item yang tidak relevan yang diambil (Blair dan Maron, 1985).
3. Kesulitan dalam menentukan perbedaan antara informasi yang berkulitas tinggi dan yang berkulitas rendah dikarenakan

keduanya bisa saja direpresentasikan dengan term yang sama. Seiring dengan meningkatnya jumlah item, jumlah item yang memiliki kategori *content-based* sama semakin meningkat yang mana menurunkan keefektifan dari pendekatan *content-based* (Claypool dkk., 1999).

4. Memerlukan komputasi yang mahal bahkan tidak mungkin untuk dilakukan pada item multimedia yang mana tidak memiliki teks (Peretz Shoval dkk, 2008).

2.6 Ekstraksi term dengan POS *Tagging*

Dari definisi tag yang mana bisa diartikan sebagai term ataupun *compound term*, maka untuk mendapatkan kandidat-kandidat tag diperlukan proses ektraksi term dari teks sumber.

Salah satu metode ektraksi term dari teks yaitu Topia Termextract yang dikembangkan oleh Stephan Richter, Russ Ferriday dan Zope Community pada tahun 2009 dengan bahasa pemrograman Python.

Pada Topia Termextract proses ekstraksi dilakukan dengan tiga tahap utama yaitu tahap tokenisasi, yaitu memecah teks menjadi satuan yang lebih kecil untuk diolah selanjutnya dan tahap *Part Of Speech Tagging* (POS *Tagging*) yaitu menentukan kategori leksikal token-token tersebut dalam kalimat dan tahap terakhir yaitu ekstrasi kata-kata tunggal dan majemuk berdasarkan kriteria yang telah ditentukan sebelumnya.

Algoritma proses ektraksi dengan menggunakan Topia Termextract dapat dilihat pada Gambar 2.1. Pada tahap pertama teks ditokenisasi dengan menggunakan fungsi *tokenize* seperti pada bagian (1) pada Gambar 2.1. Token-token pada bagian (1) kemudian menjadi masukan pada fungsi tag yaitu fungsi yang berguna untuk mengasosiasikan POS Tag pada setiap token yang ditemukan. Proses ini digambarkan pada bagian (2). Proses ekstraksi terdapat pada bagian (3) dengan menggunakan fungsi *extract* dimana token-token yang telah memiliki POS tag dianalisa dan difilter sesuai dengan kriteria. Hasil akhir dari algoritma ini yaitu sebuah larik asosiatif yang berisi pasangan antara term dengan frekuensi kemunculannya dalam teks.

```
Fungsi Termextract
Input  : string teks
Output : array result

token       : array token [token, ...]
taggedTerms : array [[term, tag, norm], ...]
result      : array {word:occur,...}

token = tokenize(teks)                                    (1)

taggedTerms = tag(token)                                  (2)

result = extract(taggedTerms)                             (3)

return result
```

Gambar 2.1 Algoritma Ekstraksi Term dengan POS *Tagging*

2.6.1 Tokenisasi

Tokenisasi merupakan proses memotong sebuah string menjadi token-token. Dalam proses tokenisasi juga bisa dilakukan pembersihan karakter-karakter tertentu seperti karakter pungtuasi. Token merupakan sebuah perwujudan dari urutan karakter pada dokumen tertentu yang di kelompokkan sebagai satuan semantik yang berguna untuk proses selanjutnya.

Contoh proses tokenisasi dapat dilihat pada Gambar 2.2, dimana input teks dipotong berdasar karakter spasi dan membuang semua tanda baca yang ada.

Input: Friends, Romans, Countrymen, lend me your ears;
Output: | Friends | Romans | Countrymen | lend | me | your | ears |

Gambar 2.2 Tokenisasi

Detil Algoritma tokenisasi yang digunakan dalam proses ektraksi term dapat dilihat pada Gambar 2.3. Pada bagian (1) teks dipecah dengan menggunakan fungsi bawaan dari PHP yaitu *explode*. Dengan fungsi ini teks dipecah berdasarkan karakter spasi menjadi larik token-token. Token-token yang kosong dilewati dan tidak ikut proses berikutnya (2). Pada bagian (3) dicari kata-kata yang memiliki simbol di dalamnya seperti kata '*post-tag*'. Kata-kata

seperti ini kemudian dipecah dan setiap bagiannya dimasukan ke dalam larik kembalian yang berisi token-token.

```
Fungsi Tokenize
Input  : string teks
Output : array terms

tokens  : array[token,...]
terms   : array[token,...]
matches : array[token,...]

tokens = explode (' ', teks)                              (1)

FOREACH token pada array tokens
   IF token == ''                                         (2)
      CONTINUE

   IF tidak memiliki simbol
      simpan token dalam array terms                      (3)
   ELSE
      pecah token pada simbol ke array matches            (4)
      FOREACH match pada array matches
         IF match != ''
             simpan token dalam array terms               (5)

return array terms
```

Gambar 2.3 Algoritma Tokenisasi

2.6.2 Part-of-speech Tagging

Part-of-speech atau kategori leksikal yaitu sebuah kategori kata atau lebih tepatnya satuan leksikal secara linguistik. Setiap bahasa memiliki kategori leksikal yang berbeda-beda. Misalnya Bahasa Inggris yang memiliki delapan kategori leksikal utama, yaitu:

1. *Noun*: kata benda
2. *Pronoun*: kata ganti benda
3. *Adjective*: kata sifat
4. *Verb*: kata kerja ataupun keadaan
5. *Adverb*: kata keterangan
6. *Preposition*: kata depan
7. *Conjunction*: kata hubung
8. *Interjection*: kata seru

Dari delapan kategori leksikal utama tersebut, ahli linguistik modern dapat mengklasifikasikan kata-kata dalam Bahasa Ingggris menjadi kategori leksikal yang lebih spesifik sesuai dengan kebutuhan.

Part-of-speech tagging (POS *tagging* atau POST) yang mana juga disebut penandaan gramatikal atau *word category disambiguation* merupakan proses menandai kata-kata pada teks dengan kategori leksikal yang bersesuaian berdasarkan dari makna maupun dari konteksnya.

Terdapat dua pendekatan utama dalam melakukan penandaan kategori leksikal secara otomatis yaitu dengan pendekatan statistik dan *rule based*. Pendekatan statistik biasanya memiliki akurasi dan kecepatan yang lebih baik dari pada pendekatan *rule based*. Namun pendekatan *rule based* memiliki beberapa kelebihan yang tidak dimiliki oleh pendekatan statistik seperti implementasi yang lebih mudah, pemanfaatan penyimpanan informasi yang lebih kecil dan portabilitas bahasa yang lebih baik (Eric Brill, 1992).

Algoritma POS *Tagging* yang digunakan dalam proses ektraksi term dapat dilihat pada Gambar 2.4. Masukkan dalam fungsi ini yaitu larik token-token pada proses tokenisasi dan *path* posisi berkas *lexicon*. *Lexicon* yang dimaksud pada fungsi ini yaitu berkas yang berisi daftar term-term umum berikut asosiasi dengan POS tag nya. Term-term umum ini dirasa cukup untuk mewakili seluruh kata-kata dalam Bahasa Inggris. Berkas *lexicon* yang digunakan pada penelitian ini telah disediakan oleh Topia Termextract dalam berkas *english-lexicon.txt* yang berisi 93.718 term-term dalam Bahasa Inggris berikut POS tagnya. Langkah awal yaitu memuat isi berkas *lexicon* ke dalam larik asosiatif (1). Kemudian setiap token di cari dalam larik *lexicon* tersebut. Jika ditemukan maka tag token tersebut adalah tag yang bersesuaian dengan yang ada pada larik *lexicon*. Jika tidak ditemukan maka kata dianggap kata benda. Proses tersebut dapat dilihat pada bagian (2) dan (3). Token-token yang telah ditentukan POS tag nya kemudian disimpan pada larik *tagged_term* (4). Langkah terakhir yaitu melakukan normalisasi pada term-term yang telah ditemukan pada larik *tagged_term*. Proses normalisasi pada bagian (5) yang dilakukan antara lain:

1. Perbaikan tag pada kata benda.
2. Memastikan kata benda pada awal kalimat.
3. Menemukan kata kerja setelah kata kerja bantu.
4. Normalisasi kata majemuk.

Daftar POS Tag yang dimanfaatkan dalam algoritma ini dapat dilihat pada Lampiran 1 di bagian akhir tulisan ini.

```
Fungsi tag
Input  : array token, string path_lexicon
Output : array tagged_terms

tagged_term : array [[term, tag, norm], ...]
lexicon     : array {term:tag, ...}


// inisialisasi
lexicon = load_file (path_lexicon)                    (1)

// fase pertama
FOREACH token
   cari token dalam array lexicon
   IF ditemukan
      tag = tag term pada lexicon                     (2)
   ELSE
      tag = kata benda                                (3)

   simpan term dalam array tagged_term                (4)


// fase kedua, normalisasi
FOREACH term pada tagged_term                         (5)
   perbaiki tag kata benda
   pastikan kata benda pada awal kalimat
   temukan kata kerja setelah modal
   normalisasi kata majemuk

return array tagged_term
```

Gambar 2.4 Algoritma POS *Tagging*

2.6.3 Ekstraksi term

Langkah terakhir dalam proses ektraksi term dengan algoritma POS *Tagging* yaitu mengekstrak kata-kata tunggal dan kata-majemuk yang memenuhi kriteria. Kriteria yang biasanya diaplikasikan yaitu frekuensi kemunculan minimum. Detil dari fungsi ekstraksi term dapat dilihat pada Gambar 2.5.

```
Fungsi Extract
Input  : array tagged_term [[term, tag, norm], ...]
Output : array terms [[term, occur, len], ...]
Konstanta : integer min_frek_tunggal
            integer min_frek_majemuk
multiterm  : array [[term, norm], ...]
terms      : array {norm:frekuensi}
multiterms : array [term, ...]
filtered   : array [[word, occur, len], ...]

state = SEARCH
WHILE tagged terms berisi
   term, tag, norm = pop(tagged_term)                    (1)
   IF state == SEARCH AND tag merupakan kata benda
      state = NOUN                                       (2)
      masukkan ke dalam array multiterm
      terms[tag] += 1

   ELSE IF state == SEARCH
        AND tag merupakan kata adjektiva
        AND term diawali dengan huruf kapital
      state = NOUN                                       (3)
      masukkan ke dalam array multiterm
      terms[tag] += 1

   ELSE IF state == NOUN
        AND tag merupakan kata benda
      masukkan ke dalam array multiterm                  (3)
      terms[tag] += 1

   ELSE IF state == NOUN
        AND tag bukan merupakan kata benda
      state = SEARCH
      IF count(multiterm) > 1                            (4)
         FOREACH term pada multiterm
            simpan term dalam array multiterms
         word = implode (' ', multiterms)
         terms[word] += 1
         multiterms = array()
      multiterm = []

   FOREACH word dan occur pada array terms
      len = jumlah kata pada word
      IF (len == 1 AND occur >= min_frek_tunggal)
        OR (len > 1 AND occur >= min_frek_majemuk)
         simpan word ke dalam array filtered             (5)

return array filtered
```

Gambar 2.5 Algoritma Ektraksi Term

Pada fungsi ektraksi ini setiap tag dari masukan yaitu larik *tagged_term* dari proses POS *Tagging* dianalisa satu persatu. Setiap term yang paling atas dikeluarkan dari larik *tagged_term* dengan fungsi *pop* (5). Pada bagian (2) dan (3) dicari term-term dengan tag kata benda dan kata sifat yang mana pada proses berikutnya kata benda yang berdekatan dikumpulkan dan digabung menjadi kata majemuk (4). Setiap proses penyimpanan juga diikuti dengan menghitung frekuensi kemunculan term pada teks. Setelah semua kata-kata tunggal dan majemuk berikut frekuensinya ditemukan, langkah terakhir yaitu menyaring kata-kata tersebut berdasarkan kriteria frekuensi minimal (5). Kata-kata yang lolos proses seleksi kemudian mejadi kembalian dari fungsi ektraksi ini.

2.7 Pembobotan Term

Pembobotan term merupakan proses mengasosiasikan bobot pada term dengan aturan tertentu sehingga dapat ditentukan bahwa suatu term lebih berarti dibandingkan term lainnya.

Contoh beberapa aturan yang bisa digunakan dalam menentukan bobot sebuah term antara lain :

1. Berdasarkan Frekuensi, yaitu term yang memiliki frekuensi lebih besar pada suatu teks atau kumpulan term memiliki bobot yang lebih tinggi karena dianggap lebih penting (Christopher D. Manning, 2009).
2. Berdasarkan Posisi, yaitu dengan menentukan bobot term tergantung pada posisinya pada teks. Asumsi yang biasa diambil yaitu bahwa term-term pada judul biasanya memiliki nilai lebih dibandingkan term-term pada isi paragraf. (Christopher D. Manning, 2009).
3. Berdasarkan kapitalisasi, yaitu dengan memberikan nilai berdasarkan jenis kapitalisasi dari term. Misal dengan memberikan bobot lebih tinggi kepada term yang ditulis dengan seluruhnya huruf kapital dibandingkan term yang hanya dikapitalisasi pada huruf depannya saja atau yang hanya menggunakan huruf kecil. (Hjortur Stefan Olafsson, 2007).
4. Berdasarkan jenis kata, yaitu menentukan bobot yang berbeda antara kata majemuk dan kata tunggal. Hjortur Stefan Olafsson, 2007).

Dari beberapa mekanisme pembobotan tersebut dapat diaggregat dengan menggunakan rumus tertentu sehingga dapat

menghasilkan nilai akhir untuk setiap termnya. Beberapa skema pembobotan yang bisa digunakan antara lain:

- Pembobotan berdasarkan jumlah kata tiap term dibagi dengan jumlah kata maksimal yang ditemukan. Pembagian dengan maksimal jumlah kata bertujuan untuk normalisasi bobot. Rumus perhitungan bobot tergantung jumlah kata ini dapat dilihat pada Persamaan 2.1.

$$bobot = \frac{jumlah\ kata}{maks\ jumlah\ kata} \tag{2.1}$$

- Pembobotan berdasarkan frekuensi kata dibagi frekuensi maksimal kata ditemukan. Pembagian dengan frekuensi maksimal kata bertujuan untuk normalisasi bobot. Rumus perhitungan bobot tergantung frekuensi kata ini dapat dilihat pada Persamaan 2.2.

$$bobot = \frac{frekuensi\ kata}{maks\ frekuensi\ kata} \tag{2.2}$$

- Pembobotan berdasarkan gabungan antara panjang kata yang telah ternormalisasi dan frekuensi kata yang juga telah ternormaliasi. Bobot yang ditemukan inipun kemudian dibagi lagi dengan maksimal bobot yang ada untuk tujuan normalisasi sehingga bisa sebanding jika digabungkan dengan bobot dari proses *Collaborative Filtering*. Rumus perhitungan bobot dengan gabungan antara panjang kata dan frekuensi kata ini dapat dilihat pada Persamaan 2.3.

$$bobot = \frac{\frac{jumlah\ kata}{maks\ jumlah\ kata} + \frac{frekuensi\ kata}{maks\ frekuensi\ kata}}{maks\ bobot} \tag{2.3}$$

2.8 Collaborative Filtering

Collaborative filtering (CF) adalah proses penyaringan informasi atau pola dengan menggunakan teknik-teknik yang melibatkan kolaborasi antara beberapa agen, sudut pandang, sumber data, dan lain-lain. Aplikasi penyaringan kolaboratif biasanya melibatkan data set yang sangat besar. Metode penyaringan kolaboratif telah diterapkan ke berbagai jenis data termasuk sensing dan data pemantauan seperti dalam eksplorasi mineral atau pemantauan lingkungan di daerah yang luas, data keuangan seperti layanan keuangan lembaga yang mengintegrasikan banyak sumber-sumber keuangan atau *ecommerce* dan web 2.0 dimana fokusnya kepada data pengguna.

Metode membuat prediksi (penyaringan) secara otomatis tentang ketertarikan pengguna dengan mengumpulkan selera (*taste*) informasi dari banyak pengguna (berkolaborasi). Asumsi yang mendasari pendekatan CF adalah bahwa mereka yang setuju di masa lalu cenderung setuju lagi di masa depan. Sebagai contoh, sebuah sistem penyaringan kolaborasi atau sistem rekomendasi untuk selera musik bisa membuat prediksi tentang musik yang mungkin disukai dari seorang pengguna didasarkan pada daftar selera musik pengguna tersebut (musik yang disukai atau tidak). Prediksi yang diberikan merupakan prediksi spesifik untuk pengguna tersebut, tetapi menggunakan informasi yang diperoleh dari banyak pengguna lainnya.

2.8.1 Metodologi

Sistem Penyaringan Kolaboratif biasanya mengambil dua langkah:

1. Cari untuk pengguna yang memiliki pola penilaian yang sama dengan pengguna aktif (user yang prediksi adalah untuk).
2. Gunakan penilaian dari orang-orang seperti hati pengguna yang ditemukan pada langkah 1 untuk menghitung prediksi untuk pengguna yang aktif.

Metode lain yaitu *item based collaborative filtering* yang dipopulerkan oleh Amazon.com (dimana pengguna yang membeli *x* juga membeli *y*) dan pertama kali diusulkan Vucetic dan Obradovic pada tahun 2000, hasil dalam item-sentris cara:

1. Membangun sebuah item-item menentukan matriks hubungan antara pasang item
2. Menggunakan matriks, dan data pengguna saat ini, menyimpulkan dengan seleranya

2.8.2 Jenis

Dalam Metode *Collaborative filtering* digunakan dua pendekatan dalam menghasilkan rekomendasi:

1. Active Filtering

Pengumpulan informasi dilakukan dengan meminta pengguna untuk memberikan rating pada item secara explisit, rating ini akan digunakan dan dibandingkan dengan rating dari pengguna lainnya untuk menentukan item lain yang mungkin disukai oleh pengguna.

2. Passive filtering

Pengumpulan informasi dilakukan secara implisit. Filter implisit ini kemudian digunakan untuk mengetahui hal lain yang akan disukai oleh pengguna dan merekomendasikan item-item potensial. Pemfilteran implisit memerlukan aksi dari pengguna untuk menentukan tingkat nilai dari konten spesifik seperti:

- pembelian item
- berulang-ulang memanfaatkan, mencetak, menyimpan sebuah item
- mereferensi atau membuat tautan ke sebuah website

2.8.3 Kelebihan

Kelebihan dari Metode *Collaborative filtering* adalah sebagai berikut :

1. Dapat menemukan item yang relevan terhadap pengguna meskipun item itu tidak berada pada konten.
2. Dapat diaplikasikan pada domain data dimana konten tidak ada atau sulit dianalisa oleh komputer.

2.8.4 Kekurangan

Kekurangan dari Metode *Collaborative filtering* adalah sebagai berikut :

1. *Sparsity.*
2. Permasalahan *first-rater* dan *cold start.*
3. Tidak memperhatikan konteks alasan sebuah item disukai, dimana dua orang pengguna bisa menyukai item yang sama dikarenakan alasan yang berbeda. Sebagai contoh seorang pengguna mungkin menyukai item tersebut karena dirasa lucu dan pengguna lain menyukainya karena dirasa informatif dan ditulis secara jelas (Reyn Nakamoto dkk., 2007).

2.9 *Item-based Top-N Recommendation*

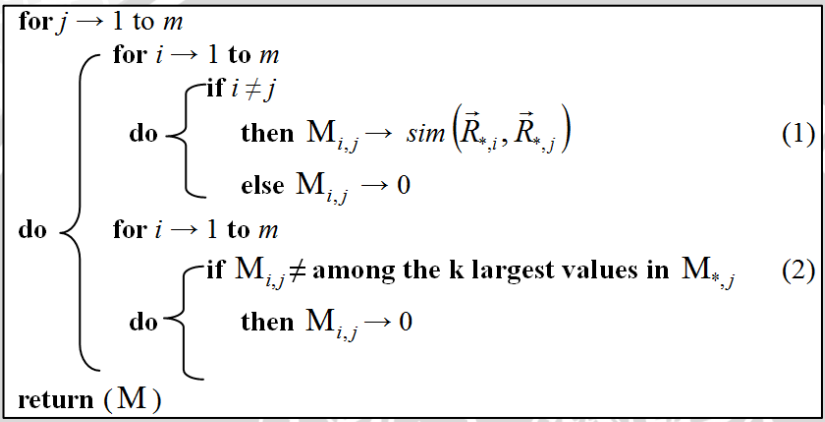
Algoritma *model-based top-N recommendation* yang mana menggunakan kesamaan antara item untuk menghitung hubungan antara item yang berbeda. Algoritma ini dikembangkan oleh Mukund Deshpande dan George Karypis pada tahun 2004. Motivasi utama di balik algoritma ini adalah kenyataan bahwa pelanggan sebenarnya lebih suka membeli item yang mirip dengan item yang ia beli di masa lalu, dengan demikian, dengan menganalisis informasi riwayat pembelian (seperti diwakili dalam matriks user-item) dapat secara otomatis mengidentifikasi himpuan item yang serupa dan menggunakannya untuk membentuk rekomendasi top N. Algoritma ini menyerupai skema *item-based* yang dikembangkan sebelumnya (Shardanand dan Maes 1995; Kitts dkk., 2000) tetapi berbeda dalam beberapa aspek kunci yang terkait dengan cara kesamaan antara item yang berbeda dihitung dan bagaimana kemiripan ini digabungkan untuk memperoleh rekomendasi akhir.

Algoritma ini terdiri dari dua komponen berbeda. Komponen pertama yang membangun sebuah model yang menggambarkan hubungan antara item yang berbeda, sedangkan komponen kedua mengaplikasikan model yang telah dihitung sebelumnya untuk menghasilkan rekomendasi top-N untuk pengguna yang bersangkutan.

2.9.1 Membangun Model

Model yang digunakan oleh algoritma rekomendasi top-N berbasis item dibuat dengan menggunakan algoritma yang ditunjukkan pada Gambar 2.6. Input untuk algoritma ini adalah matriks user-item R yang berukuran $n \times m$ dan sebuah parameter k yang menentukan jumlah kemiripan item-ke-item yang akan

disimpan untuk setiap item. Keluarannya adalah model itu sendiri, yang dilambangkan dengan sebuah matrik *M* berukuran *m* × *m* yang mana kolom ke-*j* menyimpan *k* item yang paling mirip ke item *j*. Secara khusus, jika $M_{i,j} > 0$, maka item ke *i* berada di antara *k* item yang paling mirip *j* dan nilai $M_{i,j}$ menunjukkan tingkat kesamaan antara item *j* dan *i*.



Gambar 2.6 Algoritma Pembuatan Model

Pemberian parameter *k* pada *M* didorong karena pertimbangan kinerja dan proporsi antara kinerja dan kualitas. Dengan menggunakan nilai *k* yang kecil, dapat dipastikan bahwa *M* adalah sangat *sparse* (jarang) dan dengan demikian dapat disimpan dalam memori utama bahkan di lingkungan pemfilteran kolaboratif dan aplikasi dimana *m* adalah sangat besar. Namun, jika *k* terlalu kecil, maka model yang dihasilkan hanya akan berisi informasi yang terbatas untuk membangun rekomendasi, dan dengan demikian dapat berpotensi menyebabkan kualitas lebih rendah. Tapi, berdasarkan eksperimen yang dilakukan, cukup nilai-nilai *k* kecil ($10 \leq k \leq 30$) memberikan hasil yang baik dan nilai-nilai yang lebih tinggi atau lebih rendah tidak jauh berbeda.

Algoritma yang sebenarnya untuk membangun *M* adalah cukup sederhana. Untuk setiap item *j*, algoritma menghitung kesamaan antara *j* dan item yang lain dan menyimpan hasilnya dalam kolom ke-*j* dari *M* (bagian 1 pada Gambar 2.6). Setelah kesamaan ini telah dihitung, kemudian mulai me-nol-kan semua entri dalam kolom ke-*j* dari *M* yang mengandung nilai-nilai lebih kecil daripada nilai-nilai kemiripan *k* terbesar dalam kolom tersebut.

Matriks *M* yang mengandung entri tidak nol paling banyak sejumlah *k* untuk setiap kolom menjadi model final algoritma berbasis item ini. Dengan konstruksi (bagian 2 pada Gambar 2.6) algoritma memastikan bahwa item tertentu tidak berisi dirinya dalam salah satu item *k* yang paling mirip. Hal ini dilakukan untuk memastikan bahwa suatu item yang telah dibeli tidak akan direkomendasikan lagi. Rekomendasi seperti nilai kecil karena dibutuhkan item yang disarankan untuk berbeda dari item dalam keranjang pembelian pengguna aktif.

2.9.2 Mengukur Kesamaan antara item

Salah satu cara untuk menghitung kesamaan antara dua item adalah dengan menggunakan kesamaan berbasis Kosinus (*Cosine Based Similarity*) yaitu dengan memperlakukan setiap item sebagai vektor dalam lingkup pelanggan dan menggunakan kosinus antara vektor sebagai ukuran kesamaan. Secara formal, jika *R* adalah *n* × *m* matriks user-item, maka kesamaan antara dua item *i* dan *j* didefinisikan sebagai kosinus dari *n* vektor dimensi yang sesuai dengan kolam ke *i* dan kolom ke-*j* dari matriks *R*. Kosinus antara vektor diberikan oleh Persamaan 2.4.

$$\text{sim}(I_i, I_j) = \cos\left(\vec{R}_{*,i}, \vec{R}_{*,j}\right) = \frac{\vec{R}_{*,j} \cdot \vec{R}_{*,j}}{\left\|\vec{R}_{*,i}\right\|_2 \left\|\vec{R}_{*,j}\right\|_2} \tag{2.4}$$

Dimana '·' menunjukkan operasi vektor *dot-product*. Perlu diperhatikan karena fungsi kosinus mengukur sudut antara kedua vektor itu merupakan fungsi kesamaan yang simetris. Akibatnya, item yang sering dibeli akan cenderung mirip dengan item lain yang sering dibeli dan tidak mirip dengan item yang jarang dibeli, dan begitu juga sebaliknya.

Dalam bentuknya yang paling sederhana, deretan *R* dapat mengacu kepada informasi pembelian biner asli, dalam hal ini, kesamaan fungsi kosinus memperlakukan pelanggan yang membeli item sedikit dan banyak secara sama. Namun, masing-masing dari baris dapat diturunkan sehingga dihasilkan fungsi kesamaan berbasis kosinus akan membedakan antara set pelanggan *i*. Ini dapat dilakukan dengan meng-skalakan setiap baris untuk satuan panjang

(atau norma lainnya). Efek dari peng-skalaan ini adalah bahwa pelanggan yang telah membeli item yang lebih sedikit akan memberikan kontribusi bobot yang lebih tinggi pada *dot-product* dalam Persamaan 2.4 dari pelanggan yang telah membeli item yang lebih banyak.

2.9.3 Menerapkan Model

Algoritma untuk menerapkan model berbasis item ditunjukkan pada Gambar 2.7. Input untuk algoritma ini adalah model *M*, sebuah vektor *U* berukuran *m* × 1 yang menyimpan barang-barang yang sudah dibeli oleh pengguna yang aktif, dan jumlah item yang akan direkomendasikan (*N*). Pengguna aktif informasi dalam vektor *U* dikodekan dengan mengatur U_i = 1 jika pengguna telah membeli item dan nol jika sebaliknya. Output dari suatu algorithm adalah vektor *x* berukuran *m* × *1* yang entri tidak nol nya sesuai dengan top-*N* item yang dianjurkan. Bobot entri tidak nol ini merupakan ukuran kekuatan rekomendasi dan berbagai rekomendasi dapat diurutkan pada pada kekuatan bobot rekomendasi yang tidak meningkat. Dalam kebanyakan kasus *x* akan memiliki entri tidak nol persis *N*, namun jumlah sebenarnya rekomendasi bisa kurang dari *N* karena bergantung pada nilai *k* yang digunakan untuk membangun *M* dan jumlah barang yang telah dibeli oleh pengguna yang aktif.

$\boldsymbol{x} \leftarrow M\,U$ (1)

for j ← 1 **to** m (2)

do { **if** $(U_i \neq 0)$ **then** $\boldsymbol{x_i} \leftarrow 0$

for j ← 1 **to** m (3)

do { **if** ($\boldsymbol{x_i}$ ≠ among the *N* largest values in $\boldsymbol{x}$) **then** $\boldsymbol{x_i} \leftarrow 0$

return $(\boldsymbol{x})$

Gambar 2.7 Algoritma Penerapan Model

Vektor *x* dihitung dalam tiga langkah. Pertama, vektor *x* dihitung dengan mengalikan *M* dengan *U* (Gambar 2.7 bagian 1). Entri tidak nol dari *x* sesuai dengan union dari k item yang paling mirip untuk setiap item yang sudah dibeli oleh pengguna yang aktif,

dan bahwa bobot entri ini tidak lebih daripada jumlah kesamaan ini. Kedua, entri dari *x* yang sesuai dengan item yang sudah dibeli oleh pengguna yang aktif diset ke nol (perulangan pada Gambar 2.7 bagian 2). Akhirnya, pada langkah ketiga, algoritma mengeset ke nol semua entri dari *x* yang mempunyai nilai lebih kecil dari nilai terbesar *N* pada *x* (perulangan pada Gambar 2.7 bagian 3).

Satu potensi kerugian dengan algoritma pada Gambar 2.7 adalah bahwa kesamaan mentah antara setiap item *j* dan *k* item yang paling mirip dapat secara signifikan berbeda. Yaitu, kedekatan item pada kerapatan yang berbeda. Hal ini terutama berlaku untuk item yang dibeli agak jarang, karena tumpang tindih dengan barang-barang lain yang jarang dibeli dapat menyebabkan nilai-nilai kesamaan yang relatif tinggi. Akibatnya, item ini memiliki pengaruh yang kuat dalam seleksi top-N item, kadang-kadang mengarah kepada rekomendasi yang salah.

2.9.4 Kompleksitas Komputasi

Kompleksitas komputasi algoritma rekomendasi top-N berbasis item bergantung pada jumlah waktu yang diperlukan untuk membangun model *M* (yaitu, setiap item mengidentifikasi *k* lain item yang paling mirip), dan jumlah yang diperlukan untuk menghitung rekomendasi menggunakan model ini. Selama fase pembangunan model, perlu untuk menghitung kesamaan antara setiap item *j* dan item lainnya di *R* dan kemudian pilih item *k* paling mirip.

Batas atas kompleksitas pada langkah ini adalah $\Omega(M^2N)$ yang diperlukan untuk menghitung *m (m - 1)* persamaan, masing-masing n berpotensi memerlukan *n* jumlah operasi. Namun, kompleksitas yang sebenarnya secara signifikan lebih kecil karena yang dihasilkan untuk matriks kesamaan item-item yang sangat jarang (*sparse*). Dalam dataset, item-item untuk kesamaan matriks pada umumnya memiliki tingkat kejarangan lebih dari 99%. Alasan untuk tingkat *sparsity* seperti ini adalah bahwa setiap pelanggan membeli relatif kecil dan jumlah item barang yang mereka beli cenderung sejenis. Akibatnya, dengan menggunakan struktur data yang *sparse* untuk menyimpan *R* dan hanya menghitung kesamaan antara pasangan item yang dibeli dengan setidaknya satu pelanggan dapat secara substansial mengurangi kompleksitas komputasi.

Waktu yang dibutuhkan untuk menghitung rekomendasi top-N bagi pengguna aktif yang telah membeli item *q* diberikan oleh Ω (*kq*)

karena diperlukan mengakses *k* item yang paling mirip untuk setiap salah satu item yang telah pengguna dibeli dan mengidentifikasi keseluruhan item *N* paling mirip.

2.10 Evaluasi

Untuk mengukur kualitas dan keberhasilan dari sistem yang dibangun maka dibutuhkan evaluasi dari sistem yang dibangun. Beberapa parameter standar yang bisa dijadikan acuan pada sebuah sistem *Information Retreival* menurut C. J. Van Rijsbergen pada tahun 1979 antara lain :

1. *time lag*, interval waktu mulai request diterima dan waktu yang dibutuhkan oleh sistem untuk menghasilkan rekomendasi. Persamaan yang menggambarkan perhitungan *time lag* dapat dilihat pada Persamaan 2.5.

   Time lag = Waktu Selesai – Waktu Mulai (2.5)

2. *recall*, merupakan proporsi dari rekomendasi yang relevan dari seluruh rekomendasi yang diberikan sistem sebagai hasil kembalian. Persamaan yang menggambarkan perhitungan *recall* dapat dilihat pada Persamaan 2.6.

$$Recall = \frac{|\{tag\ relevan\} \cap \{tag\ ditemukan\}|}{|\{tag\ ditemukan\}|} \quad (2.6)$$

3. *precission*, merupakan proporsi dari rekomendasi yang benar-benar relevan terhadap pengguna. Persamaan yang menggambarkan perhitungan *precission* dapat dilihat pada Persamaan 2.5.

$$Precission = \frac{|\{tag\ relevan\} \cap \{tag\ ditemukan\}|}{|\{tag\ relevan\}|} \quad (2.7)$$

BAB III
METODE DAN PERANCANGAN SISTEM

3.1 Tahapan pengembangan

Tahapan dalam pengembangan sistem rekomendasi tag ini yaitu:

1. Membagun *engine* rekomendasi
   a. Modul *Content Based*
   b. Modul *Collaborative*
2. Mengumpulkan data
3. Membangun Antarmuka
4. Evaluasi aplikasi yang telah dibangun
5. Revisi aplikasi

3.2 Kebutuhan Sistem

Dalam membangun sistem rekomendasi tag ini dibutuhkan perangkat keras dan perangkat lunak tertentu.

3.2.1 Perangkat keras

Kebutuhan perangkat keras yang digunakan oleh peneliti dalam melakukan penelitian ini adalah dua buah komputer dengan spesifikasi minimum sebagai berikut :

Penggunaan : Pengembangan
Jenis : Notebook /Desktop
Prosesor : i386 CPU dengan kecepatan minimal 1 GHz
Memory : 2 GB
Hardisk : 80 GB
Network : WiFi atau Ethernet

Penggunaan : Server Uji
Jenis : Server
Prosesor : i386/x64 CPU *multicore* @ 2.00GHz
Memory : 2 GB
Hardisk : 80 GB
Network : Ethernet

3.2.2 Perangkat lunak

Perangkat lunak yang digunakan terbagi dalam dua kategori, yaitu perangkat lunak yang digunakan pada server dan perangkat lunak pada client.

Pada server digunakan Sistem Operasi Linux, dengan aplikasi tambahan yaitu webserver Apache yang dilengkapi dengan PHP *Engine* dan server basis data Mysql, untuk penjadwalan digunakan Crontab.

Pada sisi *client* sistem operasi yang digunakan bebas, dan aplikasi tambahan yaitu browser web.

3.3 Data Penelitian

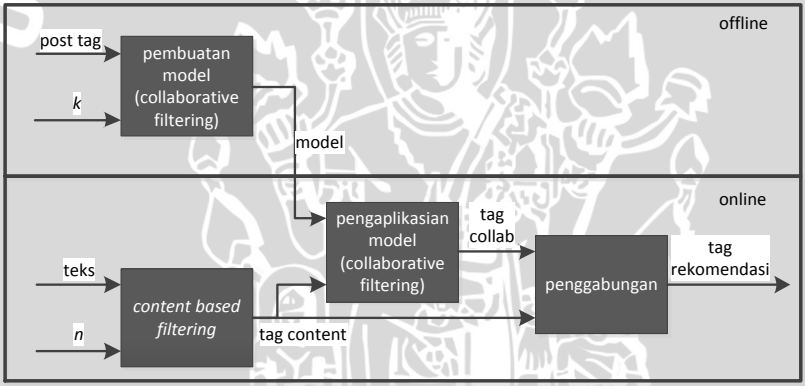
Data penelitian yang digunakan bersumber dari blog-blog yang di-*hosting* pada Wordpress.com melalui layanan Firehose (http://en.wordpress.com/firehose). Firehose merupakan layanan dari Wordpress.com untuk menyediakan *stream* konten yang sifatnya *real-time* yang berisi post-post baru yang mana bisa dimanfaatkan oleh aplikasi pihak ketiga seperti *search-engine* dan provider *market intelegence*.

Stream konten didapatkan dengan melakukan rekues HTTP GET dengan menggunakan aplikasi wget di server uji pada alamat http://im.wordpress.com:8008/firehose.xml?type=text/plain. Format *stream* post tersebut adalah xml yang kemudian disimpan pada file teks untuk pengolahan lebih lanjut. Setelah mendapatkan data cukup banyak, file xml yang telah didapat kemudian diparsing untuk dimasukkan ke dalam basis data. Data-data yang tersedia dalam file xml tersebut antara lain: judul post, isi, kategori (tag), blog, bahasa yang digunakan. Post-post yang didapatkan difilter hanya post-post yang menggunakan Bahasa Inggris saja dan membuang kategori default '*Uncategories*' dan kategori-kategori lain yang sekiranya tidak memiliki arti.

3.4 Perancangan *Engine* Rekomendasi

Engine rekomendasi terdiri dari dua proses yang berjalan terpisah yaitu proses *offline* dimana pembuatan model akan

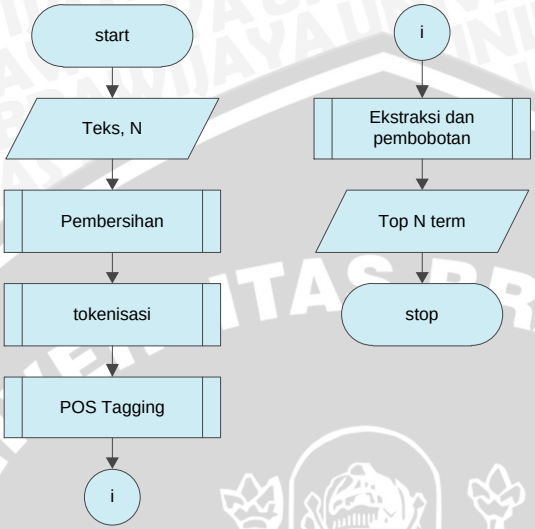
dilakukan pada modul *Collaborative* dengan masukan asosiasi post dan tag pada basisdata dan parameter *k* jumlah *similarity* yang akan disimpan pada model keluaran dan proses *online* yaitu proses yang berjalan *realtime* saat rekomendasi akan diberikan kepada pengguna. Proses *online* terdiri dari tiga bagian yaitu modul *Content Based Filtering* yang akan mengolah teks berikut parameter *n* jumlah tag yang ingin direkomendasikan. Hasil keluaran dari modul ini kemudian menjadi masukan pada bagian pengaplikasian model dari modul *Collaborative filtering* berikut model yang telah dibentuk sebelumnya. Keluaran pada proses ini yaitu tag-tag *collaborative*. Tag-tag keluaran dari modul *content* dan modul *collaborative* kemudian memasuki modul penggabungan untuk digabung dan menjadi tag rekomendasi akhir. Alur dari proses pada *engine* rekomendasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Proses *Engine* Rekomendasi

3.4.1 Modul *Content Based Filtering*

Modul ini berfungsi untuk menghasilkan tag rekomendasi awal dengan mengolah teks secara langsung. Pada modul ini masukan berupa teks dari posting blog dan jumlah N tag teratas yang akan menjadi kembalian yang mana akan menghasilkan tag-tag awal yang nantinya akan digunakan dalam proses pada Modul *Collaborative Filtering*. Tahapan dari proses pada Modul *Content Based Filtering* ini dapat dilihat pada Gambar 3.2.

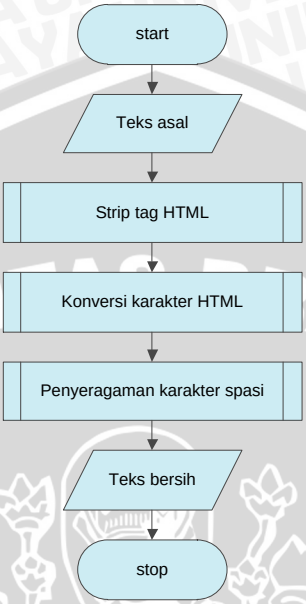


Gambar 3.2 Diagram Alir Modul *Content Based Filtering*

Pada modul ini terbagi menjadi empat tahapan, yaitu:

1. Tahap Pembersihan

   Tahap pembersihan dalam proses ektraksi kandidat term dari teks merupakan langkah tambahan yang dilakukan karena teks masukan masih berisi tag-tag dan karakter HTML. Karakter – karakter dan tag-tag HTML ini bisa menggangu proses POS *Tagging* pada tahap berikutnya. Proses pembersihan diawali dengan proses pembuangan tag-tag HTML dengan menggunakan fungsi *strip_html_tags* yang dikembangkan oleh David R. Nadeau pada tahun 2008. Setelah teks tidak memuat tag-tag HTML kemudian karakter-karakter entitas HTML seperti * *, *&quote;* dan lainnya dikonversi menjadi karakter normal dengan menggunakan fungsi standar PHP *html_entity_decode*. Proses terakhir dari fungsi pembersihan ini yaitu penyeragaman karakter spasi. Semua karakter spasi kosong (*whitespace*) seperti tab, enter, ataupun spasi berulang diseragamkan menjadi sebuah karakter spasi. Proses ini berujuan untuk memudahkan proses tokenisasi yang akan dilakukan berikutnya. Sehingga saat tokenisasi berlangsung teks tinggal dipecah pada setiap karakter spasi tunggal. Gambaran proses pembersihan dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram Alir Tahap Pembersihan

2. Tahap Tokenisasi

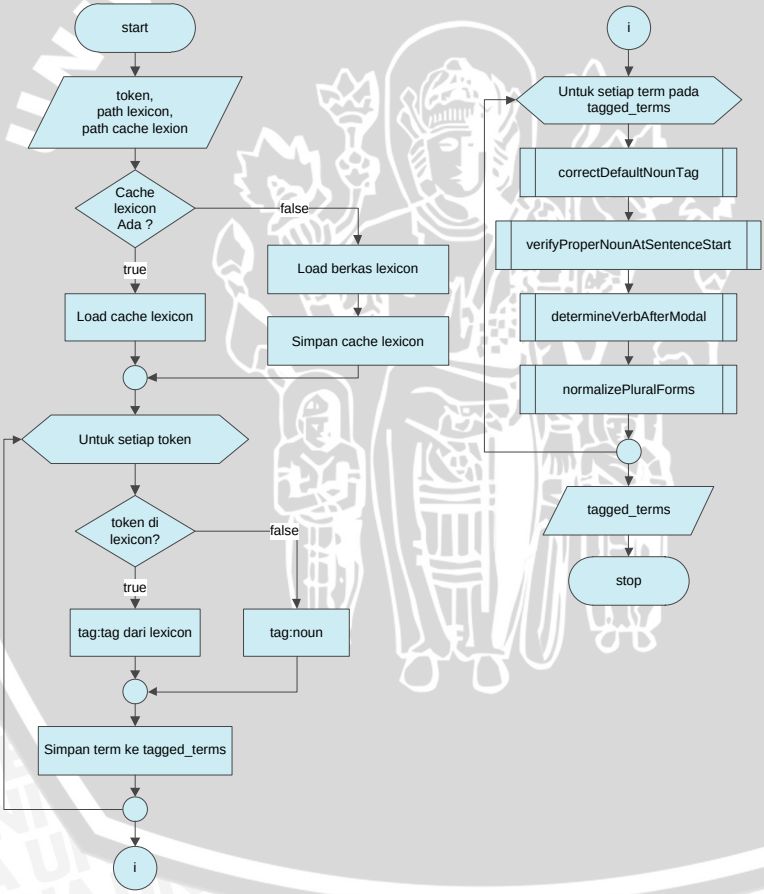
Tahap tokenisasi dilakukan dengan menggunakan algoritma tokenisasi yang telah dijelaskan pada Gambar 2.3. Teks dipecah pada setiap karakter spasi tunggalnya menjadi token-token. Token-token hasil tokenisasi ini kemudian akan menjadi parameter masukan untuk proses POS *Tagging*.

3. Tahap *POS Tagging*

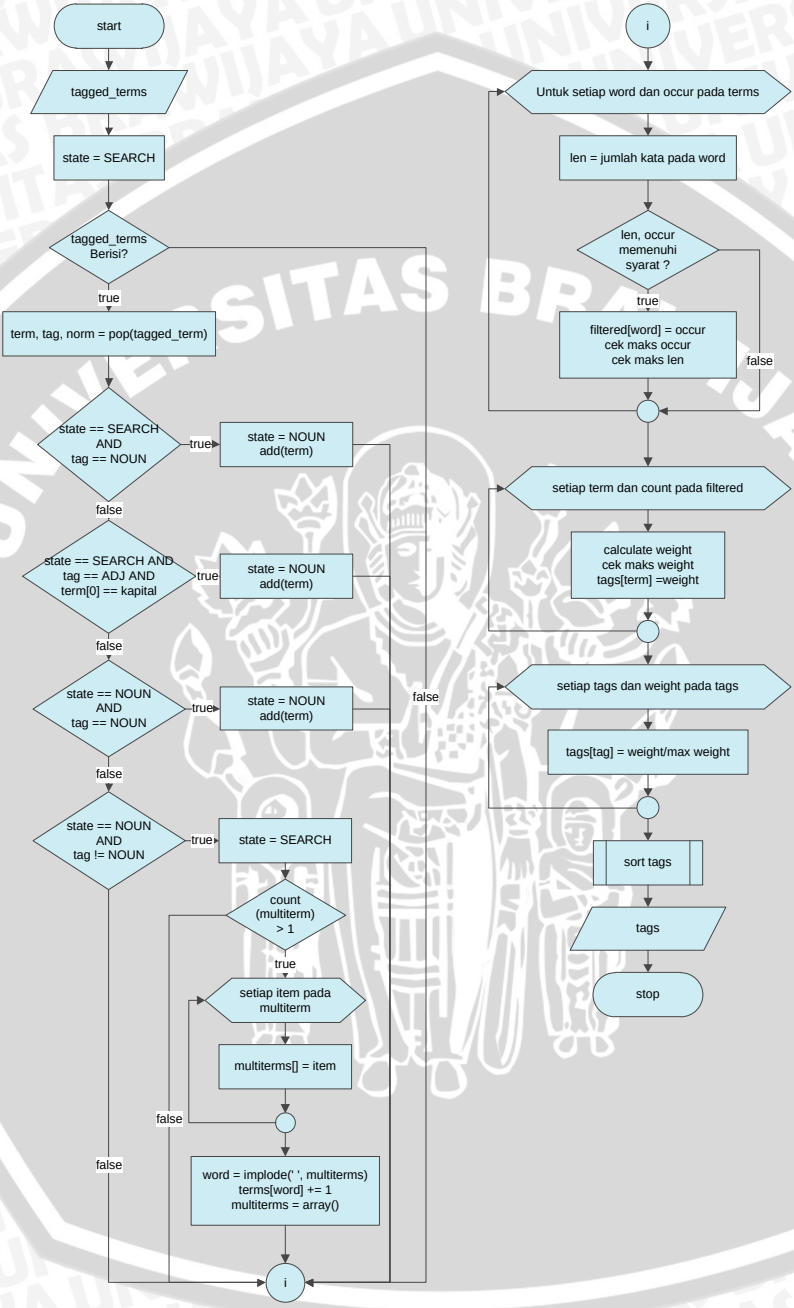
Pada tahap ini proses dilakukan menyerupai algoritma pada Gambar 2.4. Beberapa modifikasi dilakukan untuk meningkatkan performa seperti pengaplikasian *cache lexicon*. Sebelum proses tagging berlangsung fungsi mengecek keberadaan *cache*. Jika belum ada maka dilakukan proses memuat berkas, membacanya baris-perbaris, memecah barisnya pada tiap karakter spasi dan menyimpannya ke dalam larik asosiatif serta pada berkas *cache* dalam format larik ter-*serialize* dengan menggunakan fungsi PHP *serialize*. Fungsi *serialize* ini berfungsi dalam menyimpan struktur data dalam bentuk string teks tanpa kehilangan tipe data dan struktur aslinya. Format ini juga dapat dimuat dalam waktu yang lebih singkat daripada format *lexicon* asli yang masih membutuhkan

proses parsing yang lebih kompleks. Jika berkas *cache* ditemukan maka berkas akan dibaca menjadi string kemudian dikonversi kembali menjadi larik asosiatif dengan fungsi *unserialize*. Fungsi *unserialize* ini merupakan kebalikan dari fungsi *serialize*.

Proses berikutnya yaitu mencari setiap token ke dalam larik asosiatif *lexicon*. Jika ditemukan, POS tag yang akan disimpan adalan POS tag dari *lexicon* yang bersesuaian sedangkan jika tidak ditemukan maka token diasumsikan sebagai kata benda. Token-token ini kemudian akan dinormalisasi strukturnya sesuai dengan algoritma aslinya. Gambaran proses POS *Tagging* dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram *POS Tagging*



Gambar 3.5 Tahap Ektraksi dan Pembobotan Term

4. Tahap Ektraksi dan Pembobotan Term

Pada tahap ini setelah melakukan proses ektraksi dengan algoritma ekstraksi term pada Gambar 2.5 kemudian dilanjutkan dengan proses pembobotan. Pada proses ektraksi sendiri ditambahkan pembersihan term dari karakter-karakter pungtuasi seperti '!@#$%^*():;' sehingga term yang dihasilkan lebih bersih dan proses perhitungan frekuensi tiap term lebih optimal.

Proses berikutnya yaitu menghitung frekuensi term dan mengumpulkan deretan kata-kata benda menjadi sebuah kata majemuk. Setelah semua kata tunggal dan kata majemuk ditemukan berikut frekuensinya maka proses berikutnya yaitu pembobotan term.

Proses pembobotan dilakukan pada term-term yang telah memuhi kriteria frekuensi minimal. Skema pembobotan yang bisa digunakan antara lain:

- Pembobotan berdasarkan jumlah kata tiap term dibagi dengan jumlah kata maksimal yang ditemukan. Seperti yang dijabarkan pada Persamaan 2.1.
- Pembobotan berdasarkan frekuensi kata dibagi frekuensi maksimal kata ditemukan. Seperti yang dijabarkan pada Persamaan 2.2.
- Pembobotan berdasarkan gabungan antara panjang kata yang telah ternormalisasi dan frekuensi kata yang juga telah ternormaliasi. Seperti yang dijabarkan pada Persamaan 2.3.

Dari ketiga skema tersebut akan diuji mana yang bisa menghasilkan *precission* dan *recall* yang lebih baik.

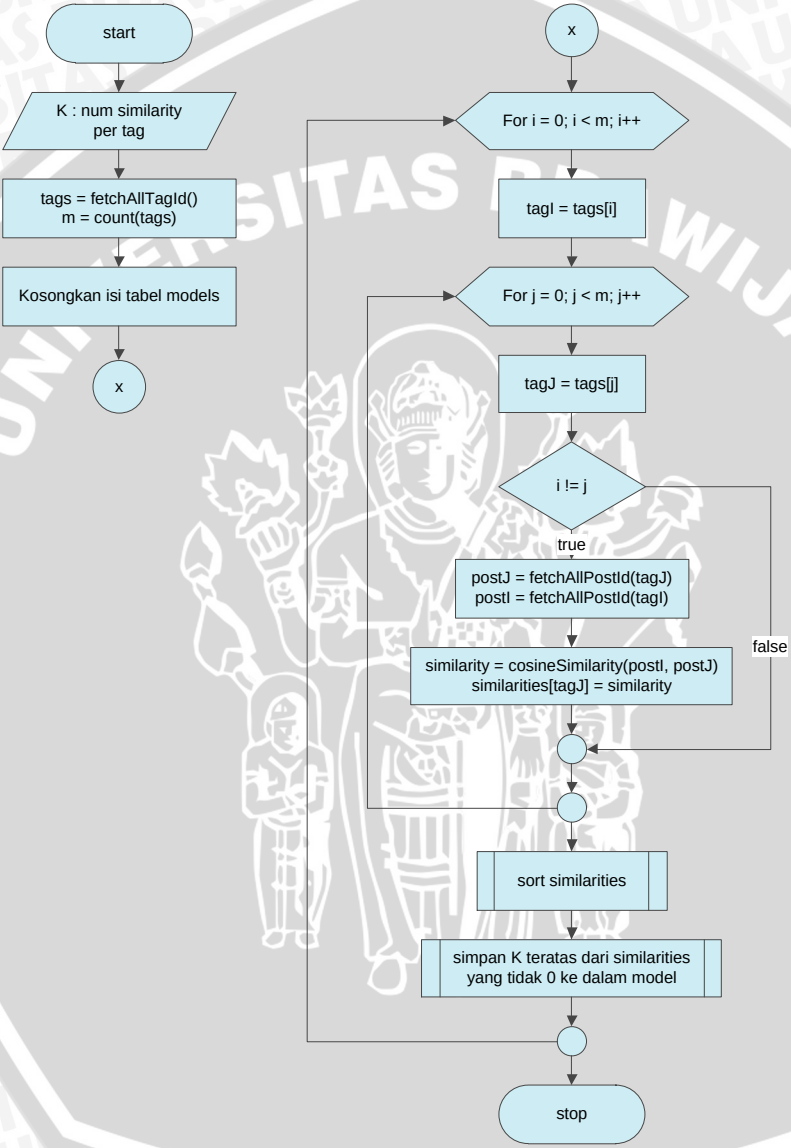
Setelah semua bobot telah didapatkan kemudian term-term diurutkan berdasarkan bobotnya mulai dari bobot yang terbesar ke yang kecil. Larik term-term yang sudah terurut ini akan menjadi keluaran untuk proses berikutnya.

Proses ekstraksi dan pembobotan term dapat dilihat pada Gambar 3.5.

3.4.2 Modul *Collaborative Filtering*

Modul *Collaborative* ini merupakan modul yang digunakan untuk menghasilkan tag rekomendasi secara kolaboratif dari tag-tag lain yang telah ada pada basis data yang memiliki karakteristik yang menyerupai tag-tag yang telah ditemukan pada modul *Content Based*

Filtering. Modul ini dibangun berdasarkan algoritma *Item-based Top-N Recommendation*. Modul ini dibagi menjadi dua tahap, yaitu:



Gambar 3.6 Tahap Pembangunan Model

1. Tahap Pembangunan Model

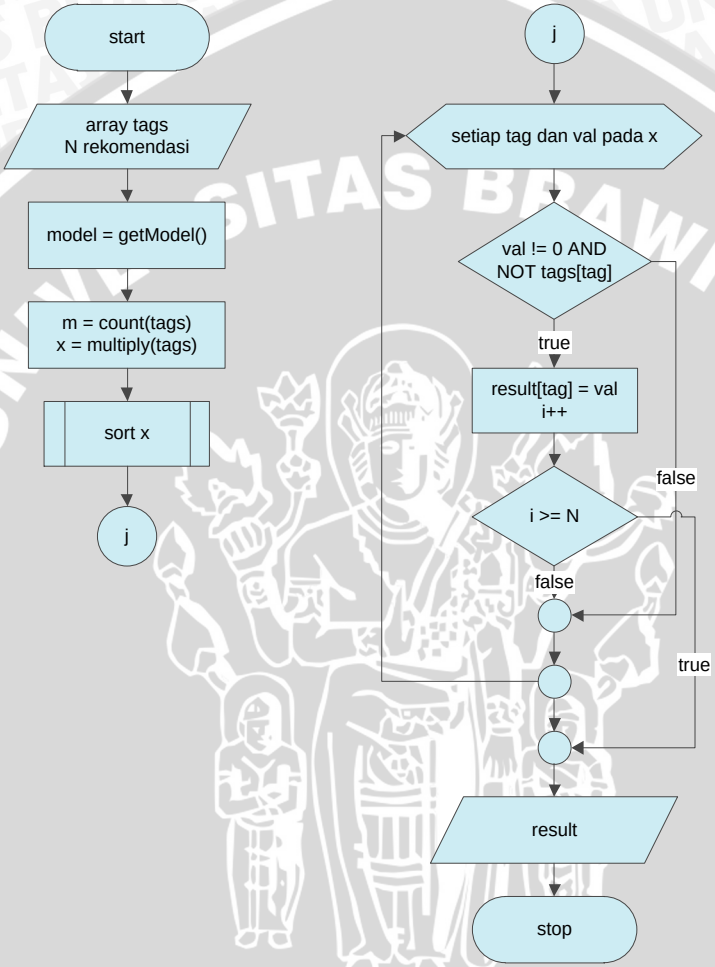
Pada tahap ini dilakukan pembentukan model yang akan menggambarkan rangkuman dari kemiripan antar tag-tag dalam basis data. Model dibentuk dengan melakukan iterasi pada semua tag yang ada dalam basis data dan menghitung kemiripan tag-tag tersebut satu sama lain. Tag-tag yang tidak memiliki post-post serupa tidak dihitung nilai kemiripannya karena dianggap tidak relevan. Jumlah kemiripan yang disimpan untuk setiap tagnya hanya sejumlah tertentu yang memiliki nilai kemiripan tertinggi untuk menjaga ukuran model yang berdampak pada skalabilitas dan performa saat perhitungan rekomendasi akhir. Karena tahapan ini membutuhkan waktu yang cukup lama, terutama pada data yang berjumlah sangat besar, tahapan ini dijalankan secara terjadwal dengan interval tertentu pada latar belakang.

Masukan dari fungsi pembuatan model yaitu faktor *k*, yaitu jumlah kemiripan yang akan disimpan ke model untuk tiap tagnya. Pembuatan model diawali dengan mengambil seluruh *tag_id* dari basis data dan menghitung jumlahnya pada *m*, kemudian tabel *models* pada basis data dikosongkan. Langkah berikutnya yaitu melakukan iterasi pada tiap pasangan tag yang mungkin. Dalam iterasi tersebut kemiripan dihitung dengan menggunakan fungsi *cosineSimilarity*. Kemiripan per tag tersebut kemudian disimpan dalam larik yang mana setelah iterasi diurutkan untuk diambil *k* teratas. Kemiripan *k* teratas yang tidak nol kemudian dimasukkan ke dalam tabel *models* pada basis data untuk perhitungan rekomendasi akhir. Diagram alur dari langkah-langkah pembuatan model dapat dilihat pada Gambar 3.6.

2. Tahap Pengaplikasian Model

Dari model yang telah terbentuk dapat dihitung tag rekomendasi dengan mengalikan model dengan vektor tag yang didapatkan dari modul *Content Based Filtering*. Elemen model yang digunakan yaitu hanya nilai-nilai kemiripan dari tag-tag yang berasosiasi pada vektor tag baru tersebut. Hasil perkalian tersebut kemudian disimpan dalam larik *x* yang mana kemudian di urutkan berdasarkan nilai kemiripan terbesar hingga kecil. Langkah terakhir yaitu melakukan iterasi pada larik *x* untuk memfilter nilai kemiripan yang tidak nol dan tag tidak terdapat pada vektor asal. Hanya sejumlah *N* teratas yang kemudian menjadi keluaran dari fungsi ini.

Diagram alur proses pada tahapan ini digambarkan pada Gambar 3.7.



Gambar 3.7 Tahap Pengaplikasian Model

3.5 Perancangan Struktur Data

Untuk menyimpan data penelitian, model yang dibuat, tag-tag yang dihasilkan, data pengguna dan data-data lain yang mendukung proses penelitian dirancang relasi basis data yang diimplementasikan pada server basis data Mysql. Tabel-tabel yang digunakan yaitu:

1. Tabel *posts*

Tabel *posts* berisi data posting blog, pada tabel ini tersimpan teks dari posting blog berikut judulnya dan waktu pembuatan. Data lengkap mengenai tabel *posts* ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tabel *posts*

Field	Jenis	Kosong	Default	Primary key
post_id	**int(10)**	**Tidak**		**Ya**
judul	varchar(255)	Ya	*NULL*	Tidak
isi	text	Ya	*NULL*	Tidak
waktu	timestamp	Ya	*NULL*	Tidak

2. Tabel *tags*

Tabel ini berisi data tag dari semua posting, untuk setiap data tag dilengkapi dengan frekuensi ditemukannya dalam seluruh post. Gambaran lengkap dari tabel tags dapat dilihat pada Tabel 3.2.

Tabel 3.2 Tabel *tags*

Field	Jenis	Kosong	Default	Primary key
tag_id	**int(10)**	**Tidak**		**Ya**
tag	varchar(255)	Ya	*NULL*	Tidak
count	int(10)	Ya	*NULL*	Tidak

3. Tabel *post_tags*

Pada tabel ini berisi hubungan antara post dan tag. Dimana hubungannya merupakan *one to many* yaitu untuk setiap post dapat memiliki nol atau lebih tag. Pada Tabel 3.3 dapat dilihat detail mengenai tabel *post_tags*.

Tabel 3.3 Tabel *post_tags*

Field	Jenis	Kosong	Default	Primary key
post_id	**int(10)**	**Tidak**		**Ya**
tag_id	**int(10)**	**Tidak**		**Ya**
is_recomendation	tinyint(3)	Ya	0	Tidak

4. Tabel *models*

Tabel ini berisi model yang dibangun dari modul *Collaborative* pada engine rekomendasi. Pada tabel ini disimpan *similarity* dari antar tag yang nantinya akan digunakan untuk menghitung rekomendasi untuk pengguna. Struktur dari tabel *models* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Tabel *models*

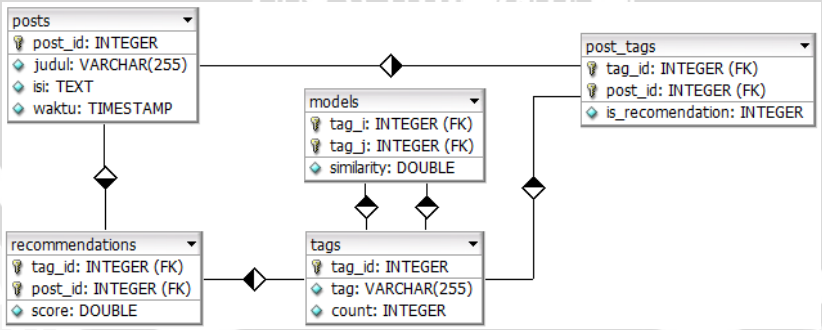
Field	Jenis	Kosong	Default	Primary key
tag_i	**int(10)**	**Tidak**		**Ya**
tag_j	**int(10)**	**Tidak**		**Ya**
similarity	double	Ya	*NULL*	Tidak

5. Tabel *recomendations*

Pada tabel ini rekomendasi yang telah dihasilkan disimpan. Tujuan dari penyimpanan ini untuk proses evaluasi kualitas rekomendasi yang diberikan. Struktur tabel *recommendations* dapat dilihat di Tabel 3.5.

Tabel 3.5 Tabel *recommendations*

Field	Jenis	Kosong	Default	Primary key
post_id	**int(10)**	**Tidak**		**Ya**
tag_id	**int(10)**	**Tidak**		**Ya**
score	double	Ya	*NULL*	Tidak



Gambar 3.8 Hubungan Antar Tabel

Hubungan antar relasi pada basis data dapat dilihat pada Gambar 3.8. Semua hubungan antar relasi tersebut merupakan hubungan "memiliki".

Selain basis data juga diperlukan penyimpanan dalam bentuk berkas. Data sumber definisi POS tag (*lexicon*) disimpan dalam pada berkas yang mana saat aplikasi berjalan akan dimuat dan disimpan dalam format larik asosiatif (*associative array)* atau *hashtable* di memori. Contoh data lain yang perlu disimpan dalam format ini antara lain token-token yang telah diproses, tag-tag yang telah dibobot dan diurutkan, hasil kembalian dari kueri pada basis data dan *lexicon*. Format larik lain yang dimanfaatkan yaitu larik numerik. Larik ini dimanfaatkan dalam penyimpanan token yang belum diproses. Teks-teks yang akan digunakan, baik teks asal, maupun teks yang telah diproses disimpan dalam format *string*.

3.6 Perancangan Hasil Penelitian

Pengujian yang akan dilakukan bertujuan untuk mengukur tingkat akurasi dari sistem rekomendasi yang dibuat. Parameter akurasi yang digunakan yaitu *precission* dan *recall. Precission* mengacu pada seberapa tepat hasil rekomendasi dan *recall* mengacu pada seberapa lengkap dari hasil rekomendasi yang dihasilkan dibandingkan dengan tag-tag yang dipilih oleh pengguna secara manual.

Pengukuran pertama yang dilakukan yaitu menemukan metode pembobotan pada Modul *Content Based Filtering* yang optimal antara waktu rekomendasi dengan *precission* dan *recall* tag yang dihasilkan. Hasil pengujian awal ini disimpan pada format seperti pada Tabel 3.6.

Tabel 3.6 Pembobotan pada modul *Content Based Filtering*

No	Metode	Precission	Recall	Recomendation time
1				
2				

Pengukuran kedua yang dilakukan yaitu menemukan parameter jumlah *similarity* antar item yang disimpan pada model pada modul *Collaborative* atau disebut parameter *k*. Parameter *k* ini

dicari nilai optimal terhadap jumlah data, waktu pembuatan model, *precission* dan *recall* yang dihasilkan serta terhadap waktu rekomendasi. Dengan mencoba beberapa nilai *k* pada sejumlah data tertentu maka akan didapatkan nilai *k* yang optimal. Hasil percobaan parameter *k* ini akan disimpan seperti pada Tabel 3.7.

Tabel 3.7 Parameter *k* pada modul Collaborative

No	N Tag	*k*	Model time	Precission	Recall	Recomendation time
1						
2						

Setelah ditemukan *k* optimal pada modul *Collaborative* maka dilanjutkan dengan menemukan komposisi antara *Content Based* dan *Collaborative*. Pengujian dilakukan untuk mengetahui berapa proporsi optimal antara modul *Content Based* dan modul *Collaborative* terhadap performa dan akurasi, performa digambarkan dengan waktu yang dibutuhkan untuk menghasilkan rekomendasi dan akurasi digambarkan dengan nilai *precission* dan *recall*.Hasil pengujian kemudian akan disimpan pada tabel seperti Tabel 3.8.

Tabel 3.8 Proporsi modul Content dan modul Collaborative

No	% Content	% Collaborative	Precission	Recall	Recom. time
1					
2					

Pengujian yang dilakukan berikutnya yaitu pengujian skalabilitas dari sistem yang dibangun. Sistem harus dapat memberikan hasil rekomendasi dalam waktu yang relatif singkat pada jumlah data yang sedikit ataupun sangat besar. Pengujian diawali dengan tanpa menggunakan data dari basis data dan kemudian ditingkatkan secara progresif hingga semua data pada basis data digunakan. Pada setiap iterasi yang dilakukan diukur waktu pembuatan model dan waktu untuk mengasilkan rekomendasi. Hasil pengujian kemudian disimpan pada tabel dengan struktur seperti Tabel 3.9.

Tabel 3.9 Skalabilitas

No	N Tag	Model time	Recomendation time
1			
2			

3.7 Perancangan Antarmuka

Antarmuka yang digunakan untuk aplikasi ini dibuat menyerupai manajemen konten pada blog. Modul utama pada aplikasi ini yaitu modul pengelolaan posting.

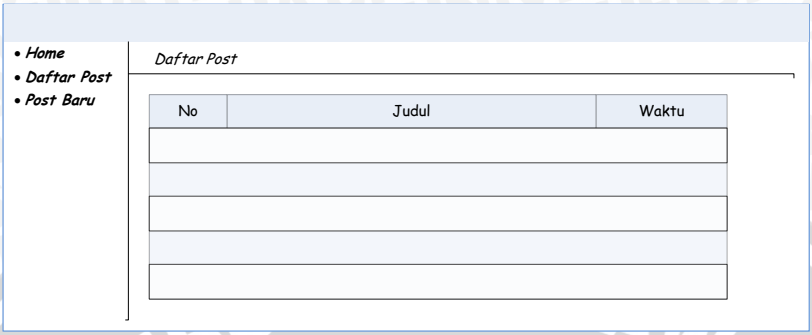
3.7.1 Halaman Beranda

Halaman beranda yaitu halaman yang pertama akan diakses pengguna saat masuk ke dalam aplikasi. Pada halaman ini berisi rangkuman jumlah data-data yang pada basis data. Pada halaman ini juga dilengkapi dengan menu-menu pada lajur kiri seperti Home, Daftar Post dan Post Baru. Tampilan dari halaman beranda dapat dilihat pada Gambar 3.9.

Gambar 3.9 Rancangan Halaman Beranda

3.7.2 Halaman Post

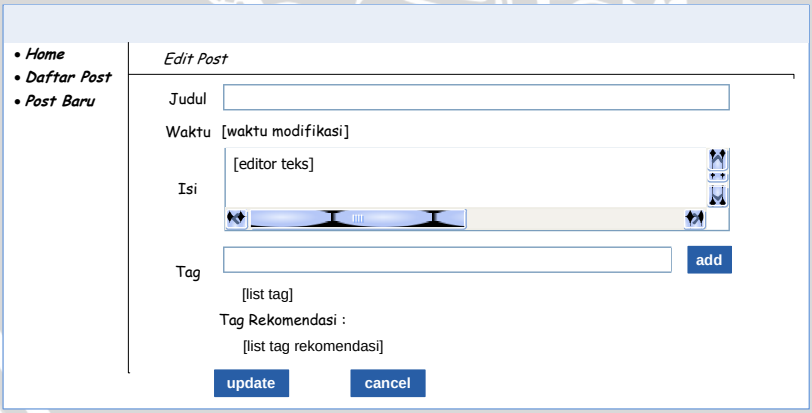
Pada halaman ini diawali dengan daftar posting dari pengguna yang aktif seperti yang ditampilkan pada Gambar 3.10.



Gambar 3.10 Rancangan Halaman Daftar Post

Judul post pada tabel merupakan tautan yang jika di-klik oleh pengguna akan masuk ke halaman edit post seperti yang digambarkan pada Gambar 3.11.

Pada halaman edit post pengguna dapat mengubah isi dari posting dan memberikan atau menghapus tag. Saat memberikan tag pengguna juga dapat menambahkan tag-tag yang dihasilkan oleh *engine* rekomendasi dengan meng-klik tautan 'rekomendasi tag'. Untuk menyimpan perubahan pengguna dapat meng-klik tombol *update* dan tombol batal untuk kembali kehalaman daftar post tanpa menyimpan perubahan.



Gambar 3.11 Rancangan Halaman Edit Posting

3.8 Contoh Perhitungan Manual

Proses perhitungan akan dibagi menjadi tiga tahap utama, yaitu Tahap *Content Based Filtering*, Tahap *Collaborative Filtering* dan Tahap Penggabungan.

3.8.1 Tahap *Content Based Filtering*

Sebagai contoh digunakan teks yang merupakan sebuah potongan dari sebuah posting blog yang bersumber dari website http://funnyhamlet.wordpress.com. Isi dari teks contoh tersebut dapat dilihat pada gambar 3.12

```
As an English teacher, college admission officer, and college
counselor, I've read thousands of college application essays and
actually have looked forward to reading most of them, but there's a
lot of junk out there, even from otherwise good students.
```

Gambar 3.12 Teks Masukkan

Teks sumber tersebut kemudian akan dipecah berdasarkan karakter spasi menjadi token-token untuk proses berikutnya. Daftar token dari teks pada gambar 3.12 dapat dilihat pada gambar 3.13.

```
As|an|English|teacher|,|college|admission|officer|,|and|college|
counselor|,|I|'ve|read|thousands|of|college|application|essays|a
nd|actually|have|looked|forward|to|reading|most|of|them|,|bu
t|there|'s|a|lot|of|junk|out|there|,|even|from|otherwise|good|
students|.
```

Gambar 3.13 Teks Masukkan setelah tokenisasi

Dari gambar 3.13 tampak token-token dipisahkan oleh karakter '|'. Token-token tersebut disimpan dalam larik untuk menjadi masukan dalam proses *Part-of-speech tagging*. Proses pertama dalam *Part-of-Speech tagging* yaitu setiap token akan dicari tag *part-of-speech* nya dalam *lexicon*. Jika tidak ditemukan maka dianggap kata benda. Larik yang didapat setelah tahap pencarian pada *lexicon* ini berakhir dapat dilihat pada tabel 3.10. Setiap item dari larik ini terdiri dari token yang diperoses, POS tag nya dan bentuk normal dari token tersebut. Bentuk normal ini biasanya berupa kata dasar dan semua hurufnya dikonversi menjadi huruf

kecil. Definisi singkatan dari tag-tag POS dapat dilihat definisinya pada Lampiran 1.

Setelah semua token telah diketahui tag *part-of-speech* nya maka proses berikutnya yaitu pengaplikasian normalisasi pada token-token tersebut.

Tabel 3.10 Token-token dengan POS tag nya

token	tag	normal	token	tag	normal
As	IN	as	looked	VBD	looked
an	DT	an	forward	RB	forward
English	NNP	english	to	TO	to
teacher	NN	teacher	reading	NN	reading
,	,	,	most	RBS	most
college	NN	college	of	IN	of
admission	NN	admission	them	PRP	them
officer	NN	officer	,	,	,
,	,	,	but	CC	but
and	CC	and	there	EX	there
college	NN	college	’s	NND	’s
counselor	NN	counselor	a	DT	a
,	,	,	lot	NN	lot
I	PRP	i	of	IN	of
’ve	NND	’ve	junk	NN	junk
read	VB	read	out	IN	out
thousands	NNS	thousands	there	EX	there
of	IN	of	,	,	,
college	NN	college	even	RB	even
application	NN	application	from	IN	from
essays	NNS	essays	otherwise	RB	otherwise
and	CC	and	good	JJ	good
actually	RB	actually	students	NNS	students
have	VBP	have	.	.	.

Contoh pengaplikasian normaliasi pada token-token yang telah ditemukan antara lain, memastikan kembali token yang tidak ditemukan pada *lexicon* merupakan kata benda jamak atau tunggal. Kemudian mencari kata dasar untuk kata benda jamak dan kemudian disimpan pada bentuk normalnya. Hasil larik setelah normalisasi dapat dilihat pada Tabel 3.11.

Tabel 3.11 Token-token dengan POS Tag Ternormalisasi

token	tag	normal	token	tag	normal
As	IN	as	looked	VBD	looked
an	DT	an	forward	RB	forward
English	NNP	english	to	TO	to
teacher	NN	teacher	reading	NN	reading
,	,	,	most	RBS	most
college	NN	college	of	IN	of
admission	NN	admission	them	PRP	them
officer	NN	officer	,	,	,
,	,	,	but	CC	but
and	CC	and	there	EX	there
college	NN	college	’s	NNS	’
counselor	NN	counselor	a	DT	a
,	,	,	lot	NN	lot
I	PRP	i	of	IN	of
’ve	NN	’ve	junk	NN	junk
read	VB	read	out	IN	out
thousands	NNS	thousand	there	EX	there
of	IN	of	,	,	,
college	NN	college	even	RB	even
application	NN	application	from	IN	from
essays	NNS	essay	otherwise	RB	otherwise
and	CC	and	good	JJ	good
actually	RB	actually	students	NNS	student
have	VBP	have	.	.	.

Setelah semua token dinormalisasi tahap berikutnya yaitu ektraksi term. Ektraksi term dimulai dengan mengelompokkan semua token-token yang merupakan kata benda dan menghitung frekuensi kemunculannya dalam teks. Dari proses ini dapat dipisahkan antara kata majemuk dan kata tunggal. Kemudian term-term difilter berdasarkan frekuensi yang telah ditentukan sebelumnya. Misal untuk kata benda tunggal minimal frekuensi yang akan digunakan adalah dua dan untuk kata benda majemuk minimal dua. Hasil ektraksi dapat dilihat pada Tabel 3.12.

Tabel 3.12 Ektraksi Term

term	frekuensi	jumlah kata
english teacher	1	2
english	1	1
teacher	1	1
college admission officer	1	3
admission	1	1
officer	1	1
college counselor	1	2
counselor	1	1
ive	1	1
thousand	1	1
college application essays	1	3
college	3	1
application	1	1
essay	1	1
reading	1	1
there	1	1
lot	1	1
junk	1	1
student	1	1

Pada Tabel 3.12 hasil ekstraksi akhir adalah term-term yang diberi penanda abu-abu, yaitu term-term yang memenuhi kualifikasi yang telah ditentukan. Term-term ini kemudian dibobot sesuai jenis

dan frekuensinya. Misal skema pembobotan yang digunakan yaitu Pembobotan berdasarkan gabungan antara panjang kata yang telah ternormalisasi dan frekuensi kata yang juga telah ternormaliasi seperti pada Persamaan 2.3 maka hasil akhir yang didapat yaitu seperti pada Tabel 3.13.

Tabel 3.13 Pembobotan dan Pengurutan Term

term	bobot			
	frek	jml kata	perhitungan	total
college	3	1	$\frac{\frac{3}{3}+\frac{1}{1}}{2}$	1

Misal hanya tiga term yang diambil maka tiga term teratas yang akan menjadi hasil kembalian yaitu: *'college'*.

3.8.2 Tahap *Collaborative Filtering*

Langkah pertama dari Tahap *Collaborative Filtering* yaitu pembuatan model. Masukan proses ini berasal dari basis data yaitu matriks post-tag seperti pada Tabel 3.14.

Tabel 3.14 Matriks Post-Tag

tag/post	1	2	3	4
college	1	1	0	1
academic	0	1	1	1
faculty	1	0	0	1
computer	1	0	0	1

Dari matriks post-tag tersebut kemudian dicari *similarity* dari tag-tag yang ada. Rumus *similarity* yang digunakan yaitu *cosine based similarity* seperti pada Persamaan 2.4. Proses perhitungan similarity antar tag dapat dilihat pada Tabel 3.15. Untuk mendapatkan nilai *similarity* yang relevan setiap pasangan tag minimal harus memiliki dua post yang sama.

Tabel 3.15 Perhitungan *similarity* antar tag

tag_i	tag_j	Perhitungan		sim
college	*academic*	$\frac{[1\ 1\ 0\ 1] \cdot [0\ 1\ 1\ 1]}{\|\|1\ 1\ 0\ 1\|\|_2 * \|\|0\ 1\ 1\ 1\|\|_2}$	$\frac{2}{1.73 * 1.73}$	0.67
college	*faculty*	$\frac{[1\ 1\ 0\ 1] \cdot [1\ 0\ 0\ 1]}{\|\|1\ 1\ 0\ 1\|\|_2 * \|\|1\ 0\ 0\ 1\|\|_2}$	$\frac{2}{1.73 * 1.41}$	0.82
college	*computer*	$\frac{[1\ 1\ 0\ 1] \cdot [1\ 0\ 0\ 1]}{\|\|1\ 1\ 0\ 1\|\|_2 * \|\|1\ 0\ 0\ 1\|\|_2}$	$\frac{2}{1.73 * 1.41}$	0.82
academic	*faculty*	Tidak memenuhi syarat	$\frac{1}{1.73 * 1.41}$	0
academic	*computer*	Tidak memenuhi syarat	$\frac{1}{1.73 * 1.41}$	0
faculty	*computer*	$\frac{[1\ 0\ 0\ 1] \cdot [1\ 0\ 0\ 1]}{\|\|1\ 0\ 0\ 1\|\|_2 * \|\|1\ 0\ 0\ 1\|\|_2}$	$\frac{2}{1.41 * 1.41}$	1

Setelah perhitungan selesai dilakukan dapat dilihat bentuk matriks hubungan *similarity* antar tag seperti pada Tabel 3.16.

Tabel 3.16 *Similarity* antar Tag

Tag	*college*	*academic*	*faculty*	*computer*
college	0	0.67	0.82	0.82
academic	0.67	0	0	0
faculty	0.82	0	0	1
computer	0.82	0	1	0

Hasil matriks pada Tabel 3.16 kemudian disimpan ke dalam basis data sebagai model, namun untuk setiap tag hanya *similarity* sejumlah *k* tertentu yang disimpan untuk menjaga performa saat kalkulasi rekomendasi akhir. Misal hanya dua *similarity* tertinggi

yang disimpan maka bentuk akhir dari matriks model dapat dilihat pada Tabel 3.17.

Tabel 3.17 Model

Tag	*college*	*academic*	*faculty*	*computer*
college	0	0	0.82	0.82
academic	0.67	0	0	0
faculty	0.82	0	0	1
computer	0.82	0	1	0

Langkah berikutnya yaitu membentuk hasil dari Tahap *Content based Filtering* menjadi vektor tag (U) yang akan digunakan sebagai masukan dalam proses perhitungan rekomendasi dengan memanfaatkan model yang telah jadi. Bentuk vektor tag dapat dilihat pada Tabel 3.18.

Tabel 3.18 Vektor tag dari *Content Based Filtering* (U)

Tag	U
college	1
academic	0
faculty	0
computer	0

Setelah vektor tag dari Tahap *Content Based Filtering* (U) jadi kemudian untuk membentuk rekomendasi diawali dengan mengalikan model pada Tabel 3.17 dengan vektor tag pada tabel 3.18. misal hasil perkalian tersebut disebut vektor *x* dan dapat dilihat pada Tabel 3.19.

Tabel 3.19 Hasil Kali Model (M) dan Vektor tag Baru (U)

Tag	x
college	0
academic	0
faculty	0.82
computer	0.82

Tahap terakhir pada tahap *Collaborative Filtering* ini yaitu memfilter sejumlah N bobot teratas. Misal yang akan digunakan sebagai hasil rekomendasi hanya dua tag tertinggi maka hasil rekomendasi dari tahap ini yaitu dapat dilihat pada Tabel 3.20.

Tabel 3.20 Hasil Rekomendasi *Collaborative Filtering*

Tag Rekomendasi	Bobot
faculty	0.82
computer	0.82

3.8.3 Tahap Penggabungan

Setelah Tahap *Content Based Filtering* dan Tahap *Collaborative Filtering* selesai maka proses terakhir yaitu menggabungkan hasil rekomendasi kedua tahap tersebut menjadi hasil rekomendasi akhir. Tag-tag rekomendasi akhir yaitu: '*college'*, '*faculty*', '*computer*'.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi pada skripsi ini terbagi menjadi dua bagian, yaitu lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan saat implementasi aplikasi terbagi menjadi dua didasarkan atas penggunaannya. Perangkat keras tersebut antara lain:

1. Perangkat Pengembangan
   - Jenis : Notebook /Laptop
   - Prosesor : Intel Pentium Dual CPU T2390 @ 1,86GHz
   - Memory : 2 GB
   - Hardisk : 120 GB
   - Network : WiFi dan Ethernet

2. Perangkat Server Uji
   - Prosesor : Intel Xeon E5504 Quad Core @ 2.00GHz
   - Memory : 3 GB
   - Hardisk : 500 GB
   - Network : Ethernet

Perangkat pengembangan yang digunakan yaitu komputer dengan kemampuan komputasi standar yang mana cukup untuk menjalankan beberapa aplikasi pengembangan seperti server web, server basis data dan editor PHP secara bersamaan. Sedangkan untuk server uji digunakan server dengan kemampuan komputasi diatas rata-rata sehingga proses pengujian tidak berlangsung terlalu lama terutama saat proses pembangunan model dengan jumlah data yang sangat besar.

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan juga terbagi menjadi dua macam menyesuaikan dengan jenis perangkat keras yang digunakan.

1. Perangkat Lunak Pengembangan
   Perangkat lunak yang digunakan saat pengembangan antara lain:
   - Sistem Operasi Windows
   - XAMPP (apache, mysql dan php)
   - Editor PHP
   - Perangkat administrasi Basis data
   - Browser Mozilla Firefox
   - SSH *client* Putty
   - SFTP *client* Winscp

2. Perangkat Lunak pada Server Uji
   Perangkat lunak yang digunakan pada server uji antara lain:
   - Sistem Operasi Linux CentOS 5.0
   - Server Web Apache Httpd 2.2.3
   - Server Basis Data Mysql 5.0.77
   - PHP 5.1.6

4.2 Persiapan Data

Data yang digunakan dalam penelitian ini bersumber dari kumpulan post-post pada wordpress.com yang dikumpulkan antara tanggal 19 Oktober 2010 hingga 17 November 2010 melalui layanan *stream* data *Firehose* dari wordpress.com melalui alamat URL http://im.wordpress.com:8008/firehose.xml?type=text/plain. Data kemudian dibagi menjadi dua yaitu data *training* dan data uji. Data *training* yang digunakan sejumlah 81.233 post, 155.058 tag dan 388.918 asosiasi antara post dan tag. Data *training* ini kemudian diturunkan menjadi dua basis data terpisah yang hanya memuat masing-masing 1.000 tag dan 10.000 tag berikut post yang berasosiasi untuk tujuan pengujian performa aplikasi pada ukuran data yang berbeda. Sedangkan data uji terdiri dari 5.794 post, 17.752 tag dan 26.970 asosiasi antara post dan tag. Data uji ini digunakan sebagai acuan saat dilakukan pengujian akurasi aplikasi. Pemisahan antara data training dan data uji bertujuan agar hasil pengujian bisa lebih objektif.

4.3 Implementasi Program

Implementasi program terbagi menjadi dua modul utama yaitu modul *Content Based* dan Modul *Collaborative*.

4.3.1 Implementasi Modul *Content Based*

Implementasi Modul Content Based terbagi menjadi lima tahapan. Semua tahapan tersebut terdapat kelas `Tags_Content`

Tahap-tahap tersebut secara umum dapat dilihat pada fungsi *getTags* seperti yang ditampilkan pada Gambar 4.1.

```
21  public function getTags($N = -1)
22  {
23      // cleanup
24      $this->cleanup();
25
26      if (!$this->text)
27          return false;
28
29      // tokenize teks
30      $token = $this->tokenize($this->text);
31
32      // POS Tagging
33      $taggedTerms = $this->tag($token);
34
35      // extract tag
36      $result = $this->extracts($taggedTerms);
37
38      if (-1 == $N)
39          return $result;
40      else {
41          return @array_slice($result, 0, $N);
42      }
43  }
```

Gambar 4.1 Kode Sumber Fungsi *getTags*

Tahapan-tahapan yang dilakukan antara lain yaitu pembersihan dengan fungsi *cleanup*, tokenisasi dengan fungsi *tokenize*, POS *Tagging* dengan fungsi tag, ektraksi dan pembobotan dengan fungsi *extracts* dan terakhir adalah memotong N teratas sebagai hasil rekomendasi tag. Rincian tiap-tiap tahap adalah sebagai berikut :

```
protected function cleanup()
{

   if (strip_tags($this->text) != $this->text)
      $this->text = Tags_Utils::strip_html_tags($this->text);

   $this->text = html_entity_decode($this->text, ENT_QUOTES, "UTF-8");
   $this->text = iconv('UTF-8', 'ISO-8859-1//TRANSLIT//IGNORE', $this->text);
   $this->text = preg_replace('/\[caption.*\[\/caption\]/', '', $this->text);
   $this->text = preg_replace('/\[picapp.*\]/', '', $this->text);
   $this->text = preg_replace('/´/', '\'', trim($this->text));
   $this->text = preg_replace('/\s+/', ' ', trim($this->text));
}
```

Gambar 4.2 Kode Sumber Fungsi *cleanup*

```
public static function strip_html_tags($text)
{
   $text = preg_replace(
      array(
         // Remove invisible content
         '@<head[^>]*?>.*?</head>@siu',
         '@<style[^>]*?>.*?</style>@siu',
         '@<script[^>]*?.*?</script>@siu',
         '@<object[^>]*?.*?</object>@siu',
         '@<embed[^>]*?.*?</embed>@siu',
         '@<applet[^>]*?.*?</applet>@siu',
         '@<noframes[^>]*?.*?</noframes>@siu',
         '@<noscript[^>]*?.*?</noscript>@siu',
         '@<noembed[^>]*?.*?</noembed>@siu',

         // Add line breaks before & after blocks
         '@<((br)|(hr))@iu',
         '@</?((address)|(blockquote)|(center)|(del))@iu',
         '@</?((div)|(h[1-9])|(ins)|(isindex)|(p)|(pre))@iu',
         '@</?((dir)|(dl)|(dt)|(dd)|(li)|(menu)|(ol)|(ul))@iu',
         '@</?((table)|(th)|(td)|(caption))@iu',
         '@</?((form)|(button)|(fieldset)|(legend)|(input))@iu',
         '@</?((label)|(select)|(optgroup)|(option)|(textarea))@iu',
         '@</?((frameset)|(frame)|(iframe))@iu',
      ),
      array(
         ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
         "\n\$0", "\n\$0", "\n\$0", "\n\$0", "\n\$0", "\n\$0",
         "\n\$0", "\n\$0",
      ),
      $text);

   // Remove all remaining tags and comments and return.
   return strip_tags($text);
}
```

Gambar 4.3 Kode Sumber Fungsi *strip_html_tags*

1. Tahap Pembersihan

Tahap pembersihan menggunakan fungsi *cleanup* pada kelas *Tag_Content*. Kode sumber dari fungsi *cleanup* dapat dilihat pada Gambar 4.2.

Langkah awal dari fungsi pembersihan yaitu membersihkan teks dari tag-tag HTML. Karena fungsi bawaan dari PHP yaitu fungsi *strip_tags* tidak memadai karena merusak struktur penulisan teks hasil maka untuk membuang tag-tag HTML digunakan fungsi *strip_html_tags* yang dikembangkan oleh David R. Nadeau pada tahun 2008. Kode sumber dari fungsi *strip_html_tags* dapat dilihat pada Gambar 4.3.

Setelah membersihkan tag-tag HTML kemudian dilakukan konversi karakter-karakter HTML menjadi karakter normal dan proses terakhir yaitu menyeragamkan semua karakter spasi kosong (*whitespace*) menjadi sebuah karakter spasi sehingga memudahkan saat proses tokenisasi berlangsung.

```
76  protected function tokenize($text)
77  {
78      $tokens = explode(' ', $text);
79      $terms = array();
80      if ($tokens) foreach ($tokens as $term) {
81
82          # If the term is empty, skip it, since we probably just have
83          # multiple whitespace cahracters.
84          if ('' == $term)
85              continue;
86
87          # Now, a word can be preceded or succeeded by symbols, so let's
88          # split those out
89          $matches = array();
90          $bool = preg_match(self::TERM_SPEC, $term, $matches);
91
92          if (!$matches)
93              $terms[] = $term;
94          else {
95              unset($matches[0]);
96              foreach ($matches as $match) {
97                  if ('' != $match)
98                      $terms[] = $match;
99              }
100         }
101     }
102
103     return $terms;
104 }
```

Gambar 4.4 Kode Sumber Fungsi *tokenize*

2. Tahap Tokenisasi

Proses tokenisasi dilakukan pada teks yang telah dibersihkan dengan menggunakan fungsi *tokenize*, kode sumber dari fungsi *tokenize* dapat dilihat pada Gambar 4.4.

Hasil kembalian dari fungsi *tokenize* adalah larik *terms* yang berisi token-token untuk proses POS *Tagging*.

3. Tahap POS *Tagging*

Proses POS *Tagging* dilakukan pada token-token melalui dua tahap utama, tahap pertama yaitu mencari tag pada *lexicon* dan tahap kedua yaitu normalisasi tag yang telah ditemukan. Kode sumber dari fungsi tag dapat dilihat pada Gambar 4.5.

```
106  public function tag($terms)
107  {
108      $tagged_terms = array();
109
110      # Phase 1: Assign the tag from the lexicon. If the term is not found,
111      # it is assumed to be a default noun (NND).
112      if ($terms) foreach ($terms as $term) {
113          $tag = $this->lexicon[$term];
114          $tagged_terms[] = array(
115              $term,
116              $tag ? $tag : 'NND',
117              strtolower($term)
118          );
119      }
120
121
122      # Phase 2: Run through some rules to improve the term tagging and
123      # normalized form.
124      if ($tagged_terms) foreach ($tagged_terms as $idx => &$tagged_term) {
125          $this->correctDefaultNounTag($idx, $tagged_term, $tagged_terms);
126          $this->verifyProperNounAtSentenceStart($idx, $tagged_term, $tagged_terms);
127          $this->determineVerbAfterModal($idx, $tagged_term, $tagged_terms);
128          $this->normalizePluralForms($idx, $tagged_term, $tagged_terms);
129      }
130
131      return $tagged_terms;
132
133  }
```

Gambar 4.5 Kode Sumber Fungsi *tag*

4. Tahap Ekstraksi dan Pembobotan

Tahap ektraksi dan pembobotan dilakukan dengan menggunakan fungsi *extract*. Kode sumber pada tahap ektraksi dapat dilihat Gambar 4.6.

Pada tahap ektraksi setiap term pada variable *taggedTerms* diiterasi satu demi satu untuk menemukan jenis kata dan membentuk kata majemuk. Dalam iterasi juga dilakukan perhitungan frekuensi kemunculan kata.

```
233 public function extracts($taggedTerms)
234 {
235     $terms = array();
236     # Phase 1: A little state machine is used to build simple and
237     # composite terms.
238     $multiterm = array();
239     $state = self::SEARCH;
240     while ($taggedTerms) {
241         list($term, $tag, $norm) = array_pop($taggedTerms);
242 
243         if ($state == self::SEARCH and $tag[0] == 'N') {
244             $state = self::NOUN;
245             $this->add($term, $norm, $multiterm, $terms);
246         }
247         elseif ($state == self::SEARCH and $tag == 'JJ'
248                 and $term[0] == strtoupper($term[0]) ){
249             $state = self::NOUN;
250             $this->add($term, $norm, $multiterm, $terms);
251         }
252         elseif ($state == self::NOUN and $tag[0] == 'N') {
253             $this->add($term, $norm, $multiterm, $terms);
254         }
255         elseif ($state == self::NOUN and $tag[0] != 'N'){
256             $state = self::SEARCH;
257             if (count($multiterm) > 1){
258                 $multiterms = array();
259                 foreach ($multiterm as $item) {
260                     $multiterms[] = $item[0];
261                 }
262                  $word = implode(' ', array_reverse($multiterms));
263                  $terms[$word] += 1;
264             }
265             $multiterm = array();
266         }
267     }
```

Gambar 4.6 Kode Sumber Fungsi *extracts*

Sedangkan proses pembobotan pada fungsi *extracts* ini dapat dilihat pada Gambar 4.7. Proses pembobotan dilakukan dua tahap. tahap pertama yaitu menentukan nilai bobot awal dan menentukan nilai batas maksimum dari bobot awal. Proses kedua yaitu melakukan normalisasi terhadap bobot awal dengan membaginya dengan batas maksimum bobot awal yang telah ditemukan. Setelah bobot akhir ditemukan larik kembalian diurutkan terlebih dahulu mulai dari bobot terbesar ke yang terkecil sebelum dikembalikan dari fungsi *extracts*.

```
269     $filtered = array();
270     $maxFreq  = 0;
271     $maxLen   = 0;
272
273     if ($terms) foreach ($terms as $word => $occur) {
274        $len = count(explode(' ', $word));
275
276        if ((1 == $len and $occur >= self::SINGLE_TERM_MIN_OCCUR)
277            or ($len > 1 and $occur >= self::COMPOUND_TERM_MIN_OCCUR)) {
278           $filtered[strtolower($word)] = array($occur, $len);
279
280           if ($len > $maxLen)
281              $maxLen = $len;
282           if ($occur > $maxFreq)
283              $maxFreq = $occur;
284        }
285     }
286
287     $maxWeight = 0;
288     $tags = array();
289     if ($filtered) foreach ($filtered as $term => $counts) {
290        $weight = ($counts[0] / $maxFreq) + ($counts[1] / $maxLen);
291
292        if ($weight > $maxWeight)
293           $maxWeight = $weight;
294
295        $tags[$term] = $weight;
296     }
297
298     // normalize weight
299     if ($tags) foreach ($tags as $tag => $weight) {
300        $tags[$tag] = $weight / $maxWeight;
301     }
302
303     arsort($tags, SORT_NUMERIC);
304
305     return $tags;
306 }
```

Gambar 4.7 Kode Sumber Pembobotan pada Fungsi *extracts*

4.3.2 Implementasi Modul *Collaborative*

1. Tahap Pembangunan Model

Pembangunan model merupakan proses yang harus dilakukan sebelum perhitungan rekomendasi dilakukan. Kode sumber dari fungsi pembangunan model dapat dilihat pada Gambar 4.8.

Dalam proses pembuatan model diawali dengan mengambil seluruh *tag_id* yang ada dalam basis data. Pada fungsi ini untuk semua koneksi ke tabel pada database menggunakan kelas tabel *mapper* seperti kelas Tag, PostTag dan Model yang telah memiliki metode-metode untuk mengambil dan memanipulasi data pada tabel yang bersangkutan. Proses berikutnya yaitu mengosongkan tabel

models dengan fungsi *truncate* pada kelas tabel *mapper* Model. Setelah tabel models kosong perulangan bersarang untuk mencari kombinasi antar tag dilakukan. Untuk setiap pasangan tag dihitung *similarity* nya dengan mengunakan Persamaan 2.4 dan dimasukkan ke dalam larik *similarities*. Masukan dari fungsi perhitungan *similarity* yaitu larik post terkait dari masing-masing tag. Untuk mendapatkan kode post terkait dari suatu tag menggunakan fungsi *fetchAllPostId* dari objek tabel *mapper* PostTag. Kode sumber fungsi perhitungan *similarity* yaitu pada Gambar 4.9.

Setelah semua pasangan yang mungkin untuk suatu tag selesai dihitung kemudian larik *similarities* diurutkan dari yang terbesar hingga yang terkecil dengan fungsi PHP *arsort*. Langkah terakhir yaitu memasukkan sejumlah *k* dari larik *similarities* ke dalam tabel *models* pada basis data. Proses memasukkan ke tabel model dilakukan dengan melakukan iterasi hingga jumlah iterasi yang dilakukan telah mencapai *k*. Di dalam proses iterasi larik *similarities* yang telah diurutkan tersebut kemudian dimasukkan ke dalam tabel model dengan menggunakan fungsi *insert* pada objek tabel *mapper* Model.

Proses perhitungan *similarity* pada Gambar 4.9 dilakukan dengan lima tahap. tahap pertama yaitu menggabungkan kedua larik kode post masukkan dan mencari kode post yang unik dari keduanya dengan menggunakan fungsi bawaan dari PHP yaitu `array_merge` dan `array_unique`. Langkah berikutnya dilakukan pengecekan jika jumlah kode post yang beririsan dari kode larik ternyata kurang dari dua kode post maka perhitungan *similarity* tidak akan dilakukan karena dianggap tidak memenuhi syarat. Untuk menghitung jumlah elemen dalam larik digunakan fungsi PHP `count`. Langkah berikutnya yaitu melakukan iterasi untuk membentuk larik binari dari kode post. Setelah larik binari selesai dibentuk baru dilakukan iterasi untuk menghitung perkalian dan penjumlahan dari elemen-elemen larik yang berasosiasi yang mana pada proses terakhir hasil perhitungan ini akan digunakan sebagai masukan dalam perhitungan cosinus antar kedua vektor larik.

```php
public function generate($k)
{
    $Tag     = new Tag();
    $PostTag = new PostTag();
    $Model   = new Model();

    // select distinct all tags
    $tags = $Tag->fetchAllTagId();
    $m    = is_array($tags) ? count($tags) : 0;

    // no tags available
    if (!$m)
        return false;

    // truncate table model
    $Model->truncate();

    for ($i = 0; $i < $m; $i ++) {
        $tagI = $tags[$i];
        $similarities = array();
        for ($j = 0; $j < $m; $j ++) {
            $tagJ = $tags[$j];
            if ($i != $j) {
                // select all item related to j and i
                $postJ = $PostTag->fetchAllPostId($tagJ);
                $postI = $PostTag->fetchAllPostId($tagI);

                // calculate similarity
                $similarity = Tags_Collaborative::cosineSimilarity($postI, $postJ);

                // save to models for M[j][i]
                $similarities[$tagJ] = $similarity;
            }
        }

        arsort($similarities);
        $x = 0;
        if ($similarities) foreach ($similarities as $tagJ => $similarity) {
            $x++;

            if ($similarity) {
                // insert model
                $data = array(
                    'tag_i'      => $tagI,
                    'tag_j'      => $tagJ,
                    'similarity' => $similarity
                );
                $Model->insert($data);
            }

            if ($x >= $k)
                break;
        }
    }
}
```

Gambar 4.8 Kode Sumber Fungsi *generate*

```php
public static function cosineSimilarity($a, $b)
{
    $i = $j = $k = 0;
    $uniqueA = $uniqueB = array();

    $merged = array_merge($a, $b);
    $uniqueMerged = array_unique($merged);

    // setiap item setidaknya harus memiliki minimal 2 post yg sama
    if ((count($merged) - count($uniqueMerged)) < 2)
        return 0;

    foreach ($a as $token)
        $uniqueA[$token] = 0;
    foreach ($b as $token)
        $uniqueB[$token] = 0;

    foreach ($uniqueMerged as $token) {
        $x = isset($uniqueA[$token]) ? 1 : 0;
        $y = isset($uniqueB[$token]) ? 1 : 0;
        $i += $x * $y;
        $j += $x;
        $k += $y;
    }
    return $j * $k != 0 ? $i / sqrt($j * $k) : 0;
}
```

Gambar 4.9 Kode Sumber Fungsi *cosineSimilarity*

2. Tahap Pengaplikasian Model

Tahap pengaplikasian model digunakan saat akan menghitung rekomendasi akhir dari modul *Collaborative* dengan menggunakan algoritma pada Gambar 2.7. Masukan pada tahap ini yaitu larik tag pos baru yang akan diberikan rekomendasinya sebagai tag-tag asal, dan jumlah rekomendasi yang diinginkan. Proses yang dilakukan yaitu mengambil entri-entri pada tabel *models* dari tag-tag asal dengan menggunakan fungsi getModel pada objek tabel *mapper* Model. Setelah model didapatkan kemudian dikalikan dengan tag-tag asal dengan fungsi *multiply*. Hasil perkalian tersebut kemudian diurutkan dari yang terbesar ke yang terkecil dengan fungsi *arsort*. Dan langkah terakhir yaitu mengambil sejumlah N rekomendasi yang diinginkan yang belum ada pada tag-tag asal. Kode sumber dari proses pengaplikasian model dapat dilihat pada Gambar 4.10.

```php
public function apply($tags, $N)
{
    // load model
    $Model = new Model();

    $this->model = $Model->getModel();

    // calculate recomendation
    $m = count($tags);
    $x = $this->multiply($tags);

    arsort($x);
    $i = 0;
    $result = array();
    if ($x) foreach ($x as $tag => $val) {
        if (0 != $val && !isset($tags[$tag])) {
            $i++;
            $result[$tag] = $val;

            if ($i >= $N)
                break;
        }
    }
    return $result;
}

protected function multiply($tag)
{
    $result = array();
    foreach ($this->model as $i => $tags) {
        foreach ($tags as $j => $similarity) {
                $result[strval($j)] += $similarity * $tag[$i];
        }
    }
    return $result;
}
```

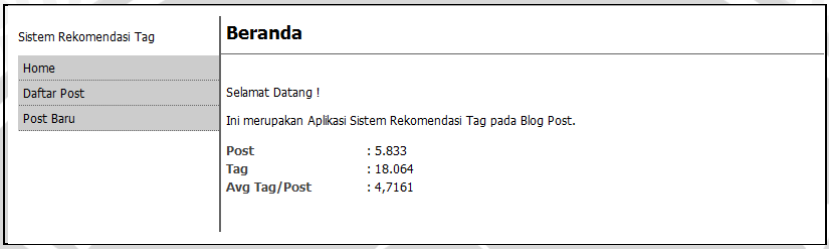
Gambar 4.10 Kode Sumber Fungsi *apply*

4.4 Implementasi Antar Muka

Implementasi antar muka terbagi menjadi lima halaman, yaitu halaman beranda, halaman daftar post, halaman detil post, halaman edit post, dan halaman post baru.

4.4.1 Halaman Beranda

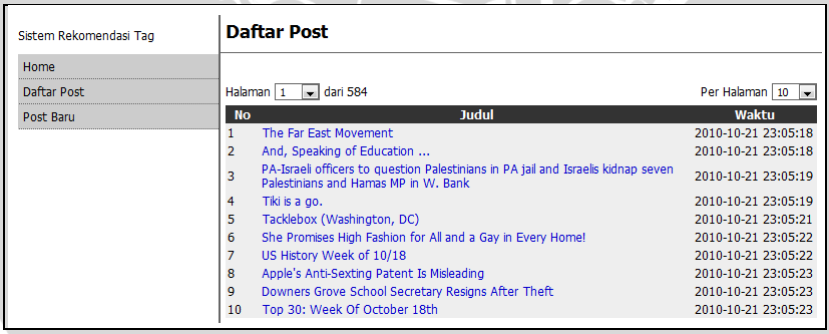
Pada halaman beranda terdapat sambutan dan jumlah data yang terdapat pada basis data yang digunakan sebagai acuan data. Tampilan dari halaman beranda dapat dilihat pada Gambar 4.11.



Gambar 4.11 Tampilan Halaman Beranda

4.4.2 Halaman Daftar Post

Pada halaman daftar post terdapat terdapat daftar post yang ada pada basis data yang mana bisa ditampilkan pada halaman dan jumlah data perhalaman tertentu. Untuk menuju halaman detail post dapat dilakukan dengan mengklik judul post. Tampilan dari halaman daftar post dapat dilihat pada Gambar 4.12.

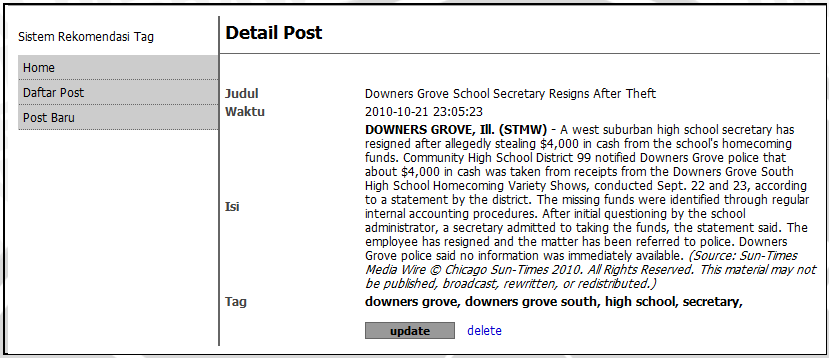


Gambar 4.12 Tampilan Halaman Daftar Post

4.4.3 Halaman Detil Post

Pada halaman detil post terdapat detail post berikut tombol aksi untuk mengubah atau menghapus post. Saat akan menghapus post akan muncul *pop up* konfirmasi untuk memastikan pengguna

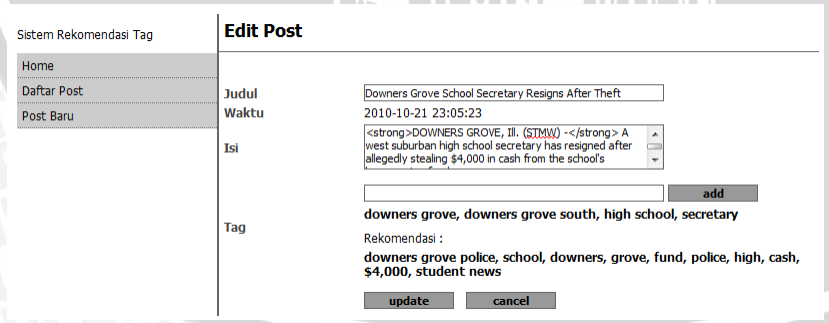
yakin akan menghapus post tersebut. Tampilan dari halaman detil dapat dilihat pada Gambar 4.13.



Gambar 4.13 Tampilan Halaman Detail Post

4.4.4 Halaman Edit Post

Pada halaman edit post terdapat detil post dengan beberapa *field* yang bisa diedit seperti judul, isi dan tag. Untuk menambahkan tag bisa dilakukan dengan memasukan pada input tag kemudian mengklik *add* ataupun dengan memilih rekomendasi tag yang ditawarkan sesuai dengan isi dan tag yang telah diberikan oleh pengguna sebelumnya. Untuk menghapus tag dapat dilakukan dengan mengklik tag yang ingin dihapus. Penyimpanan perubahan yang dilakukan dengan mengklik tombol *update* dan *cancel* untuk membatalkan perubahan. Tampilan dari halaman edit post dapat dilihat pada Gambar 4.14.



Gambar 4.14 Tampilan Halaman Edit Post

4.4.5 Halaman Post Baru

Tampilan halaman post baru menyerupai tampilan halaman edit post seperti pada Gambar 4.4 hanya saja masukan-masukan seperti judul, isi dan tag dalam keadaan kosong. Setelah pengguna selesai menulis isi post dan akan menambahkan tag dapat memanfaatkan fitur rekomendasi tag dengan mengklik tautan **Daftar Rekomendasi Tag** yang mana tag-tag rekomendasi yang bisa dipilih akan dimunculkan. Tampilan dari halaman beranda dapat dilihat pada Gambar 4.15.

Gambar 4.15 Tampilan Halaman Post Baru

4.5 Implementasi Uji Coba

Uji coba dilakukan pada data yang telah disiapkan sebelumnya seperti yang telah dijelaskan pada subbab 4.3. Pengujian menggunakan data uji sejumlah 100 post teratas yang memiliki jumlah tag antara 10-20 tag. Rincian pengujian yang dilakukan adalah sebagai berikut:

4.5.1 Pembobotan pada modul *Content Based Filtering*

Pengujian pertama yang dilakukan yaitu pengujian skema pembobotan pada modul Content yang mana menggunakan metode *Content Based Filtering*. Dari tiga skema pembobotan yang telah dirancang pada bagian 3 subbab 3.4.1 setelah dilakukan pengujian terhadap data uji dapat dilihat hasilnya seperti pada Tabel 4.1.

Tabel 4.1 Pembobotan pada modul *Content Based Filtering*

No	Metode	*Precission*	*Recall*	*Recom. Time(s)*
1	frekuensi ternormalisasi	0.153133897	0.086655022	0.097397428
2	panjang kata ternormalisasi	0.135697759	0.069218884	0.097121160
3	kombinasi frekuensi dan panjang kata yang dinormalisasi	0.157265679	0.090786805	0.097261374

Dari pengujian yang dilakukan dapat dilihat skema pembobotan yang paling optimal yaitu pada no 3 yaitu kombinasi frekuensi dan panjang kata yang dinormalisasi yang menghasilkan nilai *precission* sebesar 0.157265679 dan *recall* sebesar 0.090786805. waktu rekomendasi yang dibutuhkan juga masih jauh dibawah ambang batas 0.5 detik yaitu standar batasan maksimal waktu eksekusi fungsi terlama pada aplikasi ZendServer yaitu webserver yang dikembangkan oleh Zend yang juga mengembangkan engine PHP yang digunakan secara umum.

4.5.2 Parameter *k* pada modul *Collaborative*

Pengujian berikutnya yaitu pengujian terhadap parameter jumlah *similarity* atau kemiripan yang disimpan pada tabel *models* di basis data untuk setiap tag nya. Pengujian dilakukan pada beberapa jumlah data yang berbeda dengan nilai *k* yang berbeda-beda. Hasil pengujian dapat dilihat pada Tabel 4.2.

Pengujian dilakukan secara bertahap yaitu dengan jumlah data sebesar 1.000 dan 10.000 dengan jumlah *k* mulai dari 10, 15, 20, 25, dan 30. Setelah percobaan dilakukan ditemukan nilai *k* optimal yaitu 30 dimana nilai *k* sebesar 30 ini bisa menghasilkan nilai *precission* dan *recall* setara bahkan lebih baik saat seluruh tag digunakan dalam pembuatan model. Setelah jumlah *k* optimal ditemukan yaitu 30 tahap terakhir yaitu menjalankan semua data yang ada dengan *k* sejumlah 30. Percobaan yang melibatkan seluruh data hanya dilakukan sekali karena waktu pembuatan model membutuhkan waktu yang cukup lama yaitu 28542.33 detik atau 7 jam 55 menit 42.33 detik.

Tabel 4.2 Parameter *k* pada modul *Collaborative*

No	N Data	k	*Model time(s)*	*Precission*	*Recall*	*Recom. time*
1	1000	10	23.9752	0.053442	0.016567	0.009269
2	1000	15	31.8279	0.054838	0.018233	0.009324
3	1000	20	29.1030	0.054838	0.018233	0.009439
4	1000	25	26.6605	0.054838	0.018233	0.009593
5	1000	30	26.6083	0.054838	0.018233	0.009633
6	1000	1000	23.9675	0.054838	0.018233	0.011374
7	10000	10	716.7507	0.068913	0.051856	0.010592
8	10000	15	759.4646	0.068675	0.051838	0.010926
9	10000	20	755.1875	0.065209	0.048120	0.011074
10	10000	25	817.2579	0.065579	0.050462	0.011531
11	10000	30	749.1396	0.068645	0.055905	0.011597
12	10000	10000	698.7128	0.064333	0.067039	0.020136
13	155058	30	28542.33	0.078878	0.068132	0.017963

4.5.3 Proporsi modul *Content* dan modul *Collaborative*

Pengujian ketiga yaitu pengujian untuk mendapatkan nilai optimal dari proporsi antara jumlah tag dari modul *Content* dan modul *Collaborative*. Pengujian dilakukan dengan mengubah-ubah persentase dari kedua modul dan mencatat perubahan *pressision* dan *recall* yang dihasilkan. Selain itu juga dilihat perubahan *recommendation time* terhadap pengujian yang dilakukan. Pengujian menggunakan jumlah rekomendasi 10 tag teratas. Jumlah ini menyesuaikan dari desain tampilan antar muka penampilan hasil rekomendasi. Hasil pengujian dapat dilihat pada Tabel 4.3.

Dari hasil percobaan yang dilakukan terlihat pada saat modul *Content* tidak digunakan maka sistem akan gagal memberikan rekomendasi karena modul *Collaborative* membutuhkan masukan berupa tag-tag dari modul *Content*. Dengan mengubah-ubah persentasi dari kedua modul dapat dilihat bahwa nilai *precission* optimal yang didapatkan yaitu pada percobaan ke 6 dimana persentasi dari modul *Content* sebesar 90% dan modul *Collaborative* 10% yang mana menghasilkan nilai *precission* sebesar 0.148071429 dan waktu rekomendasi juga masih di bawah batas 0.5 detik.

Tabel 4.3 Proporsi modul *Content* dan modul *Collaborative*

No	*% Content*	*% Collab.*	*Precission*	*Recall*	*Recom. time*
1	0	100	0	0	0.119509509
2	30	70	0.090218254	0.045544429	0.124336774
3	50	50	0.109369048	0.062492411	0.125280657
4	70	30	0.122202381	0.067369182	0.125473895
5	80	20	0.134809524	0.071036225	0.125328748
6	90	10	0.148071429	0.073663532	0.125180859
7	100	0	0.144798701	0.076156263	0.125206804

4.5.4 Skalabilitas

Pengujian terakhir yaitu pengujian kemampuan skalabilitas dari sistem yang dibangun. Karena sistem yang dibangun menggunakan algoritma *Collaborative Based Filtering* yang mana membutuhkan jumlah data yang sangat besar untuk dapat menghasilkan nilai *precission* dan *recall* yang lebih tinggi. Percobaan dilakukan sebanyak enam kali dengan jumlah data tag unik 1.000, 5.000, 10.000, 50.000, 100.000 dan 155.058 serta mencatat hubungan antara jumlah data dengan waktu yang dibutuhkan untuk menghasilkan rekomendasi. Hasil percobaan dapat dilihat pada Tabel 4.4.

Tabel 4.4 Skalabilitas

No	N Data	*Recomendation time*
1	1000	0.009632535
2	5000	0.011176105
3	10000	0.011596742
4	50000	0.012333276
5	100000	0.012901573
6	155058	0.017962599

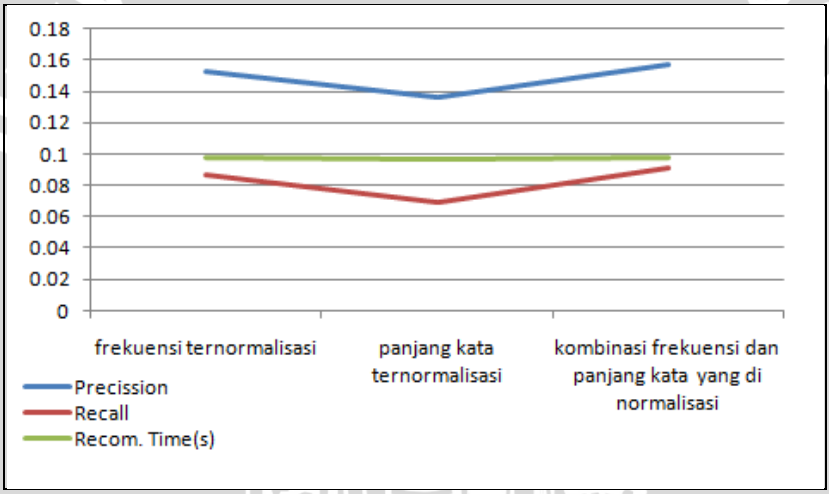
Dari hasil pengujian pada Tabel 4.4 dapat dilihat sistem masih mampu memberikan waktu rekomendasi di bawah ambang batas 0.5 detik meski semua data telah digunakan yaitu sebesar 0.017962599 detik.

4.6 Analisis Hasil

Berikut adalah analisis hasil dari data-data hasil penelitian yang telah didapatkan:

4.6.1 Pembobotan pada modul *Content Based Filtering*

Dari data pembobotan pada modul *Content Based Filtering* pada Tabel 4.1 dapat dilihat pengaruh dari skema pembobotan yang berbeda terhadap *precission*, *recall* dan waktu rekomendasi seperti pada grafik yang ditampilkan pada Gambar 4.16.



Gambar 4.16 Grafik Pembobotan Modul *Content*

Dari ketiga skema pembobotan yang digunakan tampak bahwa skema pembobotan yang paling optimal yaitu dengan skema kombinasi frekuensi dan panjang kata yang dinormalisasi. Hal tersebut tampak dari nilai *precission* dan *recall* yang paling tinggi sedangkan waktu rekomendasi tampak tidak ada perbedaan yang signifikan antara ketiga skema pembobotan tersebut.

4.6.2 Parameter *k* pada modul *Collaborative*

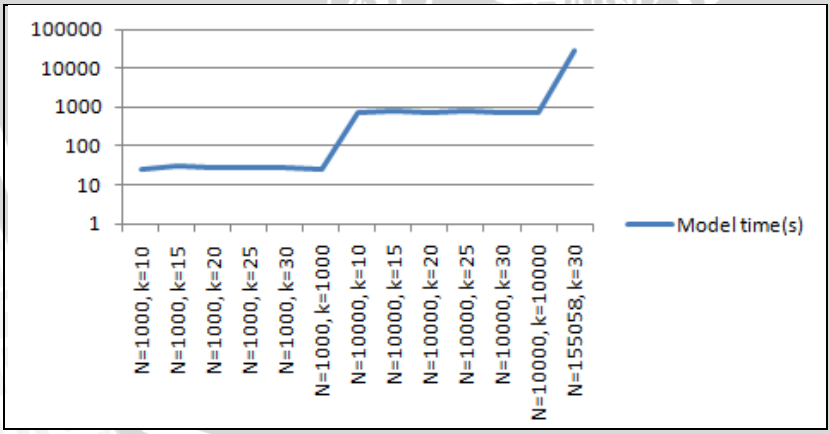
Dari data pengujian parameter *k* pada modul *Collaborative* pada Tabel 4.2 dihasilkan beberapa grafik hubungan antara jumlah *N* data dan nilai parameter *k* yang digunakan terhadap waktu

pembuatan model, *precission, recall*, dan terhadap waktu yang dibutuhkan untuk menghasilkan rekomendasi. Grafik-grafik tersebut dapat dilihat pada Gambar 4.7.

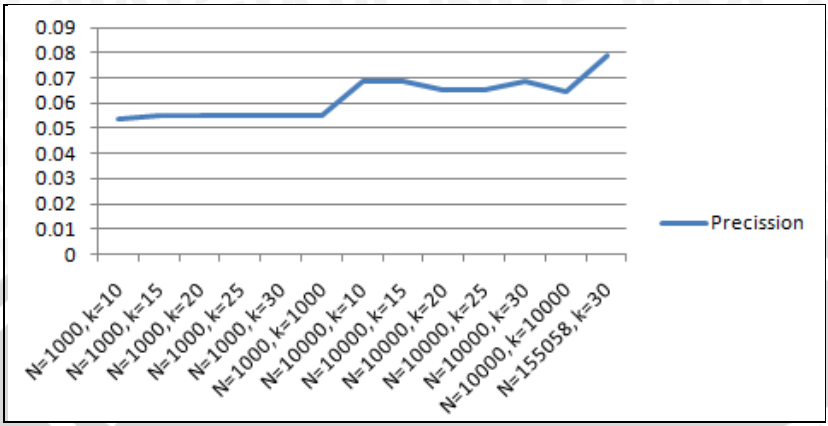
Dari Grafik Pembuatan Model pada Gambar 4.17 dapat dilihat bahwa waktu pembuatan model lebih dipengaruhi oleh jumlah *N* data yang digunakan karena parameterisasi *k* yaitu jumlah *similarity* yang disimpan ke dalam model tidak begitu berpengaruh karena proses menyimpan ke dalam tabel *models* pada basis data berlangsung sangat singkat terutama pada nilai *k* yang kecil.

Dari Grafik *precission* pada Gambar 4.18 dan *recall* pada Gambar 4.19 tampak nilai *precission* dan *recall* juga meningkat mengikuti peningkatan jumlah *N* data pada model. Semakin banyak data yang digunakan maka sistem akan memberikan hasil rekomendasi yang semakin akurat. Nilai *precission* dan *recall* juga sudah optimal pada angka *k*=30 sehingga peningkatan nilai *k* tidak diperlukan sehingga tidak terjadi penurunan performa yang dapat dilihat pada Grafik Waktu Rekomendasi pada Gambar 4.20.

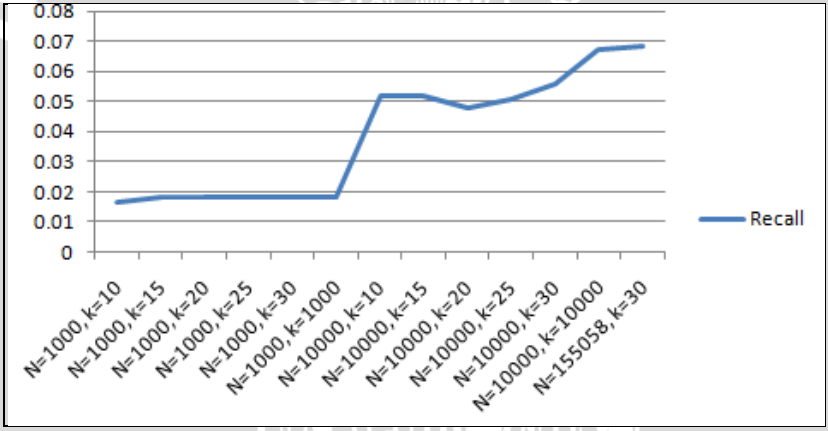
Waktu yang dibutuhkan untuk menghasilkan rekomendasi digambarkan pada Grafik Waktu Rekomendasi pada Gambar 4.20. grafik ini menggambarkan performa sistem dalam memberikan rekomendasi. Waktu rekomendasi lebih dipengaruhi oleh nilai parameter *k* yang digunakan dan tidak terpengaruh oleh jumlah N data



Gambar 4.17 Grafik Waktu Pembuatan Model



Gambar 4.18 Grafik *Precission*

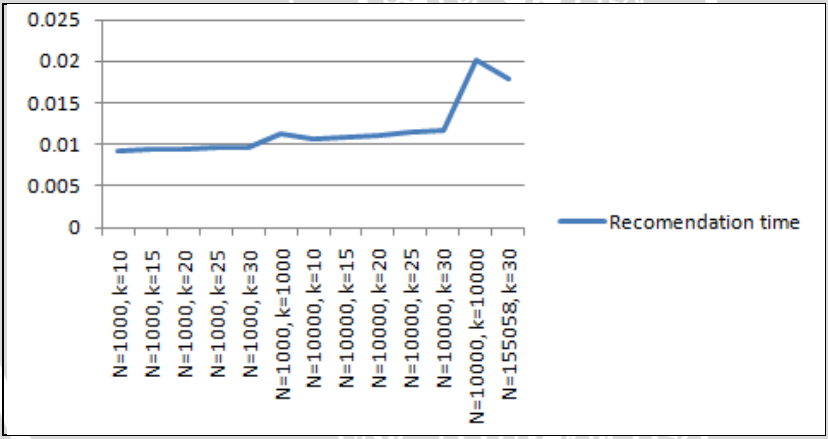


Gambar 4.19 Grafik *Recall*

4.6.3 Proporsi modul *Content* dan modul *Collaborative*

Hasil pengujian proporsi modul *Content* dan modul *Collaborative* pada Tabel 4.3 dapat digambarkan menjadi grafik seperti pada Gambar 4.21. dari grafik tersebut dapat dilihat bahwa nilai *precission* dan *recall* mencapai nilai optimalnya pada saat nilai presentasinya antara 90-100%. Hal ini menunjukan modul *Content* yang jauh lebih berpengaruh dibandingkan modul *Collaborative* dalam menghasilkan rekomendasi yang akurat. Peningkatan nilai *precission* terjadi pada saat nilai persentasi dari modul *Collaborative*

hanya bernilai 10%. Dengan jumlah rekomendasi yang diberikan adalah sejumlah 10 rekomendasi teratas maka perbandingan antara jumlah tag dari modul *Content* dan modul *Collaborative* yaitu 9 berbanding 1 tag. Dimana nilai optimal ini menunjukan rata-rata akurasi dari modul *Collaborative* hanya 1 tag dari 10 tag yang direkomendasikan sehingga baru bisa melengkapi tag-tag dari modul *Content*. Waktu rekomendasi tidak ada perbedaan yang signifikan terhadap perubahan komposisi antara kedua modul yang digunakan. Pada persentase modul Content 0% pencarian kode tag tidak perlu dilakukan sehingga dapat berjalan sedikit lebih cepat. Kenaikan waktu rekomendasi lebih banyak diakibatkan oleh peningkatan akurasi modul *Content* yang mana hasilnya akan mengakibatkan perhitungan pada modul *Collaborative* menjadi lebih banyak. Pada persentase modul *Content* 50%-100% waktu rekomendasi relatif sama dan anomali dibawah satuan mikro detik lebih dikarenakan kondisi sistem dan basisdata saat pengujian dilakukan.

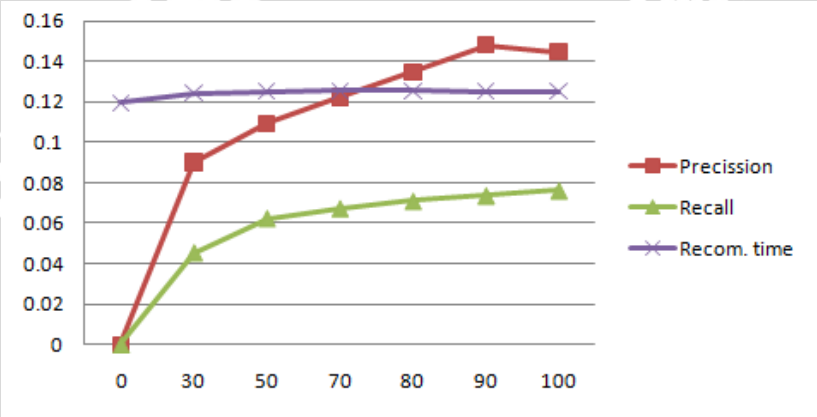


Gambar 4.20 Grafik Waktu Rekomendasi

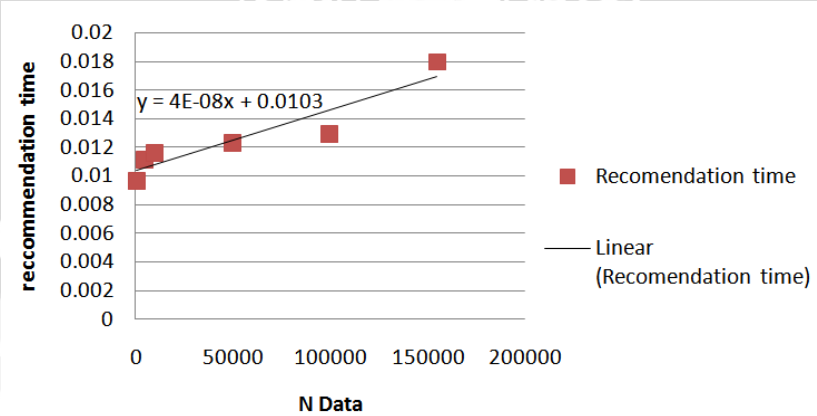
4.6.4 Skalabilitas

Dari data pengujian skalabilitas pada Tabel 4.4 dapat dibentuk grafik yang terlihat pada Gambar 4.22. Dari grafik tampak waktu rekomendasi masih dipengaruhi oleh jumlah data yang digunakan. Jumlah data yang semakin besar akan membuat proses pengambilan data pada tabel *models* semakin melambat. Namun dari grafik bisa dilihat hubungan pertambahan waktu rekomendasi terhadap jumlah

data berbentuk linear dengan kemiringan yang cukup landai dengan persamaan y = 0.00000004x + 0.0103. Dengan persamaan kemiringan tersebut batas standar waktu maksimal performa sistem yaitu 0.5 detik yang mana merupakan batas maksimal eksekusi fungsi pada ZendServer akan tercapai setelah jumlah tag unik sejumlah 12.242.500 tag yang mana hal ini akan lama tercapai karena setiap pertambahan post dan tag belum tentu menambahkan jumlah tag jika tag yang digunakan sudah pernah ada dalam basis data.



Gambar 4.21 Grafik Proporsi modul *Content* dan modul *Collaborative*



Gambar 4.22 Grafik Skalabilitas


UNIVERSITAS BRAWIJAYA

BAB IV
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil uji dan analisis yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Telah dirancang sistem rekomendasi tag untuk posting blog dengan kombinasi algoritma *Content Based Filtering* dengan algoritma *Collaborative Filtering*.
2. Telah diimplementasikan rancangan sistem rekomendasi yang telah disusun pada sebuah sistem berbasis web.
3. Telah dilakukan uji coba dan menganalisa hasil uji coba dari sistem rekomendasi yang telah dibuat yang mana penggabungan metode *Content Based Filtering* dan *Collaborative Filtering* menghasilkan rekomendasi yang lebih presisi daripada jika berdiri sendiri pada proporsi antara modul *Content* dan modul *Collaborative* adalah 9 berbanding 1 tag serta skalabilitas sistem juga masih terjaga dengan hubungan perbandingan antara perubahan waktu rekomendasi(y) dan jumlah N-data(x) pada persamaan $y = 0.00000004x + 0.0103$.

5.2 Saran

Beberapa saran yang dapat disampaikan untuk penelitian selanjutnya yaitu:

1. Pada modul *Content Based Filtering* penggunaan metode yang berbasis semantik sebaiknya tidak digunakan dikarenakan struktur penulisan blog yang sering tidak mengikuti struktur penulisan yang sesuai dengan kaidah Bahasa Inggris yang baik dan benar sehingga sering terjadi error pada saat POS tagging.
2. Metode *Collaborative Filtering* yang digunakan dibuat sebaiknya tidak tergantung pada modul *Content Based* sehingga dapat digunakan dalam menganalisa post dengan jumlah teks yang sedikit atau tidak ada sama sekali.

DAFTAR PUSTAKA

Anonim, 2009. *Tag Index* - Technorati. http://technorati.com/tag/. Diakses pada tanggal 26 November 2009.

Anonim, 2010. *Firehose*. http://en.wordpress.com/firehose/. diakses pada tanggal 19 Oktober 2010.

Anonim, 2010. *JSON*. http://www.json.org/json-id.html. Diakses pada tanggal 19 Oktober 2010.

Balabanovic, Marko dan Yoav Shoham. 1997. *Fab: content-based, collaborative recommendation*. New York. ACM.

Balby Marinho, Leandro dan Lars Schmidt-Thieme. 2008. *Collaborative Tag Recommendations*. Springer Berlin Heidelberg. Berlin.

Brown, Daniel, Nuno Lopes, Felipe Pena, Thiago Pojda dan Maciek Sokolewicz. 2010. *PHP Manual*. http://www.php.net/manual/. Diakses pada tanggal 26 November 2009

Claypool, Mark, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes dan Matthew Sartin. 1999. *Combining Content-Based and Collaborative Filters in an Online*. New York. ACM.

Deshpande, Mukund dan George Karypis. 2004. *Item-Based Top-N Recommendation Algorithms*. New York. ACM.

Golder, Scott A. dan Bernardo A. Huberman. 2005. *The Structure of Collaborative Tagging Systems*. Information Dynamics Lab, HP Labs.

Hart, Michael, Rob Johnson dan Amanda Stent. 2009. *iTag: A Personalized Blog Tagger*. New York. ACM.

Melville, Prem, Raymond J. Mooney dan Ramadass Nagarajan. 2001. *Content-Boosted Collaborative Filtering*. Department of Computer Sciences University of Texas. Texas.

Microsoft Encarta Online Encyclopedia. 2009. *Blog*. Microsoft Corporation.

Nadeau, David Robert. 2008. *PHP tip: How to extract keywords from a web page*. http://nadeausoftware.com/articles/2008/04/php_tip_how_extract_keywords_web_page. Diakses pada tanggal 19 Oktober 2010.

Nakamoto, Reyn, Shinsuke Nakajima, Jun Miyazaki, dan Shunsuke Uemura. 2007. *Tag-Based Contextual Collaborative Filtering*. Nara Institute of Science and Technology, Graduate School of Information Science, Database Laboratory. Japan.

Richter, Stephan, Russ Ferriday dan Zope Community. 2009. *Topia Termextract*. http://pypi.python.org/pypi/topia.termextract/. Diakses pada tanggal 26 November 2009

Rijsbergen, C. J. Van. 1979. *Information Retrieval 2nd ed*. Butterworth-Heinemann. St. Louis.

Schafer, J. B., J. A. Konstan, dan J. Riedl. 2001. *E-Commerce Recommendation Application*. Data Mining and Knowledge Discovery.

Shoval, Peretz, Veronica Maidel, dan Bracha Shapira. 2008. *An Ontology Content-Based Filtering Method*.

Weiyang, Lin. 2000. *Association Rule Mining for Collaborative Recommender Systems*. Thesis. Worcester Polytechnic Institute.

Zend Technologies. 2010. *Slow Function Execution*. http://files.zend.com/help/Zend-Server-IBMi/slow_function_execution.htm. Diakses pada tanggal 19 Oktober 2010.

LAMPIRAN

Lampiran 1 Tabel definisi tag *Part-of-speech*

Tag	keterangan
$	Dollar Sign or other currency symbol **
,	Comma
.	Period
:	Colon, semi-colon, and dash ('-' or '--' separating parts of a sentence)
``	Opening Quotation Marks (single or double); **tag consists of two grave accent characters**
''	Closing Quotation Marks (single or double); **tag consists of two ASCII apostrophe/single quote characters**
AFX	Affix; used in instances such as *non-, anti-*, and *pro-* (but not *-like*).
CC	Coordinating Conjunctions
CD	Cardinal Number
DT	Determiner
EX	Existential *there*, not referring to place, as in "there were"
FW	Foreign Word; ex. *in vitro, in vivo, corpora lutea* (all /FW /FW)
HYPH	Used to denote the following:
	* An English hyphen (more-or-less normal punctuation): "concentration-response"
	* A range indicator : "200-230"
	NOTE: Function was formerly called ***#*** *. Cf.* ***SYM***
IN	Preposition or Subordinating Conjunction
JJ	Adjective
JJR	Comparative Adjective
JJS	Superlative Adjective
-LRB-	"(", "[" and "{"
LS	List Item Marker; letters, numerals, and bullets used to identify list items

MD	Modal Verb; *can, could, may, might, must, ought, shall, will, would*
NN	Singular Noun
NNP	Singular Proper Noun
NNPS	Plural Proper Noun
NNS	Plural Noun
PDT	Pre-Determiner
POS	Possessive Ending *
PRP	Personal Pronoun
PRP$	Possessive Pronoun
RB	Adverb
RBR	Comparative Adverb
RBS	Superlative Adverb
RP	Particle
-RRB-	")", "]" and "}"
SYM	Symbol: "-" (negative or subtraction), +, *, /, <, >, =, +/-, Greek letter names, ...
TO	The word "to"
token	Tokenizer Tag -- no POS assigned
UH	Interjection **
VB	Base Form Verb
VBD	Past Tense Verb
VBG	Gerund or Past Participle Verb
VBN	Past Participle Verb
VBP	Non-3rd Person Singular Present Verb
VBZ	3rd Person Singular Present Verb
WDT	WH-Determiner
WP	WH-Pronoun
WP$	Possessive WH-Pronoun
WRB	WH-Adverb