

KNOWLEDGE & INFORMATION RETRIEVAL

Dr. Widodo Budiharto



Knowledge dan Information Retrieval

UU No 19 Tahun 2002 Tentang Hak Cipta

Fungsi dan Sifat hak Cipta Pasal 2

1. Hak Cipta merupakan hak eksklusif bagi pencipta atau pemegang Hak Cipta untuk mengumumkan atau memperbanyak ciptaannya, yang timbul secara otomatis setelah suatu ciptaan dilahirkan tanpa mengurangi pembatasan menurut peraturan perundang-undangan yang berlaku.

Hak Terkait Pasal 49

1. Pelaku memiliki hak eksklusif untuk memberikan izin atau melarang pihak lain yang tanpa persetujuannya membuat, memperbanyak, atau menyiarkan rekaman suara dan/atau gambar pertunjukannya.

Sanksi Pelanggaran Pasal 72

1. Barangsiapa dengan sengaja dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam pasal 2 ayat (1) atau pasal 49 ayat (2) dipidana dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp 1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 5.000.000.000,00 (lima miliar rupiah).
2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah)

Knowledge dan Information Retrieval

Dr. Widodo Budiharto





deepublish | publisher

Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp/Faks: (0274) 4533427
Website: www.deepublish.co.id
www.penerbitdeepublish.com
E-mail: deepublish@gmail.com

Katalog Dalam Terbitan (KDT)

BUDIHARTO, Widodo

Knowledge dan Information Retrieval/oleh Widodo Budiharto.--
Ed.1, Cet. 1--Yogyakarta: Deepublish, April 2016.

x, 155 hlm.; Uk:17.5x25 cm

ISBN 978-602-401-418-6

1. Teori Ilmu Pengetahuan

I. Judul
001.01

Hak Cipta 2016, Pada Penulis

Desain cover : Herlambang Ramadhani
Penata letak : Cinthia Morris Sartono

PENERBIT DEEPUBLISH
(Grup Penerbitan CV BUDI UTAMA)
Anggota IKAPI (076/DIY/2012)

Copyright © 2016 by Deepublish Publisher
All Right Reserved

Isi diluar tanggung jawab percetakan

Hak cipta dilindungi undang-undang
Dilarang keras menerjemahkan, memfotokopi, atau
memperbanyak sebagian atau seluruh isi buku ini
tanpa izin tertulis dari Penerbit.

KATA PENGANTAR

Assalamu'alaikum, Wr. Wb.

Alhamdulillah, segala puji dan syukur kami sampaikan ke hadirat Allah Swt., yang telah melimpahkan rahmat serta hidayah-Nya sehingga buku yang berjudul "*Knowledge dan Information Retrieval*" ini dapat terselesaikan dan kami terbitkan.

Pada era sebelum munculnya *search engine* seperti Google, Yahoo, Bing, orang mengandalkan pencarian informasi melalui orang lain. Saat ini hampir setiap orang menggunakan internet sebagai alat untuk menemukan informasi yang mereka butuhkan. Untuk memudahkan pencarian informasi, diperlukan peran *information retrieval* untuk mempercepat penemuan informasi yang dibutuhkan.

Information retrieval atau temu kembali informasi merupakan ilmu yang mempelajari prosedur dan metode untuk menemukan kembali informasi yang tersimpan di berbagai sumber (*resources*) yang relevan atau koleksi sumber informasi yang dicari atau dibutuhkan dengan tindakan *indexing*, *searching*, dan *recalling*.

Materi dalam buku ini secara lengkap membahas tentang prosedur dan metode *information retrieval* mulai dari *indexing*, *searching*, dan *recalling* sebagai modal awal untuk memahami *information retrieval*. Bab yang dibahas dalam buku ini meliputi *Index Construction and Compression*, *Text Classification*, *Multimedia Retrieval*, *Information Extraction* dan *Data Mining*, serta *Advanced XML Retrieval* yang dapat dijadikan acuan untuk mengembangkan kemampuan pengelolaan informasi.

Demi kesempurnaan dalam penyajian buku ini, saran dan kritik dari pembaca yang budiman sangat kami nantikan dan akan kami jadikan pedoman untuk penerbitan berikutnya, sehingga buku ini pun akan menjadi lebih sempurna, serta memperkaya khazanah pengetahuan kita bersama. Kami mengucapkan terima kasih kepada penulis, Dr. Widodo Budiharto dan Azani Cempaka Sari, S.Kom., MTI., yang telah memberikan perhatian, kepercayaan, dan kontribusi demi kesempurnaan buku ini. Semoga buku ini bermanfaat, dapat mencerdaskan dan memuliakan setiap insan.

Akhirnya, dengan diterbitkannya buku ini diharapkan dapat memberikan pemahaman komprehensif kepada mahasiswa mengenai *information retrieval*, serta bagi dosen diharapkan buku ini dapat membantu proses pembelajaran sebagai referensi ataupun bahan ajar. Selamat membaca.

Wassalamu'alaikum, Wr. Wb.

Hormat Kami,

Penerbit Deepublish

DAFTAR ISI

KATA PENGANTAR.....	v
DAFTAR ISI.....	vii
Bab 1	
Pengenalan Knowledge dan <i>Information Retrieval</i>	11
1.1 Definisi Knowledge dan <i>Information Retrieval</i>	11
1.2 Inverted Index	14
1.3 Boolean retrieval	17
1.4 Tokenization.....	19
1.5 Stemming and lemmatization	19
1.6 Dictionaries.....	20
1.7 Wildcard Queries	21
1.8 Vector Space Model	23
1.9 Implementasi.....	24
Bab 2	
Index Construction dan Compression.....	32
2.1 Pendahuluan.....	32
2.2 Blocked sort-based indexing	35
2.3 Distributed Indexing.....	36
2.4 Index Compression	38
2.5 Zipf's Law	39
2.6 Tolerant Retrieval.....	40
2.7 Implementasi.....	41
Bab 3	
Evaluasi Retrieval	51
3.1 Pendahuluan.....	51

Bab 4		
Relevance Feedback.....	58	Bab
4.1 Pendahuluan.....	58	Info
4.2 Algoritma Rocchio untuk relevant Feedback.....	59	9.1
4.3 Relevant Feedback di Web	60	9.2
4.4 Semantic Information Retrieval	61	9.3
		9.4
Bab 5		9.5
Text Classification	63	9.6
5.1 Text Classification	63	9.7
5.2 Decision Trees	66	9.8
5.3 Precision and Recall	67	9.9
5.4 SVM Classifier	68	9.10
5.5 NLTK untuk Klasifikasi.....	69	9.11
5.6 Classifier Precision dan recall	71	9.12
5.7 Naïve Bayes	72	9.13
5.8 Benchmark Classifier	81	9.14
		N
Bab 6		Bab 10
Indexing dan Searching Tingkat Lanjut	86	Topik K
6.1 Distributed Indexing dan IR.....	86	10.1 Pe
6.2 Hadoop	89	10.2 Pe
Bab 7		
Web Retrieval dan Crawling	94	
7.1 Pendahuluan.....	94	
Bab 8		
Multimedia Retrieval	108	
8.1 Multimedia dan Multimedia Retrieval.....	108	

	Bab 9	
8	Information Extraction dan Data Mining	113
8	9.1 Information Extraction Architecture[3]	113
9	9.2 Chunking	115
60	9.3 Chunking dengan Regular Expressions	118
61	9.4 Eksplorasi Teks Corpora.....	119
	9.5 Representasi Chunks: Tags vs Trees.....	121
63	9.6 Developing and Evaluating Chunkers	122
63	9.7 Simple Evaluation and Baselines	124
66	9.8 Training Classifier-Based Chunkers	128
67	9.9 Named Entity Recognition.....	136
68	9.10 Data mining	139
69	9.11 Regresion	140
71	9.12 Latent Semantic Analysis (LSA).....	141
72	9.13 143	
81	9.14 Networks Mining	144
	Bab 10	
86	Topik Khusus: Advanced XML Retrieval	147
86	10.1 Pengantar XML	147
89	10.2 Pemrograman XML.....	148

Bab 1

Pengenalan Knowledge dan *Information Retrieval*

Tujuan Instruksional Umum :

1. Mahasiswa mampu menjelaskan definisi dan bagian penting dari knowledge, big data dan Information Retrieval

Tujuan Instruksional Khusus :

1. Mahasiswa dapat menyebutkan definisi dari knowledge dan Information Retrieval
2. Mahasiswa mengenal mengenai Boolean retrieval dan vector space model dan membuat program.
3. Mahasiswa mengenal istilah index, inverted index, posting list dan dictionaries
4. Mahasiswa mampu melakukan pemrograman Python untuk IR

1.1 Definisi Knowledge dan *Information Retrieval*

Pada era 1990an, umumnya orang memperoleh informasi melalui informasi dari orang lain. Seiring berjalannya waktu, dengan kehadiran *search engine* seperti Google yang fantastis, pada tahun 2004 *Pew Internet Survey* menemukan 92% orang menggunakan internet sebagai sumber knowledge dan informasi sehari-hari.

Pengetahuan (*knowledge*) merupakan gabungan dari pengalaman, nilai, informasi kontekstual, pandangan pakar dan intuisi mendasar yang memberikan suatu lingkungan dan kerangka untuk mengevaluasi dan menyatukan pengalaman baru dengan informasi (Thomas Davenport, 1998). Drucker (1998) mendefinisikan pengetahuan sebagai informasi yang mengubah sesuatu atau seseorang. Hal ini terjadi karena informasi tersebut menjadi dasar seseorang untuk bertindak, dimana pengetahuan tersebut akan memungkinkan seseorang atau institusi untuk mengambil tindakan yang berbeda atau tindakan yang lebih efektif dibandingkan tindakan seseorang yang tidak memiliki pengetahuan.

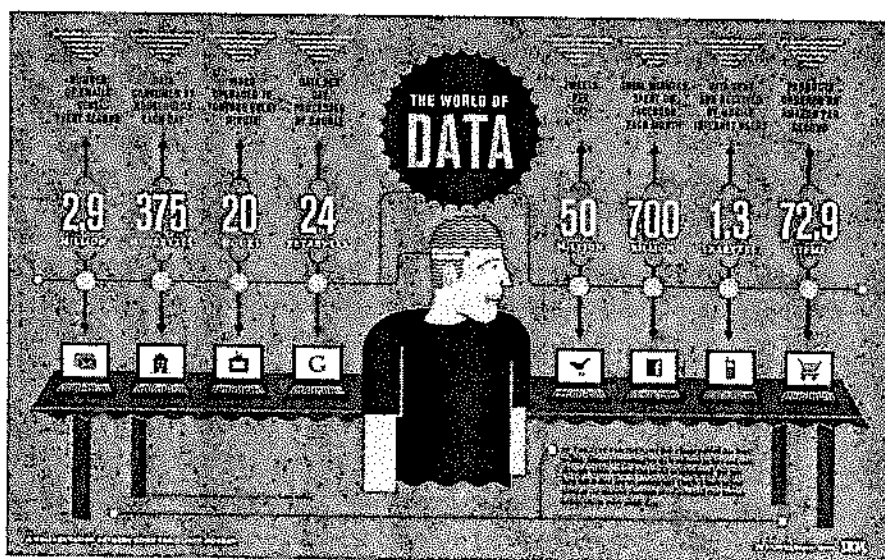
Data Mining adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini

tidak diketahui secara manual. Namun, di berbagai organisasi, terminologi “data science”, “big data”, dan “Hadoop” seakan sudah menjadi setali tiga uang. Kalau melihat dari trend, data science adalah sebuah terminologi yang mulai ngetrend di tahun 2013 ketika Hadoop dan big data sudah menjadi buzzword di berbagai organisasi. Definisi Big data dari Gartner:

“Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”

Definisi Big data dari Teradata dan Hortonworks:

“Big Data adalah gerakan atau inisiatif organisasi-organisasi untuk mengambil, menyimpan, memproses, dan menganalisa data-data yang sebelumnya tidak memungkinkan atau tidak ekonomis untuk diambil, disimpan, diproses, dan dianalisa.”



Gambar 1.1 Big Data[9]

Salah satu permasalahan yang big data coba pecahkan adalah meledaknya volume data yang suatu organisasi ingin simpan atau proses. Salah satu perusahaan telekomunikasi yang pernah saya tangani butuh menyimpan lebih dari 1 milyar record data aktivitas browsing internet pengguna. Permasalahan yang kedua: data velocity atau kecepatan data dibuat. Bisa dibilang, permasalahan ini berkaitan erat dengan

pern
berbi
jumlah
yang
platf
permi
file da

diana
memo
dikem
adalah
scienti
prosed
beruba
Web(V
ekonon
Retrie

Inform
an uns
f

M
retrieval

Informa
of, an
online
objects.
should b

P.
pada kom
pada kon
memprose
untuk mei

permasalahan volume data, karena kecepatan data dibuat umumnya berbanding lurus dengan volume data. Data tidak hanya datang dalam jumlah besar, tetapi juga dalam tempo yang lebih singkat dan bahkan ada yang real-time. Permasalahan ini akan sangat sulit dipecahkan oleh data platform tradisional, baik itu database atau data warehouse. Variety adalah permasalahan yang terjadi karena keberagaman data, baik itu dari format file data yang masuk, maupun format / struktur dari isi data tersebut.

Informasi maupun fakta yang situasi terhadap suatu subjek diamati, dianalisa dan dipelajari menjadi sesuatu yang diingat dalam memori seseorang dan dijadikan sumber pemecahan suatu masalah dikemudian hari. Menurut Rajendra (2005), 3 sumber Utama knowledge adalah literature, expert dan contoh, sedangkan 3 basis knowledge adalah scientific laws, experience and models. Pengetahuan terdiri dari fakta, prosedur dan judgemental rules. Information Retrieval(IR) telah banyak berubah dalam beberapa tahun terakhir dengan adanya ekspansi Web(WWW) dan munculnya antarmuka pengguna grafis modern dan ekonomis dan perangkat penyimpanan massal. Definsi dari *Information Retrieval* menurut Manning(2009) adalah:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

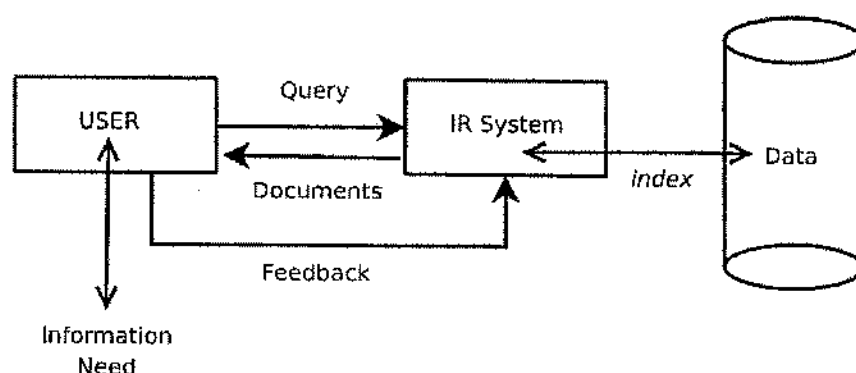
Menurut Richardo Baeza dan Berthier Ribeiro (2011), Information retrieval dapat didefinisikan sebagai:

Information retrieval deals with the representation, storage, organization of, and access to information items such as documents, Web pages, online catalog, sctructured and semi-structured records, multimedia objects. The representation and organization of the information items should be such as to provide the users with easy access to information of their interest.

Pada tataran penelitian, ada 2 pandangan yaitu padangan terpusat pada komputer dan terpusat pada sisi manusia. Pada pandangan terpusat pada komputer, IR terdiri dari pengembangan indeks yang efisien, memproses query user dengan kecepatan tinggi dan algoritma ranking untuk meingkatkan hasil. Pada padangan perpusat pada sisi manusia, IR

terdiri dari studi perilaku user, pemahaman kebutuhan manusia dan menentukan pemahaman tersebut mempengaruhi organisasi dan operasi dari sistem retrieval.

IR juga dapat mencakup jenis lain dari data dan informasi di luar yang ditentukan di atas. Istilah "unstructured" mengacu pada data yang tidak jelas dan semantically overt. Ini adalah kebalikan dari data terstruktur, contohnya database relasional yang digunakan pada perusahaan untuk menangani database persediaan produk. IR juga digunakan untuk memfasilitasi pencarian "semistructured" seperti mencari dokumen dimana judul berisi *AI* dan body berisi *Natural Language Processing*. Gambar 1.2 menampilkan sistem Information retrieval yang sangat bergantung pada *information need* dan sistem index.



Gambar 1.2 Sistem Information Retrieval

1.2 Inverted Index

Inverted index (inverted file) merupakan struktur data utama pada sistem IR. Inverted index menyediakan pemetaan antara term dan lokasi occurencenya pada sebuah koleksi teks. Cara untuk menghindari memindai teks secara linear untuk setiap query ialah membuat indeks dokumen. Index berisi daftar term-term yang ada di dokumen dengan persamaan matematis:

$$index : doc_i \mapsto \{U_j keyword_j\}$$

Index yang memetakan keyword ke daftar dokumen disebut **inverted index**, dimana sekumpulan term /keyword disebut **dictionary**. Daftar dokumen yang diasosiasikan dengan keyword yang diberikan disebut **posting list**.

ja dan
operasi

masi di
ta yang
struktur,
n untuk
untuk
dimana
nbar 1.2
ng pada



ata

ma pada
in lokasi
ghindari
t indeks
dengan

disebut
ctionary.
diberikan

$$index' : keyword_j \mapsto \{U_i doc_i\}$$

Contoh inverted index:

Dictionary posting list

Documents



Words
Split
Normalize
Cleanup
(StopWords)

Inverted Index

Term	Document Locations
roaded	{'doc1': [221]}
coach	{'doc3': [12]}
house	{'doc2': [240]}
singletary	{'doc1': [23, 206, 342]}
mandate	{'doc3': [143]}
innovations	{'doc2': [79]}
edition	{'doc2': [10]}
miners	{'doc1': [0]}
week	{'doc2': [140], 'doc1': [172, 156]}
buildings	{'doc3': [204]}
energy	{'doc2': [410]}
football	{'doc1': [326]}
coast	{'doc2': [26]}
job	{'doc1': [256]}
one	{'doc3': [234], 'doc1': [266]}
green	{'doc3': [32, 65]}
team	{'doc1': [335]}
singletary	{'doc1': [23, 206, 342]}
topics	{'doc2': [307]}
smith	{'doc3': [49, 153, 401]}

Jadi **Inverted index** adalah index yang memetakan balik term ke dokumen. Kita menggunakan dictionary untuk struktur data dan vocabulary/lexicon untuk himpunan term. Kumpulan dari dokumen yang dikoleksikan disebut **corpus**(jamak: corpora). Pada Shakespeare Collected Works yang memiliki 32.000 kata berbeda, menghasilkan term-document incidence matrix. Term adalah unit berindex yang berisi kata-kata, berikut contohnya:

Tujuan kita adalah untuk mengembangkan sebuah sistem untuk mengatasi tugas AD HOC RETRIEVAL . ini adalah tugas IR yang paling standar. Di dalamnya, sistem bertujuan untuk memberikan dokumen dari dalam koleksi yang relevan dengan kebutuhan informasi yang diinginkan pengguna, dikomunikasikan ke sistem dengan dimulai oleh query pengguna. **Information need** merupakan topik dimana user ingin tahu lebih jauh, berbeda dengan **query** dimana user memerintahkan computer untuk berkomunikasi untuk memperoleh informasi yang diinginkan/

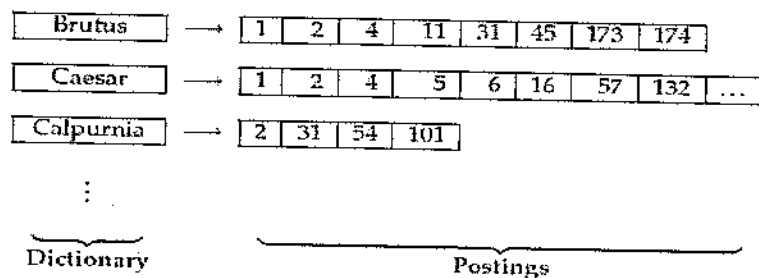
Untuk menguji efektifitas system IR (kualitas hasil pencarian), user biasanya menggunakan 2 kunci statistik:

- **Precision:** Berapa fraksi dari hasil yang dimunculkan relevan dengan informasi yang dibutuhkan

- **Recall:** berapa fraksi dari dokumen relevan pada koleksi dihasilkan oleh sistem. Untuk mendapatkan manfaat kecepatan pengindeksan pada saat pengambilan, kita harus membangun Indeks di muka.

Langkah-langkah utama dalam hal ini adalah:

1. Tentukan dokumen yang akan diindex.
Friends romans, countrymen. So let it be with Caesar ...
2. Tokenize teks, tiap dokumen menjadi token
Friends| romans| countrymen | ...
3. Buat dictionary dan posting list



Gambar 1.3. 2 bagian dari inverted index. Dictionary disimpan di memori, dengan pointer untuk tiap posting list yang tersimpan di disk.

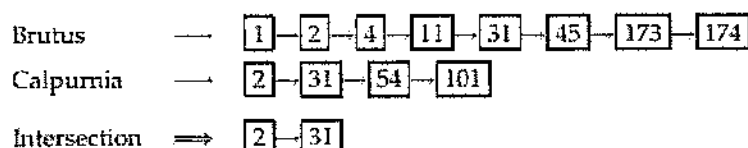
4. Melakukan preprosesing linguistik, menghasilkan token: friend roman countryman so ...
5. Mengindex dokumen dimana tiap term terjadi dengan membuat inverted index.

Dalam kumpulan dokumen, tiap dokumen memiliki serial number unik dikenal sebagai document identifier (docID). Selama index construction, kita dapat memberikan nilai integer untuk tiap dokumen baru yang ditemukan. Langkah berikutnya pada indexing ialah mengurutan sehingga term alfabetis. Dictionary juga merekam statistic seperti jumlah dokumen yang berisi tiap term (document frequency) yang berguna pada *search engine* pada saat query.

Jika kita melakukan query: Brutus AND Calpurnia, maka langkahnya:

1. Cari Brutus pada dictionary
2. Ambil posting tersebut
3. Cari Calpurnia pada dictionary
4. Ambil posting tersebut

5. Intersect 2 posting list seperti gambar di bawah



Gambar 1.4 Interseksi posting list untuk Brutus and Calpurnia dari gambar 1.2

Interseksi sangat penting, kita harus interseksi posting list untuk dapat mencari dokumen berisi kedua term tersebut(dikenal sebagai merging posting list, dapat menggunakan algoritma *merge* berikut:

```

INTERSECT( $p_1, p_2$ )
1   $answer \leftarrow \{ \}$ 
2  while  $p_1 \neq NIL$  and  $p_2 \neq NIL$ 
3  do if  $docID(p_1) = docID(p_2)$ 
4     then  $ADD(answer, docID(p_1))$ 
5          $p_1 \leftarrow next(p_1)$ 
6          $p_2 \leftarrow next(p_2)$ 
7  else if  $docID(p_1) < docID(p_2)$ 
8     then  $p_1 \leftarrow next(p_1)$ 
9     else  $p_2 \leftarrow next(p_2)$ 
10 return  $answer$ 
  
```

Gambar 1.5 Algoritma untuk interseksi dari 2 posting lists p_1 dan p_2 .

1.3 Boolean retrieval

Sebelum melakukan searching informasi pada browser, user memiliki *information need* topik. Sebagai hasil dari *information need*, user membuat query ke system IR, yang umumnya terdiri dari 2-3 term. Kita menggunakan istilah term daripada “kata” karena query dengan term pada kenyataannya bukan sebuah kata(misalnya tanggal, bilangan, note music ata frase). Operator wildcard and operator kesesuaian parsial juga diizinkan pada query term. Sebagai contoh, term “inform*” berisi semua kata yang sesuai dimulai dari “inform”, “Informs” “informative” dan lainnya.

Sebuah buku Novel terkenal, salah satunya yaitukaryaShakespeare yang berisi sekitar 1 juta kata masih dapat dicari dengan cepat menggunakan PC saat ini, namun menjadi komplek jika data yang dicari dari seluruh file di Web server, disinilah IR berfungsi. Misalnya, Anda ingin mencari bagian drama mana pada Shakespeare yang

mengandungkata Brutus AND Caesar AND NOT Calpurnia. Salah satu cara untukmelakukannya adalah memulai diawal dan untuk membaca semua teks, mencatat setiap karya apakah mengandung Brutus dan Caesar dan tidak mengandungCalpurnia.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Gambar 1.6 Sebuah term-document incidence matrix. Elemen matrik (t, d) adalah 1 jika play di kolom d berisi word di baris t, selain itu bernilai 0.

Untuk menjawab pertanyaan Brutus AND Caesar AND NOT Calpurnia, kita mengambilvektor untuk Brutus, Caesar dan Calpurnia, melengkapi yang terakhir, dan kemudian melakukanbitwise AND:

110100 DAN 110111 DAN 101111 = 100100

Dari output di atas, jawaban untuk pertanyaan ini jelasAntony dan Cleopatra dan Hamlet seperti gambar di bawah:

Antony and Cleopatra, Act III, Scene ii
Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii
Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

Gambar 1.7 Hasil dari Shakespeare untuk query Brutus AND Caesar AND NOT Calpurnia.

1.4 Tokenization

Tokenization memberikan urutan karakter dan sebuah unit dokumen terdefinisi. Tokenization merupakan tugas memecah kalimat menjadi bagian-bagian yang disebut token dan menghilangkan bagian tertentu seperti tanda baca.

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

Friends	Romans	Countrymen	lend	me	your	ears
---------	--------	------------	------	----	------	------

Gambar 1.8 Diberikan sebuah urutan karakter/kata, tokenisasi akan memecah menjadi tiap bagian yang disebut token

Boolean retrieval model berbeda dengan ranked retrieval model seperti vector space model dimana user menggunakan free text queries yaitu menulis 1 atau lebih kata daripada menggunakan bahasa tertentu dengan operator untuk membuat ekspresi query. Proximity operator digunakan untuk menentukan 2 term pada query yang harus ada berdekatan pada dokumen tersebut.

1.5 Stemming and lemmatization

Untuk memperoleh urutan karakter pada dokumen, perlu proses encoding menggunakan Unicode UTF-8 misalnya. Stemming merefer pada proses heuristic yang mengambil akhir kata. *Lemmatization* merefer pada melakukan sesuatu menggunakan vocabulary dan analisis morfologis kata-kata, untuk menghilangkan inflectional endings only dan untuk mengembalikan bentuk dictionary dari sebuah kata yang dikenal sebagai *lemma*. Algoritma untuk stemming English yang terkenal adalah Porter's algorithm (Porter 1980) yang terdiri dari 5 fase reduksi kata secara sekuensial

Rule		Example
SSES	→ SS	caresses → caress
IES	→ I	ponies → poni
SS	→ SS	caress → caress
S	→	cats → cat

Banyak aturan berikutnya menggunakan konsep pengukuran sebuah word, sebagai contoh:

($m > 1$) EMENT \rightarrow akan memetakan replacement ke replac, namun bukan cement to c.

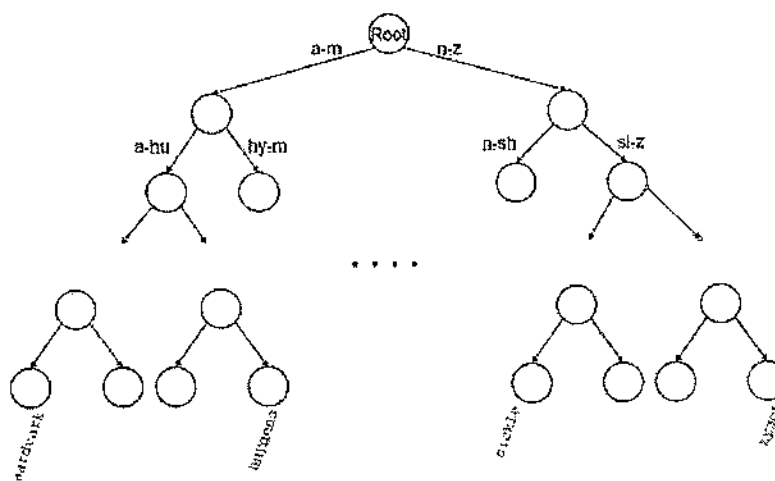
Situs resmi untuk Porter Stemmer is:

<http://www.tartarus.org/~martin/PorterStemmer/>

Sebuah **biword index** tidak hanya solusi standard. Selain itu, positional index yang paling digunakan. Jadi untuk tiap term di vocabulary, kita menyimpan postings dari bentuk docID: position1, position 2, . . . i. Anda juga dapat menggunakan **Lemmatizer**, sebuah tool dari NLP dimana melakukan analisis morfologis untuk secara akurat mengidentifikasi lemma untuk tiap word.

1.6 Dictionaries

Diberikan inverted index dan query, tugas kita menentukan apakah tiap query term ada pada vocabulary dan jika begitu, identifikasi pointer ke posting yang sesuai. Operasi lookup Vocabulary ini menggunakan struktur data klasik yang memanggil *dictionary* dan memiliki 2 solusi kelas yaitu **hashing**, dan **search trees**. Search tree mengizinkan kita mencari seluruh vocabulary terms dimual dengan automat dimana search tree terbaik yaitu binary tree yang nod einternal memiliki 2 anak.



Gambar 1.9 Sebuah binary search tree. Cabang dari root mempartisi term vocabulary menjadi 2 subtrees

Kita fokus pada 2 bentuk spelling correction yaitu *isolated-term correction* dan *context-sensitive correction*. Untuk *spelling correction*,

misalnya kesalan penuisa carrot menjadi carot, dikenal 2 metode yaitu *edit distance* dan *k-gram overlap*. Diberikan 2 karakter string s_1 dan s_2 , edit distance diantaranya ialah jumlah minimum operasi edit yang dibutuhkan untuk transformasi s_1 ke s_2 , dikenal juga sebagai *levenshtein distance*. Sel $[i, j]$ memiliki 4 entri sebagai sel 2×2 .

```

EDITDISTANCE( $s_1, s_2$ )
1  int  $m[i, j] = 0$ 
2  for  $i \leftarrow 1$  to  $|s_1|$ 
3  do  $m[i, 0] = i$ 
4  for  $j \leftarrow 1$  to  $|s_2|$ 
5  do  $m[0, j] = j$ 
6  for  $i \leftarrow 1$  to  $|s_1|$ 
7  do for  $j \leftarrow 1$  to  $|s_2|$ 
8      do  $m[i, j] = \min\{m[i-1, j-1] + \text{if } (s_1[i] = s_2[j]) \text{ then } 0 \text{ else } 1, \text{ if } (s_1[i] \neq s_2[j]) \text{ then } 1, m[i-1, j] + 1, m[i, j-1] + 1\}$ 
9
10
11 return  $m[|s_1|, |s_2|]$ 

```

Gambar 1.9 Algoritma Dynamic programming untuk komputasi edit distance diantara string s_1 dan s_2

1.7 Wildcard Queries

Wildcard queries digunakan pada kondisi:

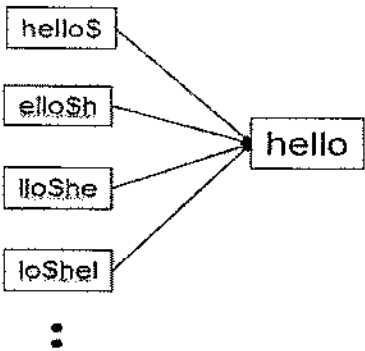
- (1) user tidak yakin pada penulisan sebuah query term (misalnya Sydney vs. Sidney, dimana menghasilkan wildcard query S*dney);
- (2) user sadar penggunaan multiple variants dari spelling a term dan mencari document berisi berbagai variant (misalnya color vs. colour);
- (3) user mencari document berisi variants dari sebuah term yang mungkin ditangkap oleh stemming, tetapi tidak yakin apakah search engine melakukan stemming (misalnya judicial vs. judiciary, menghasilkan wildcard query judicia*);

Special index untuk query wildcard umum disebut sebagai **permuterm index**, sebuah bentuk dari **inverted index**. Kita gunakan symbol spesial \$ ke character set kita, untuk menandakan akhir sebuah term. Jadi, term hello is ditunjukkan sebagai augmented term hello\$. Lalu, kita membuat permuterm index, dimana berbagai rotasi tiap term (augmented dengan \$) terhubung ke bentuk vocabulary asal. Himpunan

terms dirotasi pada permuterm index disebut sebagai *permuterm vocabulary*. Dengan stop list, kita mengecualikan kata-kata umum yang:

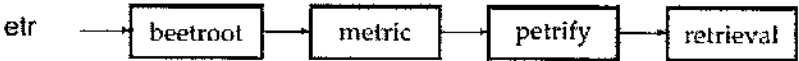
- Memiliki isi semantic sedikit: *the, a, and, to, be*
- Ada sekitar ~30% of postings untuk top 30 kata

Selain itu, kita harus normalisasikan kata pada indexed text, misalnya mencocokkan *U.S.A.* dan *USA*.



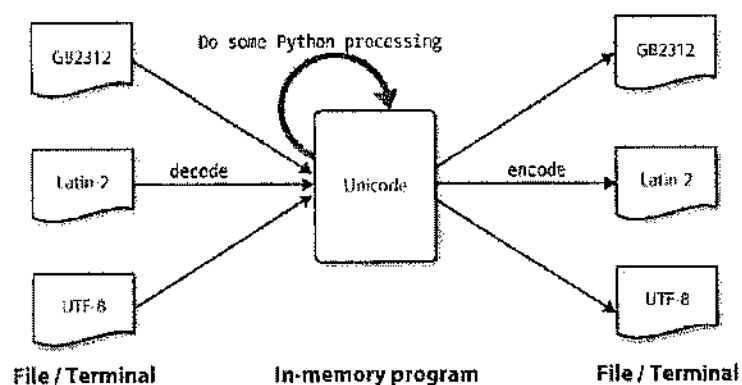
Gambar 1.10 sebuah bagian dari permuterm index

Karena permuterm index simple, maka dapat menyebabkan ledakan rotasi per term. Oleh Karena itu dikenalkan teknik kedua yaitu k-gram index, untuk memroses wildcard queries. Kita menggunakan karakter special \$ untuk melambangkan awal atau akhir term, jadi himpunan 3-grams yang dibuat untuk castle adalah: \$ca, cas, ast, stl, tle, le\$. Pada k-gram index, dictionary berisi seluruh k-grams yang terjadi pada term di vocabulary. Tiap posting list merujuk dari sebuah k-gram ke seluruh vocabulary.



Gambar 1.11 Contoh posting list pada 3-gram index. Di sini digambarkan 3-gram *etr*. Pencocokan term vocabulary secara leksikografis diurut pada posting

Unicode mendukung jutaan karakter. Tiap karakter diberikan sebuah angka yang disebut sebagai **code point** yang ditulis dalam bentuk `\uXXXX`, dimana `XXXX` merupakan 4 digit heksadesimal

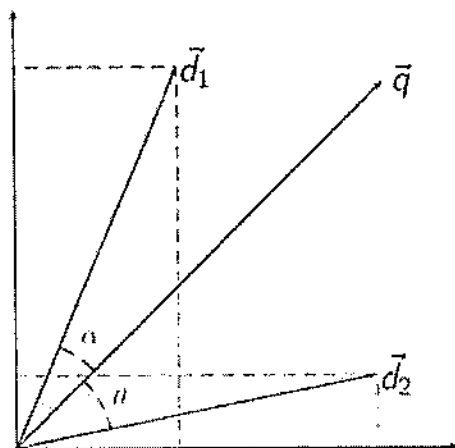


Gambar 1.12 Unicode pada bahasa Python

Text pada file boleh jadi dalam bentuk encoding tertentu, sehingga diperlukan mekanisme translasi ke Unicode yang disebut sebagai decoding. Sebaliknya, untk menulis Unicode ke file atau sebuah terminal, dibutuhkan translasi ke encoding yang sesuai.

1.8 Vector Space Model

Vector space model merupakan model IR yang paling terkenal di awal pengembangan IR oleh Gerald Salton. Vector space model merupakan model aljabar untuk representasi document teks seperti pada term index, digunakan pada ranked retrieval, termasuk document clustering and classification.



Gambar 1.13 Document similarity pada Vector space model.

Cosine dari sudut :

$$\cos \theta = \frac{\mathbf{d}_2 \cdot \mathbf{q}}{\|\mathbf{d}_2\| \|\mathbf{q}\|}$$

Dimana $\mathbf{d}_2 \cdot \mathbf{q}$ merupakan interseksi (dot product) dari dokumen (\mathbf{d}_2 pada gambar) dan query (\mathbf{q} pada gambar) vektor, $\|\mathbf{d}_2\|$ merupakan norm dari vector \mathbf{d}_2 , dan $\|\mathbf{q}\|$ merupakan norm vector \mathbf{q} . Norm vector dihitung sebagai:

$$\|\mathbf{q}\| = \sqrt{\sum_{i=1}^n q_i^2}$$

Terdapat model term frequency-inverse document frequency, dimana vector bobot untuk dokumen d adalah

$$\mathbf{v}_d = [w_{1,d}, w_{2,d}, \dots, w_{N,d}]^T, \text{ dimana}$$

$$w_{t,d} = \text{tf}_{t,d} \cdot \log \frac{|D|}{|\{d' \in D \mid t \in d'\}|}$$

dan

- $\text{tf}_{t,d}$ adalah term frequency dari term t di dokumen d (local parameter)
- $\log \frac{|D|}{|\{d' \in D \mid t \in d'\}|}$ merupakan inverse document frequency (global parameter). $|D|$ adalah total jumlah dokumen ; document set; $|\{d' \in D \mid t \in d'\}|$ adalah jumlah dokumen t term t .

Menggunakan cosine similarity diantara dokumen d_j dan query q dihitung:

$$\text{sim}(d_j, q) = \frac{\mathbf{d}_j \cdot \mathbf{q}}{\|\mathbf{d}_j\| \|\mathbf{q}\|} = \frac{\sum_{i=1}^N w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^N w_{i,j}^2} \sqrt{\sum_{i=1}^N w_{i,q}^2}}$$

1.9 Implementasi

Untuk mempelajari Boolean Retrieval dapat menggunakan R, Python dan C#, yang didesain khusus untuk dapat mendukung AI. Berikut contohnya:

Doc1= English tutorial and fast track

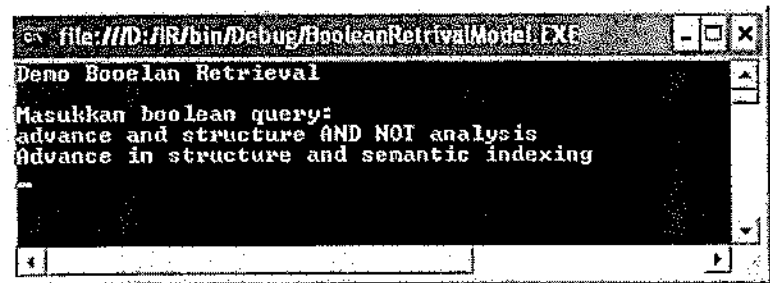
Doc2 = learning latent semantic indexing

Doc3 = Book on semantic indexing
 Doc4 = Advance in structure and semantic indexing
 Doc5 = Analysis of latent structures

Query problem: advance and structure AND NOT analysis

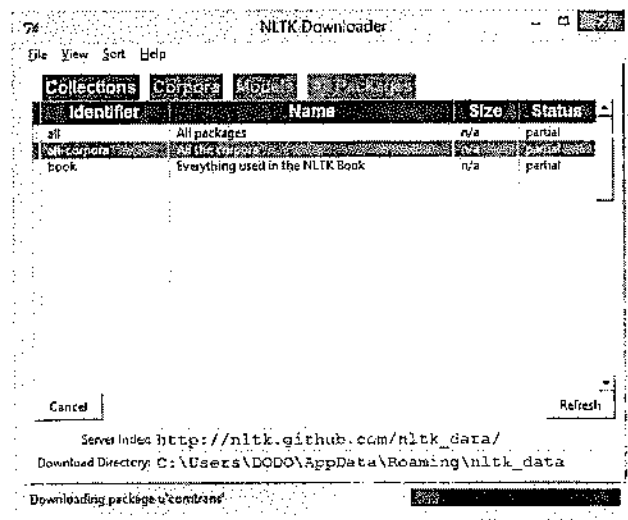
Terms/doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc1	Doc2	Doc3	Doc4	Doc5	
English	0	0	0	0	0	0	0	0	1	0	
Tutorial	0	0	0	0	0	0	0	0	1	1	(AND)
Fast	0	0	0	0	0	0	0	0	1	0	
Track	0	0	0	0	0	1	0	1	1	0	(NOT analysis)
Books	0	0	0	0	0	0	0	0	1	0	
Semantic	0	0	0	0	0	0	0	0	1	0	
Analysis	0	0	0	0	0	0	0	0	1	0	
Learning	0	0	0	0	0	0	0	0	1	0	
Latent	0	0	0	0	0	0	0	0	1	0	
Indexing	0	0	0	0	0	0	0	0	1	0	
Advance	0	0	0	0	0	0	0	0	1	0	
Structures	0	0	0	0	0	0	0	0	1	0	

Hence Doc4 is retrieved here.



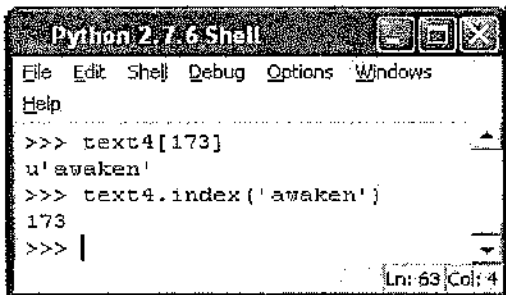
Gambar 1.14 Hasil

Anda harus mengunduh python di Python.org , serta meload paket tambahan numpy dan NLTK, serta load datanya:



Gambar 1.15 NLTK Data downloading

Untuk melihat index pada text4, digunakan fungsi index

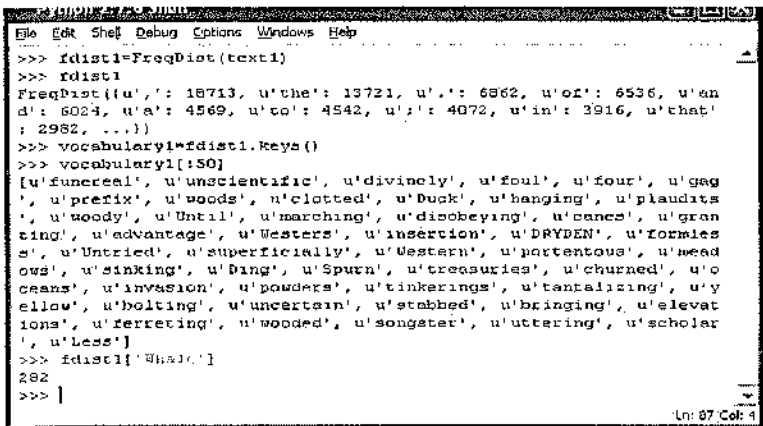


Gambar 1.19 index function

Untuk melihat distribusi frekwensi dari suatu kata serta melihat isi dictionary seperti pada gambar berikut:

Word Tally

the	
been	
message	
persevere	
nation	



Gambar 1.20 Distribusi frekwensi dan fungsi frekwensi vocabulary

Untuk menghitung panjang kata, dan melihat isi file corpus, file identifiers in this corpus:

```

Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
>>>
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
[u'austen-emma.txt', u'austen-persuasion.txt', u'austen-sense.txt',
u'bible-kjv.txt', u'blake-poems.txt', u'bryant-stories.txt', u'burgess-busterbrown.txt', u'carroll-alice.txt', u'chesterton-ball.txt', u'chesterton-brown.txt', u'chesterton-thursday.txt', u'edgeworth-parents.txt', u'melville-moby_dick.txt', u'milton-paradise.txt', u'shakespeare-caesar.txt', u'shakespeare-hamlet.txt', u'shakespeare-macbeth.txt', u'whitman-leaves.txt']
>>> emma=nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
>>> [

```

Gambar 1.2 1 fileids pada corpus

Untuk proses Stemming, dapat menggunakan fungsi PorterStemmer()

```

Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
SyntaxError: Invalid Syntax
>>> porter=nltk.PorterStemmer()
>>> lancaster=nltk.LancasterStemmer()
>>> [porter.stem(t) for t in tokens]
[u'DENNI', u'', u'Listen', u'', u'strang', u'women', u'lie', u'in', u'pond', u'distribut', u'sword', u'is', u'no', u'basi', u'for', u'a', u'system', u'of', u'govern', u'', u'Suprem', u'execut', u'po', u'ver', u'deriv', u'from', u'a', u'mandat', u'from', u'the', u'mass', u'', u'not', u'from', u'some', u'farcic', u'aquat', u'ceremoni', u'']

```

Gambar 1.22 PorterStemmer

Untuk menghapus stopwords:

Removings stopwords.py:

```

import nltk
from nltk.collocations import *
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

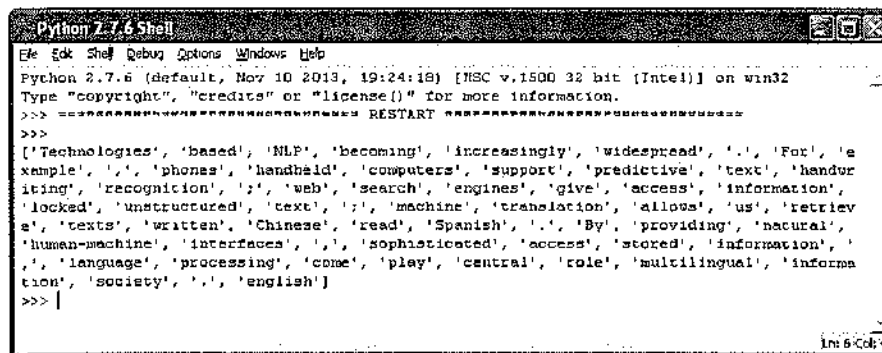
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
stopset = set(stopwords.words('english'))
with open('sentiment_text.txt', 'r') as text_file:

```

```

txt = text_file.read()
tokens=word_tokenize(str(text))
tokens = [w for w in tokens if not w in stopset]
print tokens

```

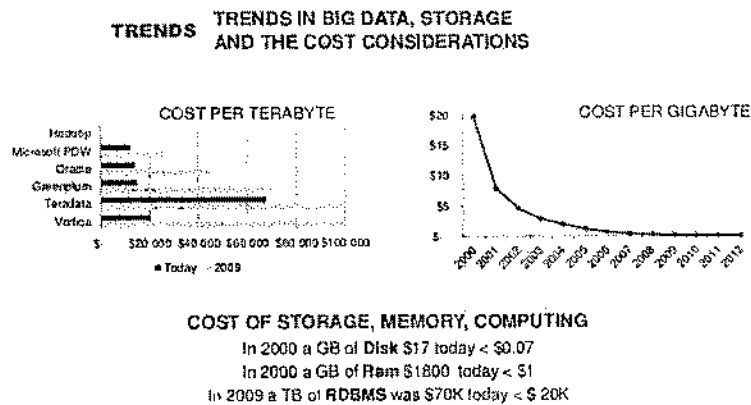


Gambar 1.23 Menghapus stopwords

Akses efisien untuk struktur inverted index merupakan aspek kunci untuk mesin pencari untuk mencapai waktu respon yang cepat bagi pengguna. Sementara kinerja suatu pencarian informasi (IR) sistem dapat ditingkatkan melalui kompresi daftar postingannya, riset terbaru di literatur yang membandingkan dan menganalisis kinerja skema kompresi bilangan bulat yang modern di berbagai jenis posting informasi (document ids, frequencies, positions) dapat dilihat di [5].

Latihan:

1. Jelaskan mengenai trend Big Data seperti ditunjukkan pada gambar berikut:



Gambar 1.24 Trend pada Big Data

2. Buatlah program inveted index dengan kemampuan searching menggunakan Python
3. Buat program "Simple Search Engine" menggunakan corpus terkenal seperti REUTERS dan menggunakan 2 metode perbandingan (Boolean retrieval dan Vector Space model). Buatlah Laporan teknis dari percobaan tersebut (3-4 halaman).

Referensi

1. Christopher D. Manning, Prabhakar Raghavan and Hinrich Scutze (2008), *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 978-0521865715.
2. Ricardo Baeza-Yates and Berthier ribeiro-Neto (2011), *Modern Information Retrieval*, 2nd edition, ACM Books Press. ISBN: 978-0-321-41691-9.
3. Stefan Butcher, Charles L.A Clarke and Gordon V. Cormack (2010), *Information Retrieval- Implementing and Evaluating Search Engines*, MIT Press, ISBN: 978-0-262-02651-2.
4. EMC Education Services,(2015), *Data Science and Big Data Analytics*, Wiley & Sons Publisher, ISBN: 978-1-118-87613-8.
5. Matteo Catena,Craig Macdonald and Iadh Ounis (2014), *On Inverted Index Compression for Search Engine Efficiency*, Advances in Information Retrieval
6. Lecture Notes in Computer Science Volume 8416, 2014, pp 359-371
7. Rajendra Akerkar (2005), *Introduction to Artifical Intelligence*, Prentice Hall India.
8. <http://th30z.blogspot.com/2010/10/python-inverted-index-for-dummies.html>
9. <http://datascience.or.id>
10. Python.org

Bab 2

Index Construction dan Compression

Tujuan Instruksional Umum :

1. Mahasiswa mampu menjelaskan definisi dan bagian penting dari Index Construction, query dan Compression

Tujuan Instruksional Khusus :

1. Mahasiswa dapat menyebutkan definisi dari Index construction dan compression
2. Mahasiswa mengenal definisi dari blocked sort-based dan dynamic indexing
3. Mahasiswa mampu menjelaskan mengenai tolerant retrieval
4. Mahasiswa mengenal berbagai metode index compression dan membuat aplikasi Search Engine

2.1 Pendahuluan

Pada bab 1 sudah dibahas mengenai struktur data, antara lain:

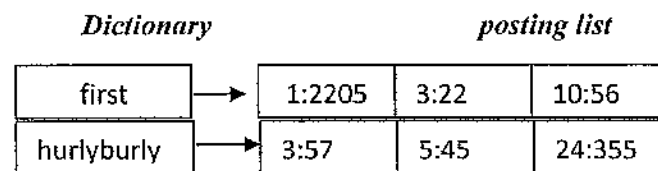
1. **Structured data:** data yang berisi tipe data yang telah didefinisikan, terformat seperti OLAP, RDBMS, CSV files dan spreadsheet
2. **Semi-Structured data:** data yang memungkinkan parsing seperti XML
3. **Quasi-structured data:** data yang membutuhkan effort untuk memformatnya. Contoh hyperlink
4. **Unstructured data:** data yang tidak memiliki struktur iheren, seperti PDF files, image dan video

Pada bab ini, kita akan membuat kompresi pada index, mengingat dokumen berukuran sangat besar. Dokumen berbahasa Inggris, cukup membutuhkan 7 bit nilai ASCII, namun tidak halnya untuk bahasa lainnya. Oleh karena itu dibutuhkan skema koding, misalnya menggunakan UTF- yang menyediakan 1-4 byte encoding. UTF-8 backward compatible dengan ASCII, sehingga teks ASCII otomatis berformat UTF- Corpus(plural: corpora) merupakan koleksi besar teks yang digunakan untuk eksperimen di NLP. Term Frequency-Inverse Docume

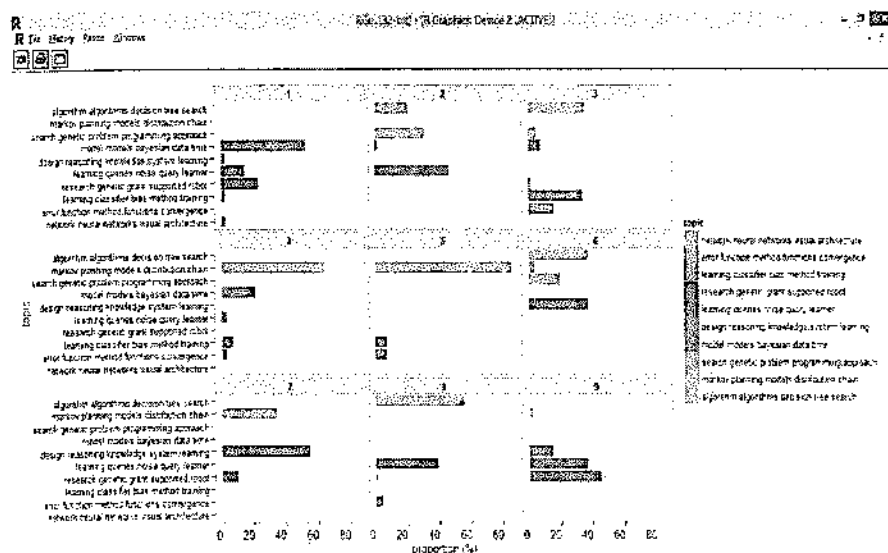
Frequency(TF-IDF) merupakan sebuah cara pengukuran information retrieval dan text analysis. Himpunan dari token yang berbeda atau sibil pada koleksi teks disebut vocabulary, yang dilambangkan dengan v . Koleksi Shakespeare memiliki $v=22.987$ simbol pada vocabularynya.

$$V = \{a, aaron, abaissize, \dots, zounds, \dots\}$$

Pada inverted index berorientasi dokumen, Contoh tiap posting memiliki format docid:within-document-position sebagai berikut:



Yang diperlukan pada searching ialah index yang bagus dan mencari dokumen top- k dan mengembalikan ke user. Berikut contoh hasil analisa teks menggunakan bahasa R



Gambar 2.1 Sentiment analysis menggunakan R

Inverted index (inverted file) merupakan mekanisme untuk indexing sebuah koleksi teks untuk mempercepat tugas pencarian. Struktur inverted index terdiri dari 2 element: vocabulary(lexicon atau dictionary) dan *occurrences*. Vocabulary merupakan himpunan seluruh kata yang berbeda pada teks.

Membuat inverted index dikenal sebagai index construction, dimana mesin atau proses yang melakukan ini disebut indexer. Proses index construction bisa berbagai macam, membaca file, pada web search harus crawled, sedangkan pada enterprise search dokumen umumnya terenkapsulasi pada berbagai content management systems seperti aplikasi email dan database. Indexer mengompresi dan dekompresi file dan final index. Untuk konstruksi index yang lebih efisien, istilah term pada bab 1 diganti menjadi termID, dimana tiap termID merupakan nomor serial yang unik.

Berikut contoh index construction:

Keyword	docID	Keyword	docID	Keyword	docID	freq
I	1	So	2			
did	1	let	2	ambitious	2	1
enect	1	it	2	be	2	1
Julius	1	be	2	Brutus	1	1
Caesar	1	with	2	Brutus	2	1
I	1	Caesar	2	Capitol	1	1
was	1	The	2	Caesar	1	1
killad	1	noble	2	Caesar	2	2
i'	1	Brutus	2			
the	1	hath	2			
Capitol	1	told	2			
Brutus	1	you	2			
killed	1	Caesar	2			
me	1	was	2			
		ambitious	2	...		

Gambar 2.2 Index construction denganmembuat daftar pasangan dan mengurutkannya

Keyword	NumDoc	Totf	docID freq	docID freq
ambitious	1	1	-> 2 1	
be	1	1		
Brutus	2	2	-> 1 1	-> 2 1
Capitol	1	1		
Caesar	2	3	-> 1 1	-> 2 2

Gambar 2.3 Membuat dictionary file dan posting file

Pada saat pengembangan system IR, keputusan didasarkan pada karakteristik hardware komputer. Akses data di memori lebih cepat daripada data ke disk. Kita menyebut teknik untuk menyimpan data yang sering digunakan di memori utama disebut *caching*. Seek time memiliki reraat 5 ms untuk tipikal disk.

Tabel 2.1 Seek time merupakan waktu yang dibutuhkan untuk memposisikan kepala disk

Symbol	Statistic	Value
s	average seek time	$5\text{ ms} = 5 \times 10^{-3}\text{ s}$
b	transfer time per byte	$0.02\text{ }\mu\text{s} = 2 \times 10^{-8}\text{ s}$
	processor's clock rate	10^9 s^{-1}
p	lowlevel operation (e.g., compare & swap a word)	$0.01\text{ }\mu\text{s} = 10^{-8}\text{ s}$
	size of main memory	several GB
	size of disk space	1 TB or more

Pada konteks system IR, efisiensi sistem dapat diukur dari:

1. Indexing time, waktu yang dibutuhkan untuk membuat index
2. Indexing space, ruang yang digunakan selama pembuatan index
3. Index storage, ruang dibutuhkan untuk menyimpan index
4. Query latency, interval waktu antara kedatangan query pada system IR dan pembangkitan jawaban.
5. Query throughput: jumlah rata-rata query yang diproses perdetik, yang dihitung secara langsung dari query latency.

2.2 Blocked sort-based indexing

Memori utama tidak memadai untuk indexing skala besar, dibutuhkan external sorting algorithm untuk meminimalkan jumlah akses disk selama sorting. Salah stu solusi menggunakan blocked sort-based indexing algorithm (BSBI) yang memecah koleksi ke bagian yang berukuran sama dan mengurutkan pasangan termID-docID di memori lalu menyimpan hasil intermediate pada disk dan menggabungkan seluruh hasil intermediate ke index final. Blocked sort-based indexin memiliki property scala yang bagus, namun membutuhkan struktur data untuk memetakan term ke termID. Untuk koleksi yang sangat besar, digunakan *single-pass in-memory indexing* (SPIMI). SPIMI menggunakan term selain termID,

menulis tiap dictionary blok ke disk dan memulai dictionary baru untuk blok berikutnya.

```

BSBINDEXCONSTRUCTION()
1   $n \leftarrow 0$ 
2  while (all documents have not been processed)
3  do  $n \leftarrow n + 1$ 
4      $block \leftarrow \text{PARSENEXTBLOCK}()$ 
5      $\text{BSBI-INVERT}(block)$ 
6      $\text{WRITEBLOCKTODISK}(block, f_n)$ 
7   $\text{MERGEBLOCKS}(f_1, \dots, f_n; f_{merged})$ 

```

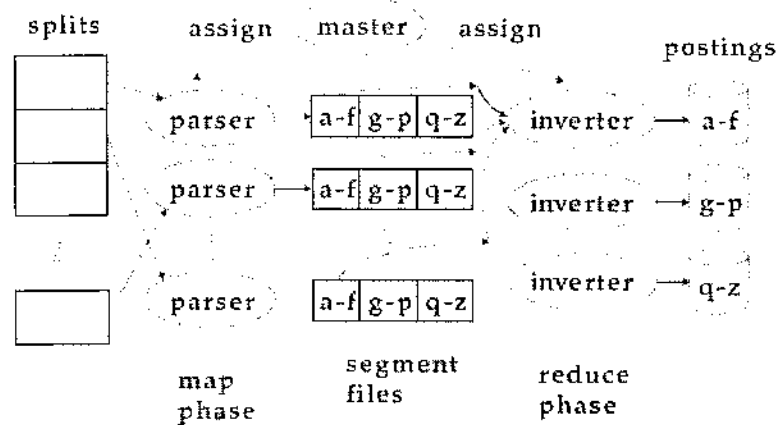
Gambar 2.4 Blocked sort-based indexing. Algoritma ini menyimpan inverted blocks pada file f_1, \dots, f_n dan index yang digabung pada f_{merged}

merged

2.3 Distributed Indexing

Koleksi yang sangat besar membutuhkan lebih dari 1 komputer, misalnya pada WWW. Oleh karena itu web search engine menggunakan algoritma **distributed indexing** untuk index construction. Hasilnya ialah distributed index yang terpartisi pada beberapa mesin. Konstruksi index terdistribusi merupakan sebuah aplikasi dari MapReduce, sebuah arsitektur umum untuk distributed computing. Sebuah master node mengarahkan proses penugasan dan penugasan ulang ke node worker.

MapReduce memecah masalah komputasi ke bagian kecil menggunakan key-value pairs. Untuk indexing, key-value pair memiliki bentuk (termID, docID). Map phase dari MapReduce berisi pemetaan pemecahan data input ke key-value pairs. Sedangkan reduce phase, kita ingin seluruh nilai untuk key yang diberikan disimpan saling berdekatan sehingga dapat dibaca dan diproses dengan cepat. Jika terdapat dokumen baru dan harus dimasukkan dengan cepat, dibutuhkan 2 index; index utama yang besar dan auxiliary index (tersimpan di memory) yang lebih kecil yang menyimpan dokumen baru. Pencarian berjalan pada kedua index serta hasilnya digabungkan.



Gambar 2.5 Contoh distributed indexing dengan MapReduce.

Schema of map and reduce functions

map: Input $\rightarrow \text{list}(k, v)$
 reduce: $(k, \text{list}(v)) \rightarrow \text{Output}$

Instantiation of the schema for index construction

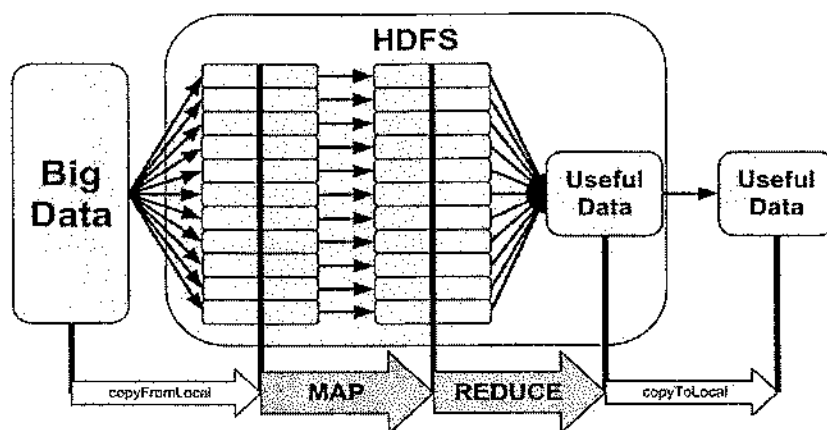
map: web collection $\rightarrow \text{list}(\text{termID}, \text{docID})$
 reduce: $(\langle \text{termID}_1, \text{list}(\text{docID}) \rangle, \langle \text{termID}_2, \text{list}(\text{docID}) \rangle, \dots) \rightarrow (\text{postings_list}_1, \text{postings_list}_2, \dots)$

Example for index construction

map: $d_2 : C \text{ died}, d_1 : C \text{ came}, C \text{ c'ed.} \rightarrow (\langle C, d_2 \rangle, \langle \text{died}, d_2 \rangle, \langle C, d_1 \rangle, \langle \text{came}, d_1 \rangle, \langle C, d_1 \rangle, \langle \text{c'ed}, d_1 \rangle)$
 reduce: $(\langle C, (d_2, d_1, d_1) \rangle, \langle \text{died}, (d_2) \rangle, \langle \text{came}, (d_1) \rangle, \langle \text{c'ed}, (d_1) \rangle) \rightarrow (\langle C, (d_1, 2, d_2, 1) \rangle, \langle \text{died}, (d_2, 1) \rangle, \langle \text{came}, (d_1, 1) \rangle, \langle \text{c'ed}, (d_1, 1) \rangle)$

Gambar 2.5 Map dan fungsi reduce di MapReduce. Fungsi map menghasilkan sebuah daftar pasangan key-value.

HDFS merupakan teknologi menarik yang menyediakan distribusi data, replikasi dan automatic recovery pada user-space filesystem yang mudah untuk konfigurasi, konseptual dan mudah dipahami. Namun, utility dating ketika job map/reduce dieksekusi pada data yang disimpan di HDFS.



Gambar 2.6 HDFS

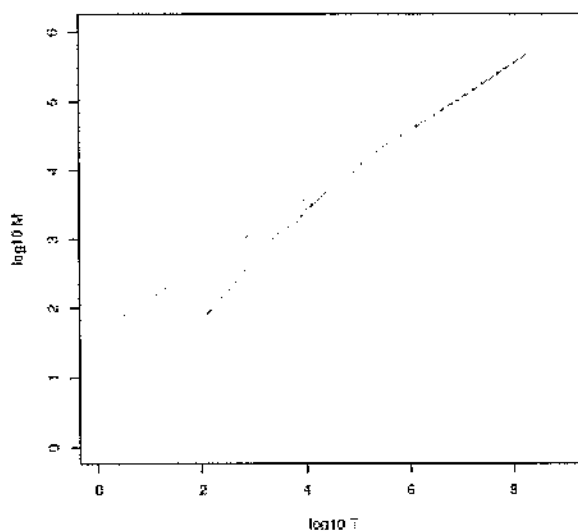
MapReduce memberikan framework simpel dan handal untuk implementasi konstruksi index di lingkungan terdistribusi. Dengan menyediakan metode semiotomatis untuk memecah konstruksi index ke task yang lebih kecil, dapat digunakan pada koleksi yang besar menjadi aplikasi cluster. Big Data merupakan data dimana skala, distribusi diversity dan timeliness membutuhkan penggunaan arsitektur teknis dan analisis untuk memungkinkan pengamatan yang membuka sumber baru bagi nilai bisnis.

“The real issue is not that you are acquiring large amounts of data. It's what you do with the data that counts. The hopeful vision is that organizations will be able to take data from any source, harness relevant data and analyze it to find answers that enable 1) cost reductions, 2) time reductions, 3) new product development and optimized offerings, and 4) smarter business decision making”

2.4 Index Compression

Benefit dari kompresi adalah ruang disk yang lebih sedikit, selain itu mengoptimalkan penggunaan caching, misalnya jika posting list yang sering digunakan pada query term t dilakukan di memory. Teknik kompresi diperoleh optimal dengan lossy compression, dimana menghapus beberapa informasi. Case folding, stemming dan stop word elimination merupakan bentuk dari **lossy compression**. Reuters-RCV1 memiliki 10

juta tokens. Mengumpulkan seluruh termID-docID pairs dari koleksi dengan 4 byte tiap untuk termID dan docID membutuhkan storage 0.8GB.



Gambar 2.6 Hasil Heaps' law. Vocabulary size M merupakan sebuah fungsi dari collection size T (jumlah token) untuk Reuters-RCV1. Garis putus-putus dengan $\log_{10} M = 0.49 * \log_{10} T + 1.64$ merupakan least-squares terbaik. Sehingga, $k = 101.64 \approx 44$ dan $b = 0.49$.

Cara terbaik untuk menangani M ialah menggunakan Heaps' law, yang mengestimasi ukuran vocabulary sebagai fungsi dari ukuran koleksi:

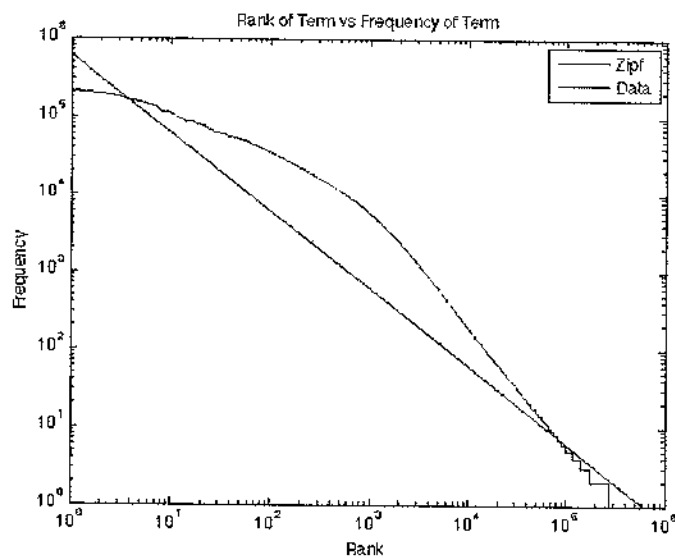
$$M = kT^b$$

Dimana T merupakan jumlah token pada koleksi. Nilai umum k dan b ialah $30 \leq k \leq 100$ and $b \approx 0.5$.

2.5 Zipf's Law

Hubungan antara reqkwendi dan ranking ditampilkan dengan garis dikenal seabgai Zipf's law. Sebuah model distribusi dari term pada koleksi ialah Zipf's law. Jika t_1 merupakan term paling umum pada koleksi, t_2 merupakan term umum berikutnya dan seterusnya, maka frekuensi koleki c_i dari term paling umum ith proposional pada $1/i$;

$$cf_i \propto \frac{1}{i}$$



Gambar 2.7 Grafik log-log plot. Sumbu x merupakan rankging dari term dan sumbu y merupakan frekwensi koleksi dari term tersebut.

2.6 Tolerant Retrieval

Tolerant retrieval dan pemrosesan query pada search engine dapat meningkatkan kehandalan dan kenyamanan pengguna. Berdasarkan perbandingan kesamaan diantara term yang sesuai dan term yang diusulkan oleh user, penelitian yang diusulkan oleh dapat toleran terhadap misspelling dan juga menampilkan saran yang sesuai[5]. Banyak kasus *spelling errors* seperti penulisan carot dan britian spears. Untuk teknik spelling corrections:

1. Pada berbagai alternative pembetulan spelling, pilih “yang terdekat”. Gunakan proximity measures seperti edit distance, k-gram index untuk spelling correction dan context-sensitive spelling correction.
2. Gunakan algoritma untuk memilih kata yang sering muncul. Misal jika kata grunt lebih sering muncul dari pada grant, maka searching grnt harus return grunt.

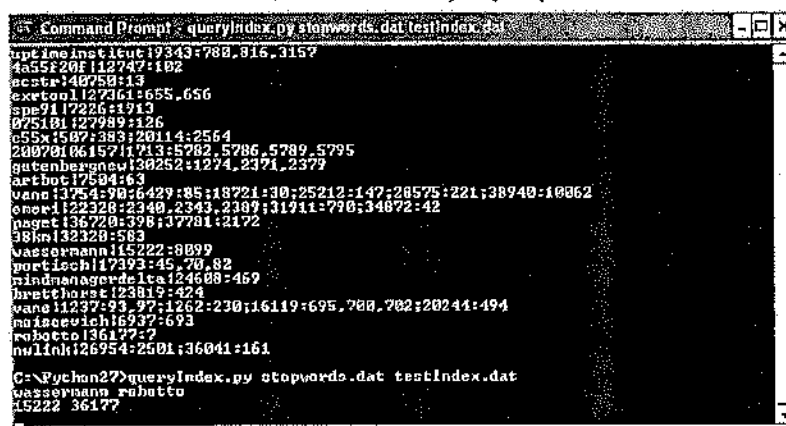
Ada banyak IR system opensource yang dapat digunakan antara lain:

- **Lucene**
Lucene merupakan search system berbasis Java, mengizinkan user mendefinisikan indexing dan rule retrieval dan formula.
- **Indri**
Indri merupakan system retrieval akademi berbasis c++, yang dikembangkan oleh peneliti di University of Massachusetts.
- **Wumpus**
Wumpus merupakan search engine akademi berbasis C++ yang dikembangkan di unviersitas Waterloo.

2.7 Implementasi

2.7.1 Implementasi Python untuk Inverted index

1. Jalankan demo kode pada
<http://www.ardendertat.com/2011/05/30/how-to-implement-a-search-engine-part-1-create-index/>
2. Buatlah index tersebut, lalu lakukan query seperti contoh berikut:



```
Python 2.7.10 Shell: C:\Python27\python.exe
C:\Python27>python queryIndex.py stopwords.dat testIndex.dat
19343:788,816,3157
4a55f20f112747:182
acstr:140758:13
extool127361:655,656
spe9117226:1913
075181127989:126
c55x1587:383;20114:2554
2007010615711713:5782,5786,5789,5795
guenbeagov130252:1274,2371,2377
arfbot17504:63
vanc13754:98;6429:85;18721:30;25212:147;28575:221;38940:10062
enord122328:2340,2343,2307;31911:790;34872:42
psget136720:398;37781:2172
38kn132328:583
wassermann115222:8899
portisch117393:45,70,82
mindnanagerdelca124688:469
bratthost123819:424
vanc11237:93,97;1262:230;16119:695,700,702;20244:494
naisscevic115937:693
robotto136177:7
nwlinh126954:2501;36041:161
C:\Python27>python queryIndex.py stopwords.dat testIndex.dat
wassermann robotto
15222 36177
```

Gambar 2.8 Hasil query untuk Wassermann dan robotto

3. Jalankan query yang menggunakan TF-IDF.

2.7.2 Form GUI

Untuk membuat search engine GUI, dapat mencoba wxPython. wxPython merupakan toolkit GUI yang paling populer untuk Python.

Gui.py:

```
import wx
class ExampleFrame(wx.Frame):
    def __init__(self, parent):
        wx.Frame.__init__(self, parent)

        self.panel = wx.Panel(self)
        self.quote = wx.StaticText(self.panel,
label="Your quote:")
        self.result = wx.StaticText(self.panel,
label="")
        self.result.SetForegroundColour(wx.RED)
        self.button = wx.Button(self.panel,
label="Save")
        self.lblname = wx.StaticText(self.panel,
label="Your name:")
        self.editname = wx.TextCtrl(self.panel,
size=(140, -1))

        # Set sizer for the frame, so we can change
frame size to match widgets
        self.windowSizer = wx.BoxSizer()
        self.windowSizer.Add(self.panel, 1, wx.ALL |
wx.EXPAND)

        # Set sizer for the panel content
        self.sizer = wx.GridBagSizer(5, 5)
        self.sizer.Add(self.quote, (0, 0))
        self.sizer.Add(self.result, (0, 1))
        self.sizer.Add(self.lblname, (1, 0))
        self.sizer.Add(self.editname, (1, 1))
        self.sizer.Add(self.button, (2, 0), (1, 2),
flag=wx.EXPAND)

        # Set simple sizer for a nice border
        self.border = wx.BoxSizer()
        self.border.Add(self.sizer, 1, wx.ALL |
wx.EXPAND, 5)

        # Use the sizers
        self.panel.SetSizerAndFit(self.border)
        self.SetSizerAndFit(self.windowSizer)
```

```

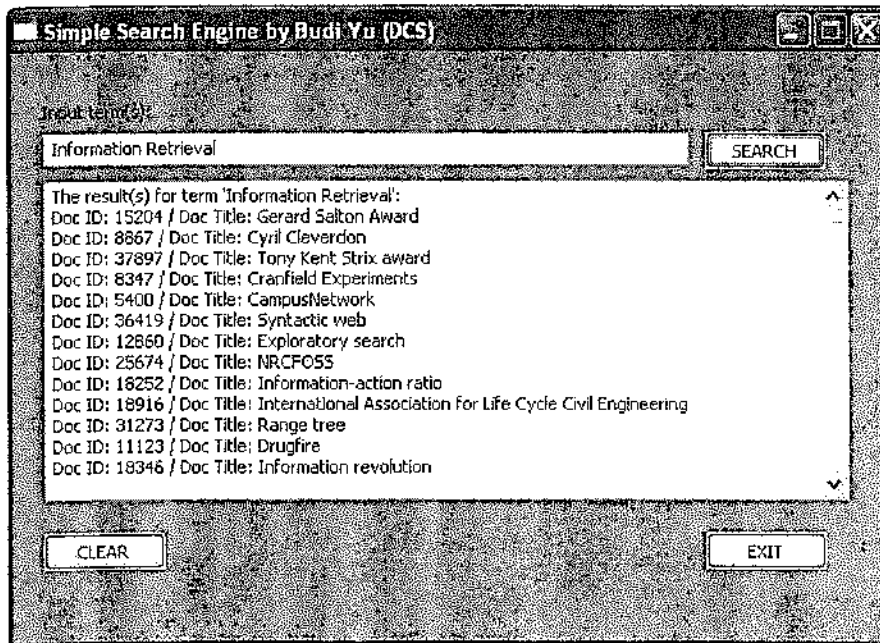
# Set event handlers
self.button.Bind(wx.EVT_BUTTON, self.OnButton)

def OnButton(self, e):
    self.result.SetLabel(self.editname.GetValue())

app = wx.App(False)
frame = ExampleFrame(None)
frame.Show()
app.MainLoop()

```

Berikut contoh penerapan sebagai form Search engine:



Gambar 2.8 Form GUI dengan wxPython

Gui2.py:

```

import wx
class ExampleFrame(wx.Frame):
    def __init__(self, parent):
        wx.Frame.__init__(self, parent, -1, 'BINUS
Search Engine by Dr. Widodo Budiharto')

    # the edit control

```

```

        self.lblname      =      wx.StaticText(self,
label="Your query:", pos=(20,30))
        self.editname     =      wx.TextCtrl(self,
value="Enter query here", pos=(80, 30),
size=(140,-1))
        self.Bind(wx.EVT_TEXT,      self.EvtText,
self.editname)
        self.Bind(wx.EVT_CHAR,      self.EvtChar,
self.editname)

        # A button
        self.button       =wx.Button(self,
label="FIND", pos=(240, 30))
        self.Bind(wx.EVT_BUTTON,
self.OnClick,self.button)

        # A multiline TextCtrl
        self.logger       =      wx.TextCtrl(self,
pos=(20,100), size=(300,200),
style=wx.TE_MULTILINE | wx.TE_READONLY)

        def EvtRadioBox(self, event):
            self.logger.AppendText('EvtRadioBox:
%d\n' % event.GetInt())
        def EvtComboBox(self, event):
            self.logger.AppendText('EvtComboBox:
%s\n' % event.GetString())
        def OnClick(self,event):
            self.logger.AppendText("    The    result
is:.... %d\n" %event.GetId())
        def EvtText(self, event):
            self.logger.AppendText('EvtText: %s\n' %
event.GetString())
        def EvtChar(self, event):
            self.logger.AppendText('EvtChar: %d\n' %
event.GetKeyCode())
            event.Skip()
        def EvtCheckBox(self, event):
            self.logger.AppendText('EvtCheckBox:
%d\n' % event.Checked())

app = wx.App(False)

```

```

frame = ExampleFrame(None)
frame.Show()
app.MainLoop()

```

untuk menampilkan data hasil query, dapat digunakan rutin berikut:

```

def getTerms(self, line):
    line=line.lower() #NOTE: mengecilkan semua
huruf
    line=re.sub(r'[^a-z0-9 ]',' ',line) #NOTE:
mengganti karakter non-alfanumerik dengan spasi
    line=line.split() #NOTE: isi line menjadi
array, adalah ['kata1', 'kata2', 'kata3', ...]
    line=[x for x in line if x not in
self.stop_words] #NOTE: menghapus kata stopwords
    #line=[ porter.stem(word, 0, len(word)-1) for
word in line] #NOTE: membuang imbuhan
    line=[ porter.stem(word) for word in line]
#NOTE: membuang imbuhan
    return line

def getPostings(self, terms):
    #NOTE: all terms in the list are guaranteed to
be in the index
    #NOTE: mengembalikan indeks yang ada kata
dicari (docID, dan posting)
    return [ self.index[term] for term in terms ]

def getDocsFromPostings(self, postings):
    #NOTE: no empty list in postings
    #NOTE: mengembalikan DocID-nya
    return [ x[0] for x in p] for p in postings ]

def readIndex(self):
    #read main index
    f=open(self.indexFile, 'r');
    #first read the number of documents
    self.numDocuments=int(f.readline().rstrip())
    for line in f:
        line=line.rstrip()

```

```

        term, postings, tf, idf = line.split('|')
#NOTE:                                     term='termID',
postings='docID1:pos1,pos2;docID2:pos1,pos2'
        postings=postings.split(';')      #NOTE:
postings=['docID1:pos1,pos2','docID2:pos1,pos2']
        postings=[x.split(':') for x in postings]
#NOTE: postings=[['docID1', 'pos1,pos2'], ['docID2',
'pos1,pos2']]
        postings=[          [int(x[0]),          map(int,
x[1].split(','))] for x in postings ]      #NOTE:
postings=[ [docID1, [pos1,pos2]], [docID2, [pos1,pos2]]
]

        self.index[term]=postings #NOTE: hashtable
term terhadap postings

        #NOTE: read term frequencies
        tf=tf.split(',')      #NOTE:      mengubah
"0.1,0.2,0.3" menjadi array ['0.1','0.2','0.3']
        self.tf[term]=map(float,      tf)      #NOTE:
mengubah array ['0.1','0.2','0.3'] menjadi array
[0.1,0.2,0.3]

        #read inverse document frequency
        self.idf[term]=float(idf)
f.close()

```

2.7.3 Hadoop untuk Distributed IR

Koleksi yang sangat besar membutuhkan lebih dari 1 komputer, misalnya pada WWW. Oleh karena itu web search engine menggunakan **distributed indexing** algorithms untuk index construction. Lakukan instalasi Hadoop di Linux dengan 1 Master dan 2 client. Contoh penerapan Hadoop dan MapReduce untuk menghitung banyaknya kata:

WordCount.java

```

package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

```

```

|
public class WordCount {
    public static class Map extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(LongWritable key, Text value, OutputCollector<Text,
        IntWritable> output, Reporter reporter) throws IOException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
    }

    public static class Reduce extends MapReduceBase implements
    Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter) throws
        IOException {
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
    }
}

```

```

FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));
JobClient.runJob(conf);
}}

```

Penggunaan:

Assuming HADOOP_HOME is the root of the installation and HADOOP_VERSION is the Hadoop version installed, compile WordCount.java and create a jar:

```

$ mkdir wordcount_classes
$ javac -classpath ${HADOOP_HOME}/hadoop-
${HADOOP_VERSION}-core.jar -d wordcount_classes WordCount.java
$ jar -cvf /usr/joe/wordcount.jar -C wordcount_classes/ .

```

Assuming that:

/usr/joe/wordcount/input - input directory in HDFS
 /usr/joe/wordcount/output - output directory in HDFS

Sample text-files as input:

```

$ bin/hadoop dfs -ls /usr/joe/wordcount/input/
/usr/joe/wordcount/input/file01
/usr/joe/wordcount/input/file02

```

```

$ bin/hadoop dfs -cat /usr/joe/wordcount/input/file01
Hello World Bye World

```

```

$ bin/hadoop dfs -cat /usr/joe/wordcount/input/file02
Hello Hadoop Goodbye Hadoop

```

Run the application:

```

$ bin/hadoop jar /usr/joe/wordcount.jar org.myorg.WordCount
/usr/joe/wordcount/input /usr/joe/wordcount/output

```

Output:

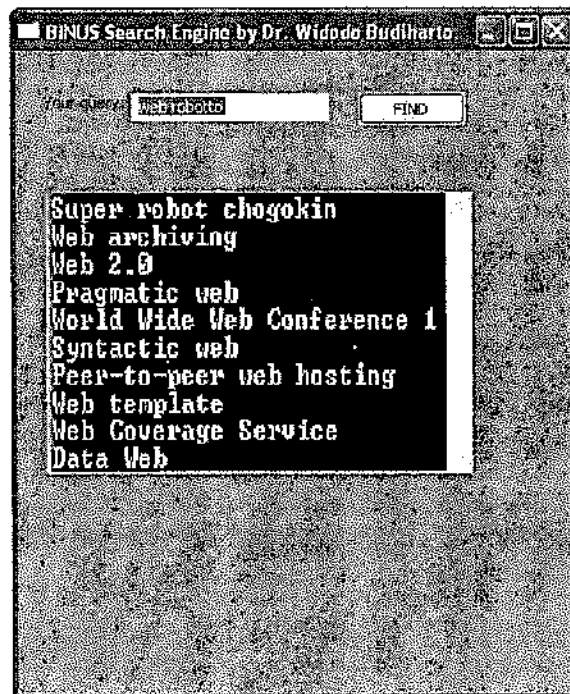
```

$ bin/hadoop dfs -cat /usr/joe/wordcount/output/part-00000
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2

```


Latihan

1. Gunakan contoh query menggunakan inverted index, tingkatna untuk membuat search engine seperti berikut:



Gambar 2.9 Form Search Engine

2. Buatlah sebuah sistem search engine untuk Reuters-RCV1 Collection dan implementasikan menggunakan MapReduce (Berbasiskan Hadoop). Lalu, buatlah technical report yang terdiri dari: concept, comparison, analyze dan diskusi hasil. (4-5 halaman).

References

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Scutze (2008), *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 978-0521865715.
- EMC Education Services,(2015), *Data Science and Big Data Analytics*, Wiley & Sons Publisher, ISBN: 978-1-118-87613-8.Python.org.
- <http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/>
- <http://www.ardendertat.com/2011/05/30/how-to-implement-a-search-engine-part-1-create-index/>
- <http://www.glennklockwood.com/di/hadoop-streaming.php>
- Kai Gao et al (2008). *Tolerant Retrieval and Query Processing in Search Engine*, International conference on Computer Science and Software Engineering, pp. 593 - 596, China.
- Ricardo Baeza-Yates and Berthier ribeiro-Neto (2011), *Modern Information Retrieval*, 2nd edition, ACM Books Press. ISBN: 978-0-321-41691-9.
- Stefan Butcher, Charles L.A Clarke and Gordon V. Cormack (2010), *Information Retrieval- Implementing and Evaluating Search Engines*, MIT Press, ISBN: 978-0-262-02651-2.

Bab 3

Evaluasi Retrieval

Tujuan Instruksional Umum :

1. Mahasiswa mampu menjelaskan definisi dan bagian penting dari evaluasi retrieval (retrieval evaluation)

Tujuan Instruksional Khusus :

1. Mahasiswa dapat menyebutkan mengenai Cranfield paradigm
2. Mahasiswa dapat menjelaskan mengenai retrieval metrics
3. Mahasiswa dapat menjelaskan mengenai reference collections
4. Mahasiswa dapat menjelaskan mengenai user-based evaluation

3.1 Pendahuluan

Ada 2 prinsip untuk mengukur performa Sistem IR, efisiensi dan efektifitas. Efisiensi diukur dalam skala waktu (detik per query) dan space (misalnya byte per document). Aspek paling penting dari efisiensi ialah response time(juga dikenal sebagai latency) yang dialami user diantara mengeluarkan query dan menerima hasil. Ketika banyak user harus disupport, query througput, diukur dalam query per detik menjadi sangat penting pada performa system. Pada search engine standar, througput yang diinginkan harus mampu dalam order 10 ribu query per detik. Efektifitas lebih susah diukur daripada efisiensi, karena tergantung pada penilaian manusia. Kata kunci efektifitas ialah relevance(dokumen disebut relevant sesuai query yang diberikan jika kontennya secara lengkap atau parsial memenuhi information need dari query. Untuk menentukan relevance, seseorang reviewer memberikan nilai relevance dalam bentuk grade atau biner.

Tujuan mendasar dari relevance ranking ialah ditunjukkan dalam bentuk Probability Ranking Principle (PRP), dimana :

Jika Sistem IR merespon ke tiap query adalah sebuah ranking dari dokumen di dalam koleksi agar menurunkan probabilitas relevance, maka efektifitas keseluruhan dari system ke penggunaanya akan dimaksimalkan.

Dokumen berbahasa Inggris, cukup membutuhkan 7 bit nilai ASCII, namun tidak halnya untuk bahasa lainnya. Oleh karena itu dibutuhkan skema koding, misalnya menggunakan UTF-8 yang menyediakan 1-4 byte encoding.

Untuk mengukur informasi ad hoc efektivitas pengambilan (ad hoc information retrieval effectiveness) dengan cara standar, kita perlu *test collection* yang terdiri dari tiga hal:

1. Sebuah koleksi dokumen
2. Sebuah suite uji kebutuhan informasi, dinyatakan sebagai query
3. Satu set penilaian relevansi, penilaian biner baik yang relevan atau tidak relevan untuk masing-masing pasangan permintaan-dokumen.

Standar uji koleksi antara lain:

1. Cranfield collection. Ini adalah koleksi uji perintis dalam memungkinkan ukuran kuantitatif yang tepat efektivitas pencarian informasi, tetapi saat ini terlalu kecil untuk apa pun kecuali percobaan percontohan yang paling dasar. Dikumpulkan di Inggris dimulai pada akhir 1950-an, berisi 1.398 abstrak aerodinamis artikel jurnal, himpunan 225 pertanyaan, dan penilaian relevansi lengkap pasangan (query, dokumen)
2. Text Retrieval Conference (TREC). NIST menjalankan *IR Test bed evaluation series* sejak 1992. Dalam kerangka ini, ada banyak lagu selama rentang koleksi uji beda, tetapi koleksi tes terbaik dikenal adalah yang digunakan untuk lagu TREC Ad Hoc selama 8 pertama evaluasi TREC antara tahun 1992 dan 1999. Seperti kebanyakan koleksi tes, koleksi TREC adalah terdiri dari tiga bagian:
 - a. Dokumen-dokumen
 - b. Contoh permintaan informasi (disebut topik)
 - c. Satu set dokumen yang relevan untuk setiap permintaan informasi misalnya
3. Reuter-RCV1 untuk klasifikasi teks.

Precision, Recall dan F-Measure

Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan *recall* adalah tingkat keberhasilan sistem dalam

menemukan kembali sebuah informasi. Accuracy didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Ilustrasi berikut ini memberikan gambaran perbedaan antara *accuracy* dan *precision*. Mereka dihitung menggunakan set unordered dokumen. Kita perlu untuk memperpanjang langkah-langkah ini (atau untuk menentukan langkah-langkah baru) jika kita mengevaluasi hasil pengambilan yang peringkat yang sekarang standar dengan mesin pencari.

Misalkan kita ingin mengukur kinerja dari sebuah mesin pemisah ikan yang bertugas memisahkan ikan-ikan salmon dari semua ikan yang telah didapat. Untuk mengujinya kita akan memasukkan 100 ikan salmon dan 900 ikan lain (bukan ikan salmon). Hasilnya mesin tersebut memisahkan 110 yang dideteksi sebagai ikan salmon. Ke 110 ikan tersebut kemudian dicek kembali oleh manusia, ternyata dari 110 ikan tersebut hanya 90 ekor yang merupakan ikan salmon, sedangkan 20 lainnya merupakan ikan lain.

Dari kasus tersebut maka kita dapat simpulkan bahwa mesin tersebut memiliki *precision* sebesar 82%, *recall* sebesar 90% dan *accuracy* sebesar 97% yang didapatkan dari perhitungan berikut:

$$precision = \frac{\text{jumlah salmon yang dipisahkan dengan benar}}{\text{jumlah ikan yang dipisahkan}}$$

$$precision = \frac{90}{110} = 0.82 = 82 \%$$

$$recall = \frac{\text{jumlah salmon yang dipisahkan dengan benar}}{\text{jumlah salmon sebenarnya}}$$

$$recall = \frac{90}{100} = 0.9 = 90 \%$$

$$accuracy = \frac{\text{jumlah ikan yang dipisahkan dengan benar}}{\text{jumlah ikan total}}$$

$$accuracy = \frac{(\text{jumlah salmon} + \text{jumlah ikan bukan salmon}) \text{ yg dipisahkan dg benar}}{\text{jumlah ikan total}}$$

$$accuracy = \frac{90 + 880}{1000} = 0.97 = 97 \%$$

Secara umum precision, recall dan accuracy dapat dirumuskan sebagai berikut:

		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	TP (True Positive) <i>Correct result</i>	FP (False Positive) <i>Unexpected result</i>
	FALSE	FN (False Negative) <i>Missing result</i>	TN (True Negative) <i>Correct absence of result</i>

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sehingga untuk kasus mesin pemisah ikan diatas dapat dituliskan sebagai berikut:

		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	90	20
	FALSE	10	880

$$precision = \frac{90}{90 + 20} = \frac{90}{110} = 0.82 = 82\%$$

$$recall = \frac{90}{90 + 10} = \frac{90}{100} = 0.9 = 90\%$$

$$accuracy = \frac{90 + 880}{90 + 880 + 20 + 10} = \frac{970}{1000} = 0.97 = 97\%$$

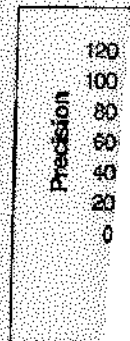
Sekarang ditunjukkan hal yang menarik. Menggunakan *precision* atau *accuracy* saja dalam sebuah mengukur kinerja dari sebuah sistem / metode bisa menimbulkan bias yang sangat fatal. Sebagai contoh, misalnya dari pengujian menggunakan 100 ikan salmon dan 900

ikan lain terny dicek oleh m Pengujian ini c

Nilai prediksi

acc

Dar accuracy st recall yang dapat men masih bany



(MAP)

ikan lain ternyata mesin hanya memisahkan 1 ikan salmon, dan setelah dicek oleh manusia, 1 ikan tersebut benar merupakan ikan salmon. Pengujian ini dapat kita tuliskan sebagai berikut:

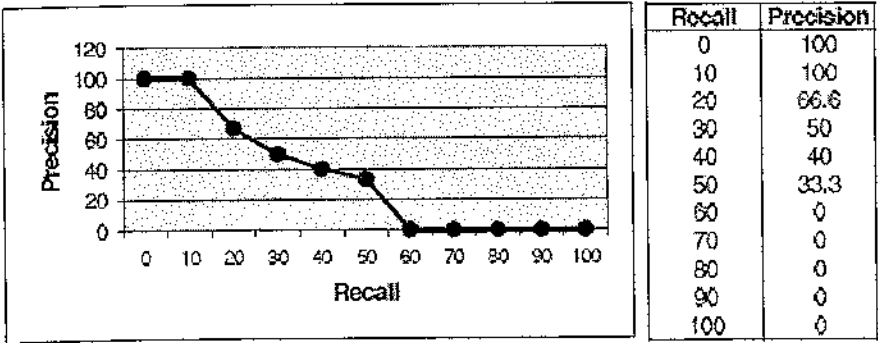
		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	1	0
	FALSE	99	900

$$precision = \frac{1}{1 + 0} = \frac{1}{1} = 1 = 100\%$$

$$recall = \frac{1}{1 + 99} = \frac{1}{100} = 0.01 = 1\%$$

$$accuracy = \frac{1 + 900}{1 + 900 + 0 + 99} = \frac{901}{1000} = 0.901 = 90.1\%$$

Dari hasil perhitungan kita dapatkan *precision* sebesar 100% dan *accuracy* sebesar 90.1%. Sekilas tampak baik, namun perhatikan nilai *recall* yang hanya sebesar 1%. Hal ini menunjukkan bahwa sistem hanya dapat memisahkan ikan salmon dalam jumlah yang sedikit sekali dan masih banyak ikan-ikan salmon yang lolos dari pemisahan.



Gambar 3.1 Perbandingan Precision dan recall

Selain itu, terdapat metode lain seperti Mean Average Precision (MAP) untuk qI diberikan sebagai

$$MAP_1 = \frac{1 + 0.66 + 0.5 + 0.4 + 0.33 + 0 + 0 + 0 + 0 + 0}{10} = 0.28$$

F-measure juga merupakan ukuran tunggal yang menggabungkan recall dan presisi

$$F(j) = \frac{2}{\frac{1}{r(j)} + \frac{1}{p(j)}}$$

Dimana:

- $r(j)$ is the recall at the j -th position in the ranking
- $P(j)$ is the precision at the j -th position in the ranking
- $F(j)$ is the harmonic mean at the j -th position in the ranking

Latihan

1. Jelaskan mengenai Mean Average Precision

References

1. Christopher D. Manning, Prabhakar Raghavan and Hinrich Scutze (2008), *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 978-0521865715.
2. Ricardo Baeza-Yates and Berthier ribeiro-Neto (2011), *Modern Information Retrieval*, 2nd edition, ACM Books Press. ISBN: 978-0-321-41691-9.
3. Stefan Butcher, Charles L.A Clarke and Gordon V. Cormack (2010), *Information Retrieval- Implementing and Evaluating Search Engines*, MIT Press, ISBN: 978-0-262-02651-2.
4. EMC Education Services,(2015), *Data Science and Big Data Analytics*, Wiley & Sons Publisher, ISBN: 978-1-118-87613-8.Python.org.

Bab 4

Relevance Feedback

Tujuan Instruksional Umum :

1. Mahasiswa mampu menjelaskan definisi dan bagian penting dari Relevance Feedback

Tujuan Instruksional Khusus :

1. Mahasiswa dapat menyebutkan mengenai Cranfield paradigm
2. Mahasiswa dapat menjelaskan mengenai retrieval metrics
3. Mahasiswa dapat menjelaskan mengenai reference collections
4. Mahasiswa dapat menjelaskan mengenai user-based evaluation

4.1 Pendahuluan

Dalam kebanyakan koleksi, konsep yang sama dapat disebut menggunakan kata-kata yang berbeda. Masalah ini, dikenal sebagai sinonim, memiliki dampak pada penarikan sebagian besar sistem information retrieval. Misalnya, Anda akan ingin mencari pesawat untuk mencocokkan pesawat (tapi hanya untuk referensi untuk pesawat terbang) dan untuk pencarian di termodinamika untuk mencocokkan referensi untuk diskusi yang menarik. Pengguna sering mencoba untuk mengatasi masalah ini sendiri secara manual memperbaiki query; kita membahas cara dimana sistem dapat membantu dengan query refinement, baik sepenuhnya otomatis atau dengan pengguna sekitarnya. Relevance Feedback (RF) merupakan suatu teknik temu kembali informasi dimana user memberikan feedback (pengaruh) pada dokumen hasil temu kembali yang dianggap relevan. Query Expansion (QE) merupakan suatu teknik kembali informasi untuk memperbaiki query sehingga dapat memperoleh hasil yang lebih baik.

Metode untuk mengatasi masalah ini dibagi menjadi dua kelas utama: metode global dan metode lokal. metode global merupakan teknik untuk memperluas atau reformulasi istilah permintaan independen dari permintaan dan hasil kembali dari itu, sehingga perubahan dalam kata-kata permintaan akan menyebabkan permintaan baru untuk mencocokkan istilah lain yang secara semantik sangat persis. Metode global yang meliputi: Permintaan ekspansi/reformulasi dengan tesaurus atau

WordNet,ekspansi Query melalui generasi tesaurus otomatis dan teknik seperti koreksi ejaan.

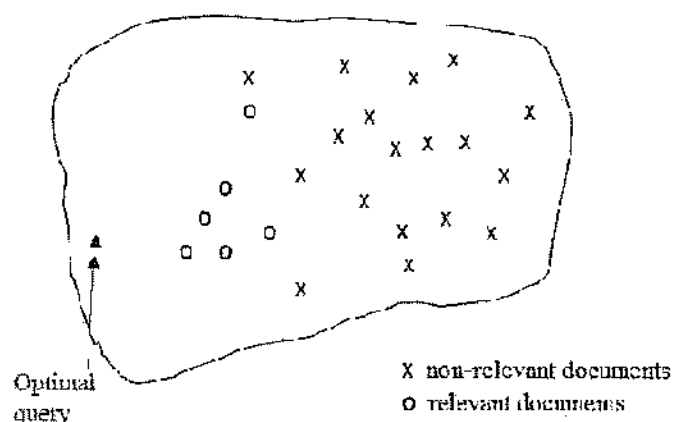
Motivasi untuk merequest pada Information retrieval system ialah information need (Lancaster, 1979), dan sukses sebuah system retrieval tergantung pada kemampuan system untuk menyediakan informasi yang dibutuhkan dengan waktu yang masuk akal dan dengan antarmuka yang baik untuk meminta data dan menerima hasil (Austin, 2001)

Pada model IR, belief network merupakan bagian model probabilistik yang dapat diterapkan pada relevance feedback. Keberhasilan relevant feedback tergantung pada asumsi-asumsi tertentu. Pertama, pengguna harus memiliki pengetahuan yang cukup untuk dapat membuat query inisial, yang setidaknya dekat dengan dokumen yang mereka inginkan. Hal ini diperlukan bagaimanapun untuk pencarian informasi yang berhasil dalam hal dasar, tetapi penting untuk melihat jenis masalah yang relevansi umpan balik tidak bisa memecahkan sendiri. Kasus di mana relevant feedback saja tidak cukup meliputi:

- a. Salah eja. Jika pengguna mengucapkan istilah dengan cara yang berbeda dengan cara dieja dalam dokumen koleksi, maka relevansi umpan balik tidak mungkin efektif. Hal ini dapat diatasi dengan teknik koreksi ejaan.
- b. Cross language information retrieval(Lintas bahasa pencarian informasi). Dokumen dalam bahasa lain tidak di dekatnya dalam ruang vektor berdasarkan term distribution. Sebaliknya, dokumen dalam cluster bahasa yang sama lebih dekat.

4.2 Algoritma Rocchio untuk relevant Feedback

Algoritma Rocchio adalah algoritma klasik untuk menerapkan relevance feedback. Ini model cara menggabungkan informasi relevant feedback ke dalam vector space model



Gambar 4.1 Rocchio optimal query untuk memisahkan dokumen relevant dan non relevant

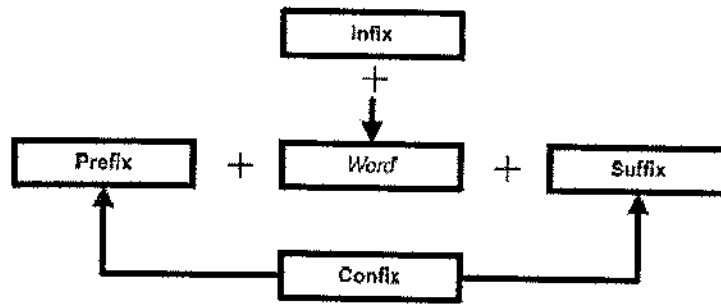
4.3 Relevant Feedback di Web

Beberapa search engine menawarkan fitur halaman yang sama/terkait: pengguna menunjukkan dokumen hasil ditetapkan sebagai teladan dari sudut pandang rapat informasinya perlu dan meminta lebih banyak dokumen seperti itu. Hal ini dapat dilihat sebagai bentuk sederhana tertentu umpan balik relevansi. Namun, dalam umpan balik relevansi umum telah sedikit digunakan dalam pencarian web. Satu pengecualian adalah Excite mesin pencari web, yang awalnya tersedia umpan balik relevansi penuh. Namun, fitur itu dalam waktu turun, karena kurangnya penggunaan. Di web, beberapa orang menggunakan antarmuka pencarian canggih dan paling ingin menyelesaikan pencarian mereka di satu interaksi. Tetapi kurangnya serapan juga mungkin mencerminkan dua faktor lainnya: umpan balik relevansi sulit untuk menjelaskan kepada pengguna rata-rata, dan umpan balik relevansi terutama penarikan meningkatkan strategi, dan pengguna pencarian web hanya jarang peduli dengan mendapatkan cukup recall.

Spink et al. (2000) menampilkan penggunaan relevant feedback di mesin pencari Excite. Hanya sekitar 4% dari sesi permintaan pengguna menggunakan opsi umpan balik relevansi, dan ini biasanya memanfaatkan "More like this" tautan berikutnya untuk setiap hasil. Sekitar 70% dari pengguna hanya melihat halaman pertama hasil dan tidak mengejar hal-hal lebih jauh. Bagi orang-orang yang menggunakan umpan balik relevansi, hasilnya meningkat sekitar dua pertiga dari waktu

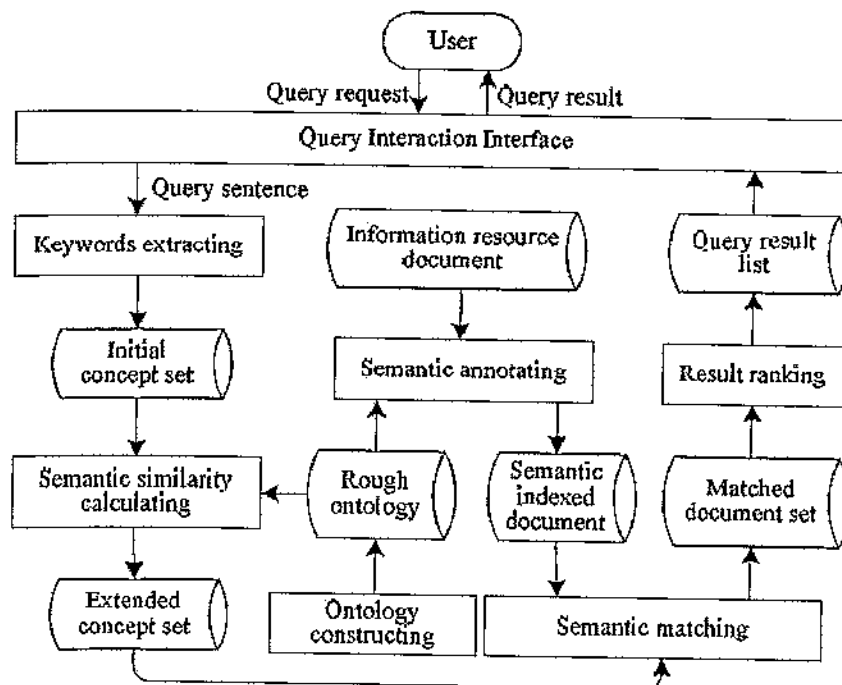
4.4 Semantic Information Retrieval

Stemming merupakan proses menurunkan kata kata dasar dengan menghilangkan affix dari kata tersebut. Bahasa indonesia terdiri dari prefix, infix, suffix dan confix



Gambar 4.2 konsep Affix concept pada bahasa Indonesia

Semantic information retrieval merupakan mekanisme pencocokan berdasarkan konsep relasinya seperti ditampilkan pada Gambar 4.3



Gambar 4.3 Semantic Information Retrieval Model[3]

References

- Brice Austin, Mooers' law: in and out of context, *Journal of Am. Soc. Inf. Sci.*, 52(8), 2001.
- Christopher D. Manning, Prabhakar Raghavan and Hinrich Scutze (2008), *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 978-0521865715.
- EMC Education Services,(2015), *Data Science and Big Data Analytics*, Wiley & Sons Publisher, ISBN: 978-1-118-87613-8.Python.org.
<http://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/>
- Kai Gao et al (2008). *Tolerant Retrieval and Query Processing in Search Engine*, International conference on Computer Science and Software Engineering, pp. 593 - 596, China.
- Lancaster, *Information Retrieval Systems; Characteristics Testing, Evaluation*, John Wiley and Sons, sendo edition, 1979.
- Ricardo Baeza-Yates and Berthier ribeiro-Neto (2011), *Modern Information Retrieval*, 2nd edition, ACM Books Press. ISBN: 978-0-321-41691-9.
- Stefan Buttcher, Charles L.A Clarke and Gordon V. Cormack (2010), *Information Retrieval- Implementing and Evaluating Search Engines*, MIT Press, ISBN: 978-0-262-02651-2.
- Yinghui Huang et al, Rough Ontology Based Semantic Information Retrieval, 2013 Sixth International Symposium on Computational Intelligence and Design.

Bab 5

Text Classification

Tujuan Instruksional Umum :

1. Mahasiswa mampu menjelaskan definisi dan bagian penting dari Text Classification

Tujuan Instruksional Khusus :

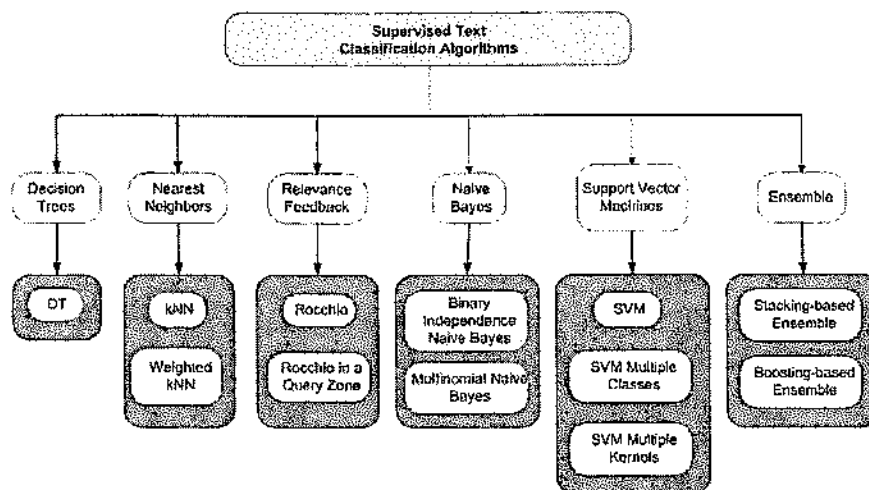
1. Mahasiswa dapat menyebutkan definisi dari text classification
2. Mahasiswa mengenal mengenai berbagai metode klasifikasi
3. Mahasiswa mengenal istilah kNN, Bayesian, Precision, Recall dan metode pengukurannya
4. Mahasiswa mampu melakukan riset dan pemrograman Python untuk Text Classification

5.1 Text Classification

Klasifikasi (*Classification*) ialah tugas memilih label kelas yang tepat jika diberikan sebuah input. Pada proses klasifikasi, biasanya dapat menggunakan proses training untuk melakukan proses klasifikasi. Beberapa tugas klasifikasi antara lain:

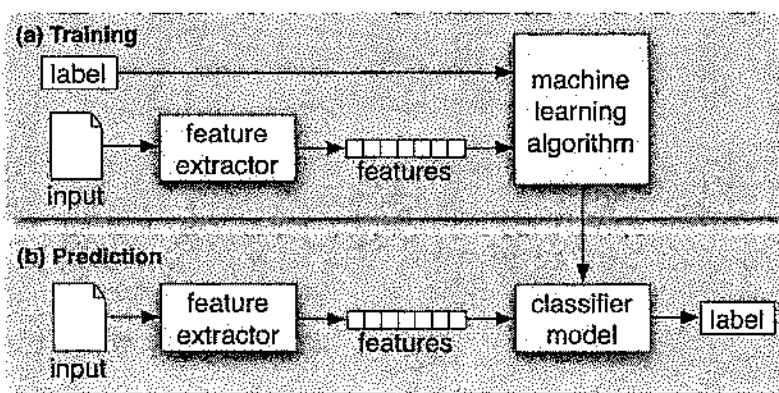
- Memutuskan apakah email spam atau tidak
- Memutuskan topic sebuah artikel masuk ke kategori "sport" atau "technology"
- Sentiment analysis.

Sebuah classifier disebut terawasi (*supervised*) jika dibangun berdasarkan training corpora berisi label yang benar untuk tiap input (misalnya menggunakan Decision tree, kNN, Naïve Bayes, SVM dan Ensemble). Hampir tiap metode klasifikasi merupakan **supervised**, seperti gambar di bawah:



Gambar 5.1 supervised text classification

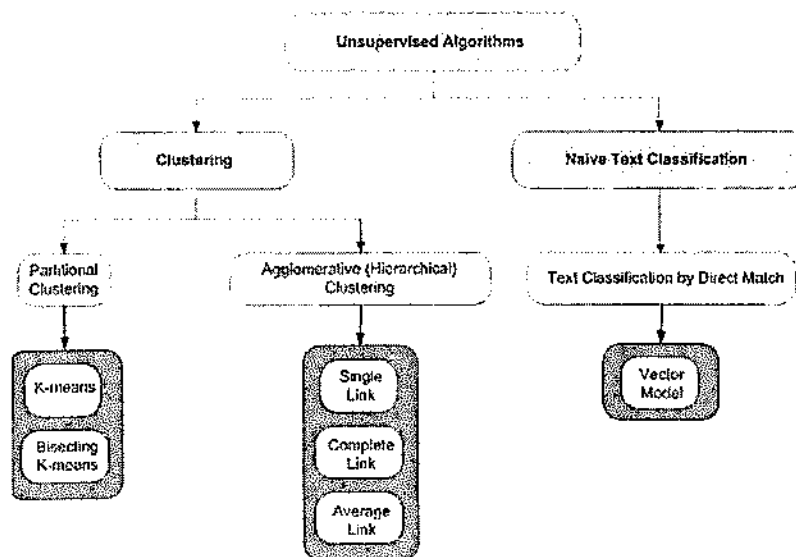
Pada supervised learning, digunakan training data untuk proses pelatihan, pada saat prediksi, maka test data/dokumen yang ingin diklasifikasikan akan dikenal menggunakan model classifier yang telah dihasilkan, untuk dapat ditentukan masuk ke kelas mana dokumen tersebut. Feature extractor akan mengambil fitur penting dari data yang diberikan.



Gambar 5.2 Supervised learning

Pada *unsupervised learning*, intinya kita tidak memberikan pasangan input output pada data pelatihan, dimana sistem akan berusaha mencari sendiri pattern data tersebut. Misalnya menggunakan SOM(Self

Organizing Map) pada Neural Network, Clustering dan Naïve Text Classification. Sistem akan memprediksi sendiri kelas output dari input yang diberikan. Contoh: diberikan 1 juta warna, setelah pelatihan, sistem unsupervised mampu mengelompokkan warna tersebut menjadi 20 jenis warna.



Gambar 5.3 Unsupervised learning

Fitur berisi informasi penting suatu kelompok data. Berikut ini contoh gender classification menggunakan Naïve Bayes Classifier:

```

Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
0.748
Most Informative Features
last_letter = u'a'      female : male = 33.0 : 1.0
last_letter = u'k'      male : female = 32.8 : 1.0
last_letter = u'f'      male : female = 19.6 : 1.0
last_letter = u'p'      male : female = 11.9 : 1.0
last_letter = u'v'      male : female = 11.2 : 1.0
>>> |
  
```

Gambar 5. 4 Hasil gender classification

Terlihat bahwa nama laki dan wanita memiliki karakteristik tertentu. Nama berakhiran *a, e, dan l cenderung* nama wanita, sedangkan berakhiran *k, o, r, s, t cenderung* lelaki.

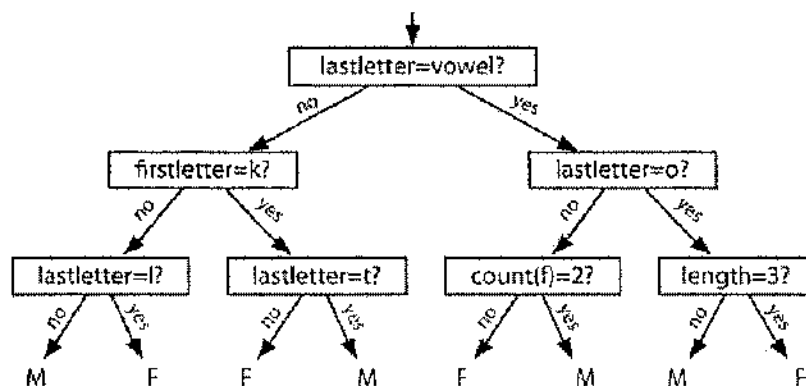
Gender.py:

```
import nltk
from nltk.corpus import names
def gender_features(word):
    return {'last_letter': word[-1]}

labeled_names = ([ (name, 'male') for name in names.words('male.txt')] +
[ (name, 'female') for name in names.words('female.txt')])
import random
random.shuffle(labeled_names)
featuresets = [(gender_features(n), gender) for (n, gender) in
labeled_names]
train_set, test_set = featuresets[500:], featuresets[:500]
classifier = nltk.NaiveBayesClassifier.train(train_set)
classifier.classify(gender_features('Neo'))
classifier.classify(gender_features('Trinity'))
print(nltk.classify.accuracy(classifier, test_set))
classifier.show_most_informative_features(5)
```

5.2 Decision Trees

Sebuah **decision tree** merupakan flowchart yang memilih label untuk nilai input, terdiri dari **decision nodes**, dimana cek nilai fitur dan **leaf nodes**,

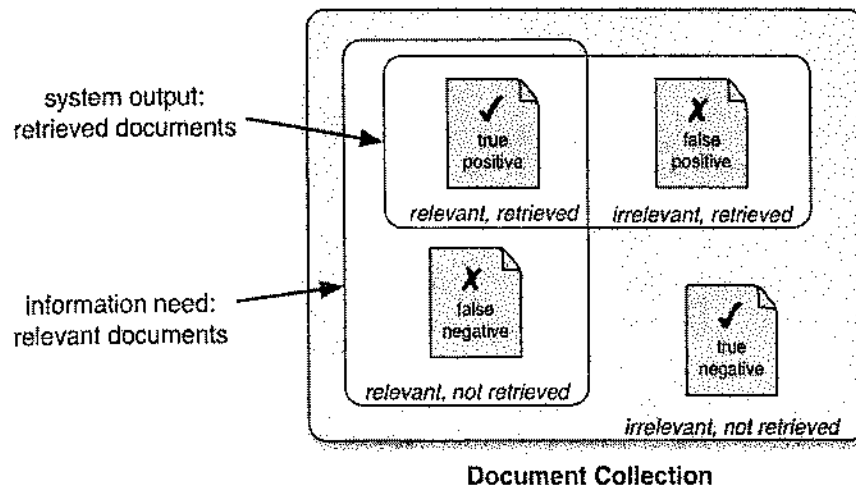


Gambar 5.5 Decision Tree model for the name gender task. Note that tree diagrams are conventionally drawn “upside down,” with the root at the top, and the leaves at the bottom.

5.3 Precision and Recall

Precision mengukur ketepatan sebuah classifier. Sebuah precision yang lebih tinggi berarti lebih sedikit false positives. Ini sering bertentangan dengan recall, dimana cara termudah untuk meningkatkan presisi adalah menurunkan recall. Recall mengukur kelengkapan (completeness) atau sensitivity dari sebuah classifier. Higher recall berarti lebih sedikit false negatives. Meningkatkan recall dapat sering menurunkan precision. Precision dan recall dapat dikombinasikan untuk menghasilkan metric bernama F-measure, dimana rata-rata harmonic berbobot dari presisi dan recall.

Bentuk pengukuran akurasi menggunakan recall and precision, seperti berikut:



Gambar 5.6 True and False Positives and

Negatives

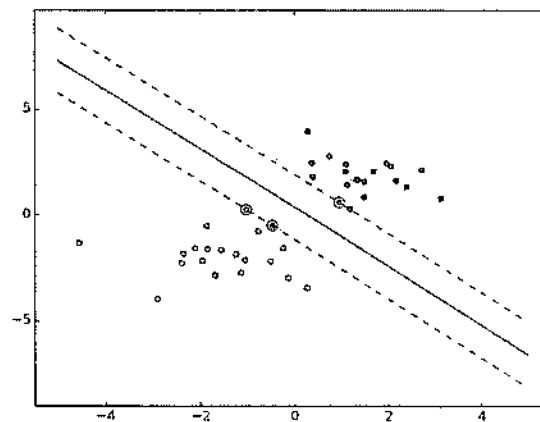
- True positives are relevant items that we correctly identified as relevant.
- True negatives are irrelevant items that we correctly identified as irrelevant.
- False positives (or Type I errors) are irrelevant items that we incorrectly identified as relevant.
- False negatives (or Type II errors) are relevant items that we incorrectly identified as irrelevant.

Diberikan 4 bilangan di atas, dapat didefinisikan metric berikut:

- Precision, which indicates how many of the items that we identified were relevant, is $TP/(TP+FP)$.
- Recall, which indicates how many of the relevant items that we identified, is $TP/(TP+FN)$.
- The F-Measure (or F-Score), which combines the precision and recall to give a single score, is defined to be the harmonic mean of the precision and recall: $(2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$.

5.4 SVM Classifier

SVM merupakan supervised classifier, sering digunakan untuk mengekstraksi statistic corpus untuk sentiment analysis.



Gambar 5.7 hyperplane pada SMV

Support vector machine membentuk hyper-plane atau himpunan hyper-planes pada ruang dimensi infinite, yang dapat digunakan untuk klasifikasi dan regresi. Keuntungan dari SVM adalah Efektif pada high dimensional spaces, serta kernel fungsi yang berbeda dapat ditentukan untuk fungsi keputusan.

Berikut contohnya:

```
>>> from sklearn import svm
>>> X = [[0], [1], [2], [3]]
```

```

>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC()
>>> clf.fit(X, Y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3,
gamma=0.0, kernel='rbf', max_iter=-1, probability=False,
random_state=None,
shrinking=True, tol=0.001, verbose=False)
>>> dec = clf.decision_function([[[1]]])
>>> dec.shape[1] # 4 classes: 4*3/2 = 6
6

```

5.5 NLTK untuk Klasifikasi

Demo NLTK berisi demo sentiment analysis movie reviews corpus dengan mereview kategori pada *pos* dan *neg*, serta sebuah classifier. Pada NaiveBayesClassifier menggunakan boolean word feature extraction. Seluruh classifier NLTK bekerja menggunakan featstructs, dimana dapat sebagai dictionary yang memetakan sebuah *feature name* ke sebuah *feature value*. Movie reviews corpus memiliki 1000 positive files dan 1000 negative files. Kita gunakan 3/4 darinya sebagai training set, dan sisanya test set. Sehingga diperoleh 1500 training instances dan 500 test instances. Untuk evaluasi akurasi, digunakan `nltk.classify.util.accuracy`. Berikut contoh simple dari klasifikasi kalimat positif/negatif:

Sentenceclassification.py:

```

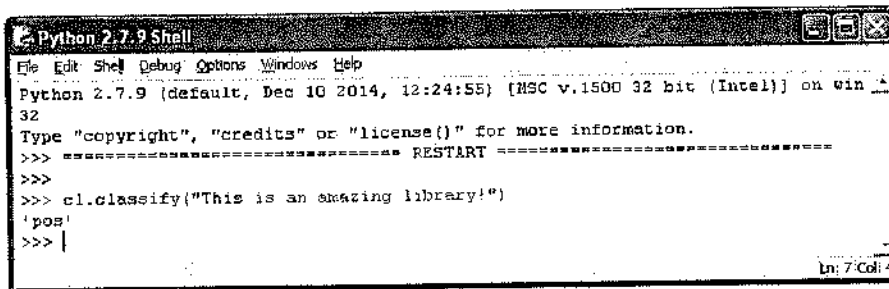
train = [
('I love this sandwich.', 'pos'),
('this is an amazing place!', 'pos'),
('I feel very good about these beers.', 'pos'),
('this is my best work.', 'pos'),
('what an awesome view', 'pos'),
('I do not like this restaurant', 'neg'),
('I am tired of this stuff.', 'neg'),
('I can't deal with this', 'neg'),
('he is my sworn enemy!', 'neg'),
('my boss is horrible.', 'neg')
]
test = [
('the beer was good.', 'pos'),

```

```

('I do not enjoy my job', 'neg'),
("I ain't feeling dandy today.", 'neg'),
('I feel amazing!', 'pos'),
('Gary is a friend of mine.', 'pos'),
("I can't believe I'm doing this.", 'neg')
]
from textblob.classifiers import NaiveBayesClassifier
cl = NaiveBayesClassifier(train)
cl.classify("This is an amazing library!")

```



Gambar 5.8 Hasil

Contoh berikut menggunakan NaiveBayes:

NaiveBayes.py:

```

import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews

def word_feats(words):
    return dict([(word, True) for word in words])

negids = movie_reviews.fileids('neg')
posids = movie_reviews.fileids('pos')

negfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'neg') for f in
negids]
posfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'pos') for f in
posids]

negcutoff = len(negfeats)*3/4

```

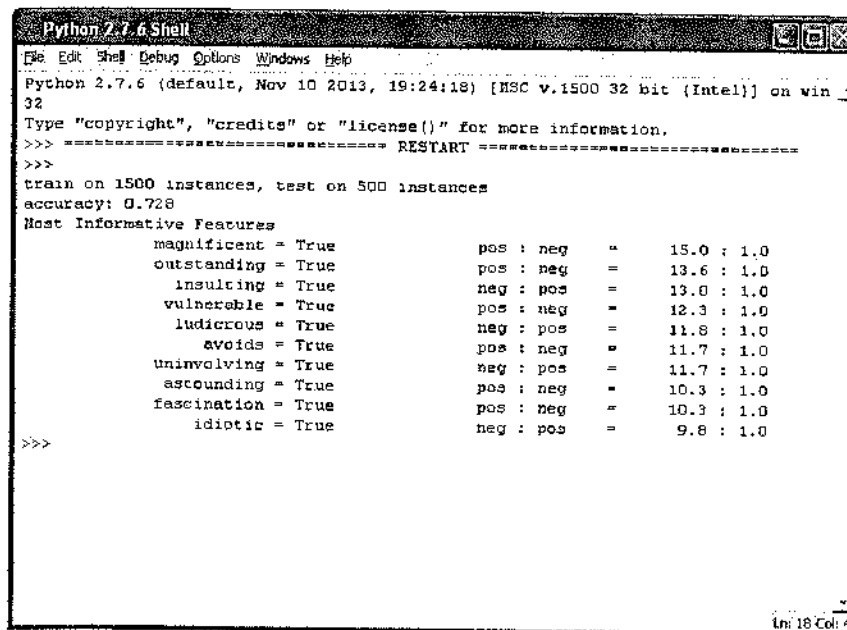
```

poscutoff = len(posfeats)*3/4

trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]
print 'train on %d instances, test on %d instances' % (len(trainfeats),
len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
classifier.show_most_informative_features()

```



Gambar 5.9 Hasil

5.6 Classifier Precision dan recall

Precision mengukur ketepatan sebuah classifier. Sebuah precision yang lebih tinggi berarti lebih sedikit false positives. Ini sering bertentangan dengan recall, dimana cara termudah untuk meningkatkan presisi adalah menurunkan recall. Recall mengukur kelengkapan (completeness) atau sensitivity dari sebuah classifier. Higher recall berarti lebih sedikit false negatives. Meningkatkan recall dapat sering menurunkan precision.

Precision dan recall dapat dikombinasikan untuk menghasilkan metric bernama F-measure, dimana rata rata harmonic berbobot dari presisi dan recall.

5.7 Naïve Bayes

Prinsip dasar Naïve Bayes:

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y)$$

$$\Downarrow$$

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y), \quad (1)$$

GaussianNB mengimplementasikan algoritma Gaussian naïve Bayes untuk klasifikasi. Kemungkinann dari features diasumsikan Gaussian:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp \left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right) \quad (2)$$

Parameter σ_y dan μ_y diestimasi menggunakan maximum likelihood.

GaussianBayes.py:

```
from sklearn import datasets
iris = datasets.load_iris()
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
print("Number of mislabeled points out of a total %d points : %d"
      % (iris.data.shape[0], (iris.target != y_pred).sum()))
```

Number of mislabeled points out of a total 150 points : 6

NaiveBayesPrecRecall.py:

```
import nltk.classify.util
import collections
import nltk.metrics
from nltk.classify import NaiveBayesClassifier
```



```

from nltk.corpus import movie_reviews

def word_feats(words):
    return dict([(word, True) for word in words])

negids = movie_reviews.fileids('neg')
posids = movie_reviews.fileids('pos')

negfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'neg') for f in
negids]
posfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'pos') for f in
posids]

negcutoff = len(negfeats)*3/4
poscutoff = len(posfeats)*3/4

trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]
print 'train on %d instances, test on %d instances' % (len(trainfeats),
len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
classifier.show_most_informative_features()

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (feats, label) in enumerate(testfeats):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)

print 'pos precision:', nltk.metrics.precision(refsets['pos'], testsets['pos'])
print 'pos recall:', nltk.metrics.recall(refsets['pos'], testsets['pos'])
print 'pos F-measure:', nltk.metrics.f_measure(refsets['pos'], testsets['pos'])
print 'neg precision:', nltk.metrics.precision(refsets['neg'], testsets['neg'])
print 'neg recall:', nltk.metrics.recall(refsets['neg'], testsets['neg'])

```

```
print 'neg F-measure:', nltk.metrics.f_measure(refsets['neg'], testsets['neg'])
```

```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
train on 1500 instances, test on 500 instances
accuracy: 0.738
Most Informative Features
magnificent = True          pos : neg = 15.0 : 1.0
outstanding = True          pos : neg = 13.6 : 1.0
insulting = True            neg : pos = 13.0 : 1.0
vulnerable = True           pos : neg = 12.3 : 1.0
ludicrous = True            neg : pos = 11.8 : 1.0
avoids = True               pos : neg = 11.7 : 1.0
uninvolving = True          neg : pos = 11.7 : 1.0
astounding = True           pos : neg = 10.3 : 1.0
fascination = True          pos : neg = 10.3 : 1.0
idiotic = True              neg : pos = 9.8 : 1.0
pos precision: 0.651595744681
pos recall: 0.98
pos F-measure: 0.782747603834
neg precision: 0.959677419355
neg recall: 0.476
neg F-measure: 0.636363636364
>>>
```

Gambar 5.10 Hasil Naïve Bayes

Berdasarkan data di atas:

1. Nearly every file that is pos is correctly identified as such, with 98% recall. This means very few false negatives in the pos class.
2. But, a file given a pos classification is only 65% likely to be correct. Not so good precision leads to 35% false positives for the pos label.
3. Any file that is identified as neg is 96% likely to be correct (high precision). This means very few false positives for the neg class.
4. But many files that are neg are incorrectly classified. Low recall causes 52% false negatives for the neg label.
5. F-measure provides no useful information. There's no insight to be gained from having it, and we wouldn't lose any knowledge if it was taken away.

Berikut contoh klasifikasi dengan Naïve Bayes

NaiveBayesPrediction.py:

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews
```

```

from collections import defaultdict
import numpy as np

SPLIT=0.8 #definisi 80/20 split untuk train/test
def word_feats(words):
    feats=defaultdict(lambda: False)
    for word in words:
        feats[word]=True
    return feats

posids=movie_reviews.fileids('pos')
negids=movie_reviews.fileids('neg')
posfeats=[(word_feats(movie_reviews.words(fileids=[f])), 'pos')
           for f in posids]
negfeats=[(word_feats(movie_reviews.words(fileids=[f])), 'neg')
           for f in negids]
cutoff=int(len(posfeats) * SPLIT)

trainfeats=negfeats[:cutoff] + posfeats[:cutoff]
testfeats=negfeats[cutoff:] + posfeats[cutoff:]

print 'Train on %d instances \n Test on %d instances' %
(len(trainfeats), len(testfeats))

classifier =NaiveBayesClassifier.train (trainfeats)
print 'Accuracy:', nltk.classify.util.accuracy(classifier, testfeats)

classifier.show_most_informative_features()

pos=[classifier.classify(fs) for (fs,l) in posfeats[cutoff:]]
pos=np.array(pos)
neg=[classifier.classify(fs) for (fs,l) in negfeats[cutoff:]]
neg=np.array(neg)

print 'Confusion matrix:'
print '\t'*2, 'Predicted class'
print '-'*40

```