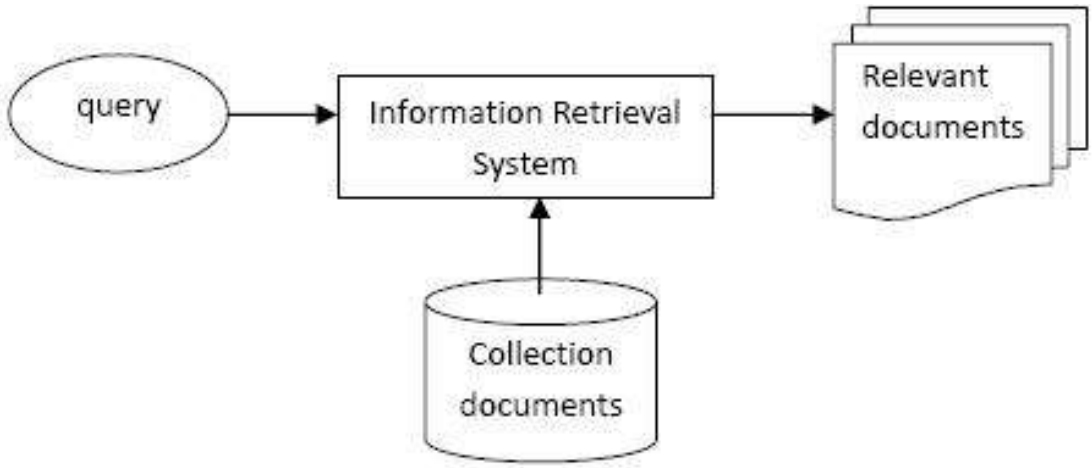
BAB 2
LANDASAN TEORI

2.1 *Information Retrieval* (IR)

Information Retrieval (IR) adalah suatu bidang penelitian yang menargetkan pencarian informasi secara efektif dan efisien dalam dokumen teks dan multimedia**.** *Information Retrieval* digunakan untuk menemukan kembali informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis berdasarkan *query* yang diinputkan oleh pengguna. *Query* adalah kebutuhan informasi pengguna yang biasanya direpresentasikan dengan satu atau beberapa kata.

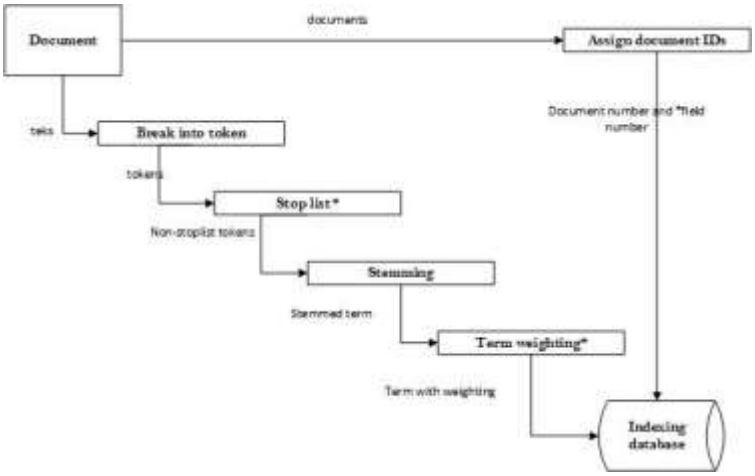


Gambar 2.1 Gambaran besar sistem Information Retrieval[7]

Pada Gambar 2.1 dapat dilihat bahwa *Information Retrieval System* akan menyaring atau mencari suatu data berdasarkan *query* yang diinputkan oleh *user*. *Information Retrieval System* terdiri dari dua bagian utama, yaitu *Indexing subsytem* dan *Searching subsystem (matching system).*[7]

a. *Indexing Subsystem*

Proses *indexing* pada *information retrieval* dapat dilihat pada gambar 2.2.



Gambar 2.2 Alur *Indexing subsystem*[7]

Pada gambar 2.2 dapat dilihat bahwa ada beberapa tahapan untuk melakukan *indexing* sampai tersimpang didalam *database* antara lain menyimpan ID dokumen dan menyimpan *term index* melalui memecah isi dokumen kedalam token (*break into token*), melakukan *stoplist, stemming, term indexing*.

1. *Break Into Token*

   Merupakan proses memecah (*parsing*) dokumen menjadi token-token yaitu dengan memotong menjadi *term* (kata, frase).

2. *StopList*

   Merupakan proses menghilangkan kata-kata umum. Misalkan dalam, dan, karena, mana, untuk dan lainnya. Dalam pembahasan ini kita batasi hanya dilakukan pada bahasa Indonesia saja. Pada proses selanjutnya juga akan dibatasi hanya untuk bahasa Indonesia.

3. *Stemming*

   Merupakan proses pengelompokan kata secara morfologis sesuai dengan kelompoknya. Ada pengertian lain yaitu proses ini merubah kata-kata yang berimbuhan menjadi kata dasar dengan menghilangkan imbuhan, sisipan maupun akhiran. Sebagai contoh kata ”memakan”, ”dimakan”, ”termakan”, ”makan-makan” akan dikelompokkan kedalam kata ”makan”. Kata ”cinta” memiliki turunan kata ”pecinta”, ”mencintai”, ”dicintai” dan lain sebagainya, sehingga

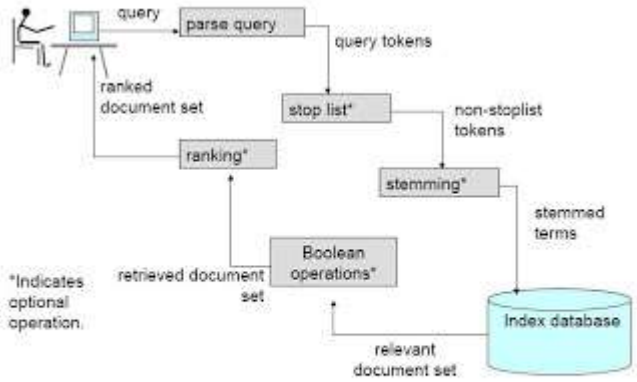
ketika orang memasukkan kata-kata turunan tersebut akan keluar dokumen yang relevan.

4. *Term Weighting*

   Term weighting adalah proses pembobotan *term* yang ada di dokumen (seberapa sering kemunculan term pada dokumen).

b. *Searching Subsystem*

Searching merupakan suatu proses mencari atau melakukan *query* ke dalam *database* berdasarkan *key word* yang dimasukkan oleh *user*. Alur dari *searching subsystem* dapat dilihat pada gambar 2.3 berikut :



Gambar 2.3 Alur proses *searching subsystem*[7]

Pada gambar 2.3 dapat dijelaskan bahwa tahapan dalam melakukan serching pada information retrieval yaitu :

1. *Parse query*

   Tahap awal dalam melakukan *searching* adalah memarsing *query*, *key word* yang di*input*kan *user* dijadikan sebagi *query* untuk melakukan pencarian ke dalam *database*.

2. *Stop list* Hasil dari proses *parse query* adalah *query token* (berisi *query* yang sudah dipotong berdasarkan *token*), kemudian dilakukan proses *stoplist* yaitu menghilangkan kata-kata yang umum.

3. *Stemming*

   Setelah *query* tersebut melewati proses *stoplist*, kemudian *query* tersebut melalui proses *stemming* yaitu suatu proses mengembalikan menjadi kata dasar dengan cara membuang awalan, akhiran atau sisipan. Hal ini dilakukan karena yang disimpan di dalam *index database* adalah kata dasar saja.

4. *Boolean operation*

   Setelah mendapatkan dokumen yang relevan dengan *key word* yang dicari maka proses berikutnya mengecek operator *boolean*.

5. *Ranking*

   Tahap yang terakhir adalah melakukan perangkingan, dari dokumen – dokumen yang di dapatkan. Perangkingan disusun berdasarkan tingkat kemiripan dokumen dengan *query*, dokumen yang paling besar tingkat kemiripanya dengan *query* menjadi dokumen dengan *ranking* teratas.

2.2 Model *Information Retrieval*

Model sisterm temu kembali informasi menentukan detail dari sistem temu kembali informasi, yaitu meliputi representasi dokumen maupun *query*, fungsi pencarian (*retrieval function*) dan notasi kesesuaian (*relevance notation*) dokumen terhadap *query*. Salah satu model sistem temu kembali informasi yang paling awal digunakan adalah model *Boolean*. Model *Boolean* merepresentasikan dokumen sebagai suatu himpunan kata kunci (*set of keywords*), sedangkan *query* direpresentasikan sebagai ekspresi *Boolean. Query* dalam ekspresi *Boolean* merupakan kumpulan kata kunci yang saling dihubungkan melalui operator *Boolean* seperti AND, OR, dan NOT, serta menggunakan tanda kurung untuk menentukan *scope* operator. Hasil pencarian dokumen dari model *Boolean* adalah himpunan dokumen yang relevan. Kekurangan model *Boolean* antara lain[4] :

1. Hasil pencarian dokumen berupa himpunan, sehingga tidak dapat dikenali dokumen-dokumen yang paling relevan atau agak relevan (*partial match*).

2. *Query* dalam ekspresi *boolean* dapat menyulitkan pengguna yang tidak mengerti tentang ekspresi *boolean*.

2.3 Arsitektur Sistem *Information Rerieval*

Secara garis besar arsitektur sistem IR mempunyai dua pekerjaan yang ditangani oleh sistem, yaitu melakukan *pre-processing* terhadap *database* dan kemudian menerapkan metode tertentu untuk menghitung kedekatan (relevansi atau *similarity*) antara dokumen di dalam *database* yang telah di*preprocess* dengan *query* pengguna. Pada tahapan *preprocessing*, sistem yang berurusan dengan dokumen *semi-structured* biasanya memberikan *tag* tertentu pada *term-term* atau bagian dari dokumen; sedangkan pada dokumen tidak terstruktur proses ini dilewati dan membiarkan *term* tanpa imbuhan *tag*. *Query* yang dimasukkan pengguna dikonversi sesuai aturan tertentu untuk mengekstrak *term-term* penting yang sejalan dengan *term-term* yang sebelumnya telah diekstrak dari dokumen dan menghitung relevansi antara *query* dan dokumen berdasarkan pada *term-term* tersebut. Sebagai hasilnya, sistem mengembalikan suatu daftar dokumen terurut *descending* (*ranking*) sesuai nilai kemiripannya dengan *query* pengguna[4].

Setiap dokumen (termasuk *query*) direpresentasikan menggunakan model *bag-of-words* yang mengabaikan urutan dari kata-kata di dalam dokumen, struktur sintaktis dari dokumen dan kalimat. Dokumen ditransformasi ke dalam suatu “tas“ berisi kata-kata *independen*. *Term* disimpan dalam suatu *database* pencarian khusus yang ditata sebagai sebuah *inverted index*. Indeks ini merupakan konversi dari dokumen asli yang mengandung sekumpulan kata ke dalam daftar kata yang berasosiasi dengan dokumen terkait dimana kata-kata tersebut muncul.

2.4 *Stemming*

Stemming merupakan suatu proses pembuangan atau pemotongan afiks (baik prefiks maupun sufiks) untuk mendapatkan bentuk akar atau dasar dari suatu *term* (berupa kata). Dalam konteks *information retrieval*, *stemming* digunakan untuk mereduksi bentuk *term* untuk menghindari ketidakcocokan. Hal ini dikarenakan afiks

dapat mengandung informasi seperti bagian dari percakapan, *plurarity*, dan sebagainya. Perlu diingat bahwa proses *stemming* di sini bukan merupakan pekerjaan dalam *etimologi grammar*, sehingga dapat ditoleransi apabila algoritma *stemming* menghasilkan kata yang tidak bermakna. Algoritma *stemming* dapat dibedakan menjadi *context-free* dan *context-sensitive*. *Context-free* membuang akhiran tanpa adanya suatu batasan, sedangkan *context-sensitive* melibatkan banyak batasan kontekstual untuk mencegah pembuangan akhiran yang mengakibatkan *stem* yang dihasilkan menjadi rusak.

2.4.1 *Stemming* Bahasa Indonesia

Morfologi adalah bagian dari ilmu bahasa yang membicarakan atau mempelajari seluk-beluk bentuk kata serta pengaruh perubahan bentuk kata terhadap golongan dan arti kata. Kata yang dibentuk dari kata lain pada umumnya mengalami tambahan bentuk pada kata dasarnya. Perubahan-perubahan bentuk kata menyebabkan adanya perubahan golongan dan arti kata. Tiga macam proses morfologis, yaitu pertama, bergabungnya morfem bebas dengan morfem terikat disebut afiksasi. Kedua, pengulangan morfem bebas disebut reduplikasi, dan ketiga, bergabungnya morfem bebas dengan morfem bebas disebut pemajemukan. Pada proses yang pertama menghasilkan kata berimbuhan, yang kedua menghasilkan kata ulang, dan yang ketiga menghasilkan kata majemuk. Imbuhan (afiks) adalah bentuk (morfem) terikat yang dipakai untuk menurunkan kata.Pada umumnya imbuhan (afiks) hanya dikenal ada empat, yaitu awalan (prefiks), sisipan (infiks), akhiran (sufiks), awalan dan akhiran (konfiks)[1].

Penggunaan afiks dalam teks berbahasa Indonesia termasuk dalam kategori yang kompleks. Afiks yang dimaksud dalam konteks ini adalah prefiks (awalan), sufiks (akhiran), infiks (sisipan), konfiks, bentuk perulangan, dan kombinasi dari afiks-afiks tersebut. Penghapusan afiks untuk mengubah sebuah kata menjadi kata dasarnya, dapat menggunakan *rule-rule* tertentu yang dipertimbangkan secara cermat agar dapat

menghasilkan proses *stemming* yang baik. Efektifitas algoritma *stemming* dipengaruhi oleh beberapa faktor[8]:

a. Kesalahan dalam proses pemenggalan imbuhan dari kata dasarnya. Kesalahan ini dapat berupa:
    1. *Overstemming*: yaitu pemenggalan imbuahan yang melebihi dari yang seharusnya. Contoh: kata masalah menjadi masa. Kesalahan ini dapat timbul karena bentuk kata dasar yang menyerupai imbuhan.
    2. *Understemming*: yaitu pemenggalan imbuhan yang terlalu sedikit dari yang seharusnya. Contoh: kata belajar menjadi lajar. Kesalahan ini dapat timbul karena kekurangan pada aturan pola imbuhan yang didefinisikan.
    3. *Unchange*: yaitu kasus khusus dari *understemming*, dimana tidak terjadi pemenggalan imbuhan sama sekali. Contoh: telapak, yang memiliki kata dasar tapak tetapi setelah pemenggalan kata dasar yang didapat tetap telapak. Kesalahan ini dapat ditimbulkan karena kekurangan pada aturan pola imbuhan yang didefinisikan.
    4. *Spelling exception*: yaitu huruf pertama kata dasar yang didapat tidak benar yang diakibatkan dari pemenggalan awalan. Contoh: kata memukul menjadi ukul. Kesalahan ini dapat ditimbulkan karena ada beberapa imbuhan yang berubah bentuk ketika ditempelkan pada suatu kata dasar. Misalnya awalan beR-, meN-, teR-, peR-, akan bergantung pada huruf pertama kata dasar dimana imbuhan tersebut ditempelkan (Contoh: ber- + ajar = belajar, pen- + lihat = penglihatan, pen- + sakit = penyakit). Atau sebaliknya ada imbuhan yang mengakibatkan huruf pertama kata dasar yang ditempelinya menjadi luluh. Misalnya meng- / peng- meluluhkan huruf ‘k’ (Contoh: mengarang dari meng- dan karang) atau men- / pen- meluluhkan huruf ‘p’ (Contoh: menuai dari men- dan tuai).

b. Kekurangan dalam perumusan aturan penambahan imbuhan pada kata dasar. Hal ini dapat terjadi karena morfologi bahasa Indonesia yang kompleks, sehingga sangat sulit atau bahkan tidak mungkin untuk merumuskan aturan yang sempurna.
c. Jumlah total aturan imbuhan yang didapat berhubungan dengan efektifitas proses temu kembali. Dimana semakin banyak pola penambahan imbuhan yang dapat dirumuskan, maka proses temu kembali akan semakin efektif.

Untuk kasus *stemming* bahasa Indonesia ada beberapa pendekatan yang digunakan. Teknik pendekatan untuk proses *stemming* tersebut, ada yang menggunakan kamus, namun ada juga yang tidak menggunakan kamus. Contoh teknik *stemming* yang tidak menggunakan kamus adalah algoritma Vega, sedangkan *stemming* yang menggunakalan kamus adalah algoritma Nazief dan Adriani tahun 1996, algoritma Idris tahun 2001 dan yang terakhir algoritma *Enhanced Confix Stripping Stemmer* tahun 2008. Algoritma *Enhanced Confix Stripping Stemmer* dikembangkan oleh Putu Adhi Kerta Mahendra pada tahun 2008 sebagai hasil evaluasi penelitian sebelumnya, yaitu memperbaiki kekurangan pada algoritma *Confix Stripping Stemmer*. Perbaikan dilakukan dengan menambah dan memodifikasi tabel aturan pemenggalan, lalu menambah fungsionalitas pengembalian akhiran untuk mengantisipasi kesalahan pemenggalan yang seharusnya tidak dilakukan.

Secara struktur morfologi Bahasa Indonesia, proses *stemming* Bahasa Indonesia dapat dilakukan hal-hal berikut :

1. Pembuangan partikel

Dalam proses ini, dapat dilakukan pembuangan partikel, seperti –lah, -kah, dan –pun. Contohnya adalah adalah, siapakah, dan walaupun dapat dilihat pada tabel 2.1 dibawah ini :

Tabel 2.1 Pembuangan Partikel

Kata awal	Partikel	Kata setelah *stemming*
apalah	-lah	apa
diakah	-kah	dia

siapapun	-pun	siapa

2. Pembuangan *possessive pronoun*

Pembuangan *possessive pronoun* adalah penghilangan suatu kata yang mengacu pada kepemilikan sesuatu. Contohnya adalah kau-, ku-, -ku, -mu, dan –nya yang dapat dilihat pada tabel 2.2 dibawah ini :

Tabel 2.2 Pembuangan *Possesive Pronoun*

Kata awal	*Posessive pronoun*	Kata setelah *stemming*
kaurasa	kau	rasa
kusinggah	ku-	singgah
rumahku	-ku	rumah
padamu	-mu	pada
segalanya	-nya	segala

3. Pembuangan *circumfix*

Dalam proses *stemming* ini akan dilakukan pembuangan *circumfix* (kombinasi awalan dan akhiran). Berikut adalah contoh dari *circumfix* yang dapar dilihat pada tabel 2.3 dibawah ini :

Tabel 2.3 Pembuangan *circumfix*

Kata awal	*Circumfix*	Kata setelah *stemming*
bertahtakan	ber – kan	tahta
dihujani	di – i	hujan
dihapuskan	di – kan	hapus
kecepatan	ke – an	cepat
memperbarui	memper – i	baru
mempertanyakan	memper – kan	tanya
mempunyai	mem – i	punya
menumpahkan	me – kan	tumpah

melawannya	me – nya	lawan
persatui	per – i	satu
pertaruhan	per – an	taruh
pembelajaran	pe – an	belajar

4. Pembuangan prefiks

Prefiks adalah nama lain dari awalan. Pada proses *stemming* Bahasa Indonesia akan dilakukan pemotongan prefiks. Berikut ini beberapa contoh prefiks yang dilihat pada tabel 2.4 dibawah ini :

Tabel 2.4 Pembuangan Prefiks

Kata awal	Prefiks	Kata setelah *stemming*
berusaha	ber-	usaha
dicoba	di-	coba
kemana	ke-	mana
memperkuat	memper-	kuat
merayu	me-	rayu

5. Kombinasi Terlarang

Kombinasi terlarang adalah kombinasi antara awalan dan akhiran yang menyebabkan kata dasar mendapat bentuk baru ketika diberi imbuhan tersebut tetapi menjadi kata yang tidak baku. Berikut adalah kombinasi terlarang dapat dilihat pada tabel 2.5 dibawah ini :

Tabel 2.5 Kombinasi awalan-akhiran yang tidak diperbolehkan

Awalan	Akhiran
be-	-i
di-	-an
ke-	-i,-kan
me-	-an
se-	-i,-kan
te-	-an

2.4.2 Algortima Idris

Algoritma Idris merupakan pengembangan dari algoritma sebelumnya, yaitu algoritma Ahmad, Yusoff, dan Sembok yang awalnya dikembangkan untuk Bahasa Melayu[9]. Algoritma ini memperluas skema algoritma Ahmad, Yusuf dan Sembok untuk *stemming* dan *recoding* dalam dokumen teks Bahasa Melayu. Kemiripannya dengan dokumen teks Bahasa Indonesia, membuat algoritma Idris dapat diterapkan juga untuk dokumen teks berbahasa Indonesia. Algoritma ini menghapus prefiks dan sufiks sampai akar kata ditemukan dalam kamus. Dalam tugas akhir ini, algoritma Idris mengadopsi asumsi Arifin dan Setiono bahwa kata-kata bahasa Indonesia dapat memiliki paling banyak dua awalan (prefiks) dan tiga akhiran (sufiks)[10]. Aturan-aturan (*rules*) dalam algoritma Idris, diadopsi dari *rules* pada algoritma Othman, yaitu[11] :

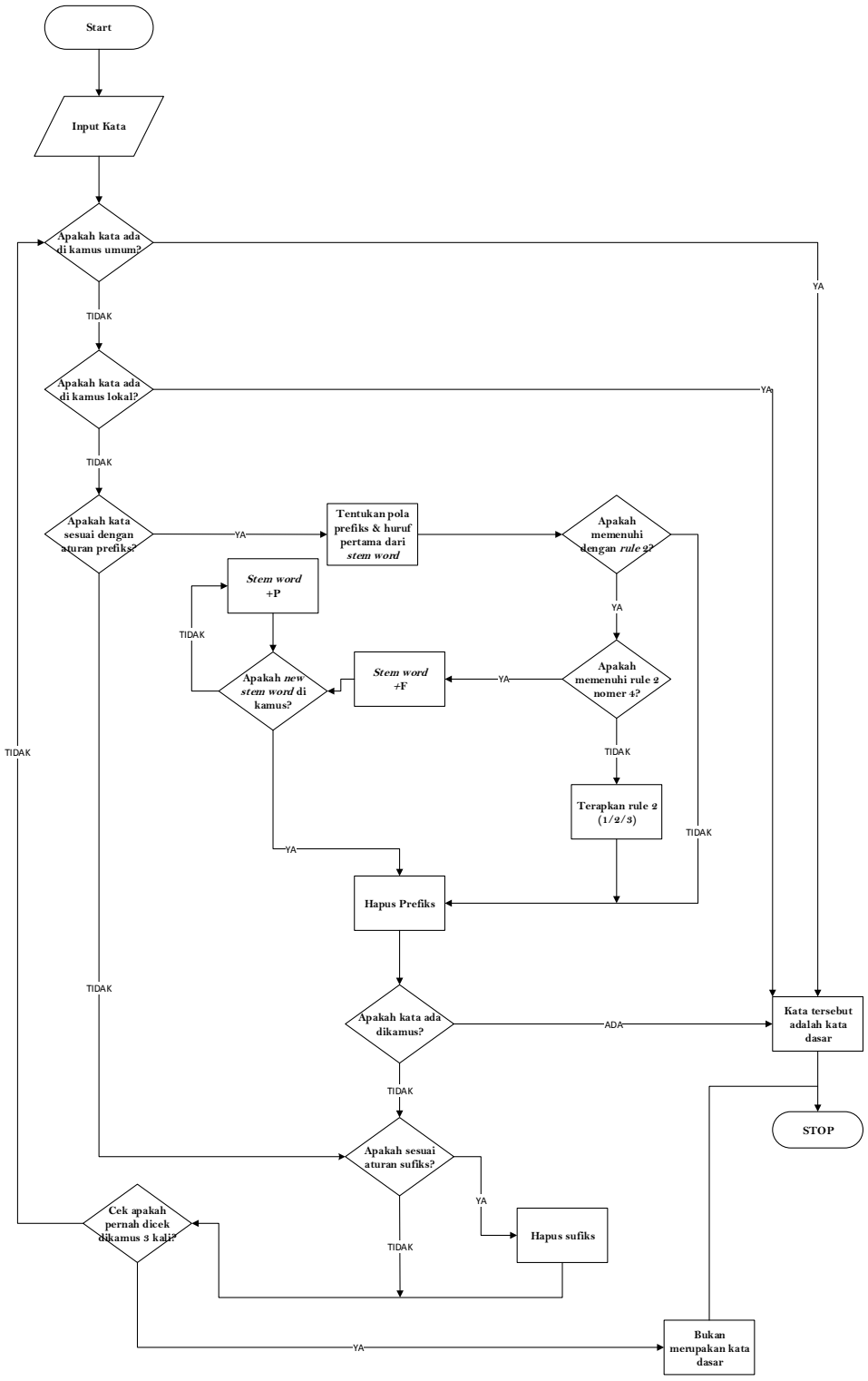
(1) Prefiks rules format : Prefiks+
    Contoh : di + jajah → dijajah
(2) Suffiks rules format : +Suffiks
    Contoh : jajah + an → jajahan
(3) Infix rules format : +Infix+
    Contoh : tapak (+el+) → telapak
(4) Prefiks-Suffiks pair rules format : Prefiks+Suffiks
    Contoh : meN + takluk → menakluki

Jika dalam kenyataannya, hasil *stemming* dari *rules* di atas tidak lengkap, maka cek huruf pertama dari kata tersebut. Aturan ini biasa disebut Rule 2. Jika huruf pertama adalah huruf hidup (vokal), maka :

(1) Tambahkan *t* setelah menghilangkan imbuhan *men-* atau *pen-*
(2) Tambahkan *k* setelah menghilangkan imbuhan *meng-* atau *peng-*
(3) Tambahkan *s* setelah menghilangkan imbuhan *meny-* atau *peny-*
(4) Tambahkan *f* atau *p* setelah menghilangkan imbuhan *mem-* atau *pem-*

Fitur dari algoritma ini adalah menggunakan dua buah kamus: kamus umum dan kamus khusus untuk konten dokumen, misalnya yang mengandung istilah medis atau hukum[4]. Dua varian dari algoritma ini, prefiks diubah sebelum *recoding* dan prefiks diubah setelah *recoding*. Asumsi yang digunakan adalah kata-kata yang digunakan mempunyai prefiks yang berbeda. Sebagai contoh, kata "memasukkan" yang berasal dari kata "masuk", dapat menjadi "mem-asuk-kan" atau "me masukkan". Varian kedua, mengecek *recoding* dahulu. Contoh, setelah menghapus prefiks "mem-", diperoleh "asuk" yang tidak ada dalam kamus. Dalam aturan *recoding*, dapat diketahui bahwa prefiks "mem-", huruf "p" dapat dihilangkan, sehingga menambahkan huruf "p" menjadi kata yang salah dan tidak berada dalam kamus "pasuk".

Penghapusan afiks yang salah dapat menyebabkan *overstemming*. Contoh, untuk kata "medannya" yang berasal dari kata "medan" (daerah, lapangan, atau tempat). Dalam algoritma ini, prefiks "me-" akan dihapus menjadi "dannya" dimana kata tersebut tidak berada dalam kamus. Maka, algoritma akan mengecek dan menghapus akhirannya (sufiks) "-nya" menjadi "dan". Kata tersebut memang benar berada dalam kamus, namun kata tersebut bukan kata yang dimaksud. Karena didesain untuk Bahasa Melayu, algoritma ini menggunakan prefiks dan sufiks yang sedikit berbeda dari Bahasa Indonesia, yang dapat berpeluang menjadi *overstemming*. Secara garis besar, skema algoritma Idris dapat dilihat pada gambar 2.4.



Gambar 2.4 Skema Algortima Idris[4]

Berdasarkan skema pada gambar 2.4 dapat dijelaskan bahwa skema dari Algoritma Idris adalah :

(1) Cek kata yang dicari dalam kamus umum. Jika kata tersebut ada dalam kamus umum,maka kata tersebut merupakan kata akar (kata dasar) dan keluar dari algoritma. Jika tidak ada, maka dilanjutkan ke langkah (2).

(2) Cek kata yang dicari dalam kamus khusus. Jika kata tersebut ada dalam kamus khusus, maka kata tersebut merupakan kata akar (kata dasar) dan keluar dari algoritma. Jika tidak ada, maka dilanjutkan ke langkah (3).

(3) Cek kata pada aturan *prefiks*. Jika kata tersebut cocok dengan aturan *prefiks*, cek pola dari *prefiks* tersebut dan huruf pertama dari kata yang akan di-*stem*. Jika tidak cocok, makan dilanjutkan ke langkah (4).

(4) Jika pola *prefiks* pada kata cocok dengan *Rules* 2, maka ubah kata tersebut sesuai dengan *rule*. Jika tidak cocok, maka hilangkan *prefiks* dan lanjutkan ke langkah (7).

(5) Cek *prefiks* dari kata tersebut, cocokkan polanya pada *rules* 2. Jika cocok dengan keempat *rule* tersebut, lalu cek kata yang telah di-stem pada kamus dan lanjutkan ke langkah (6). Jika tidak cocok, maka hilangkan prefiks dan lanjutkan ke langkah (7).

(6) Cek prefiks dari kata tersebut, cocokkan polanya pada *rules* 2. Jika cocok dengan keempat *rule* tersebut, lalu cek kata yang telah di-*stem* pada kamus dan lanjutkan ke langkah (6). Jika tidak cocok, maka hilangkan prefiks dan lanjutkan ke langkah (7). Jika kata tersebut tidak ada pada kamus, maka kembali ke langkah (5). Jika cocok, maka hilangkan prefiks dan lanjutkan ke langkah (7).

(7) Cek kata telah cocok pada kamus, lalu kata tersebut merupakan kata dasar, lalu keluar dari algoritma. Jika tidak cocok, maka lanjutkan ke langkah (8).

(8) Cek kata pada aturan suffiks. Jika cocok dengan aturan tersebut, maka hilangkan suffiks dan lanjutkan ke langkah (1). Jika tidak cocok, lanjutkan juga ke langkah (1).

Rule-rule yang digunakan pada algortima Idris dapat dilihat pada tabel 2.6 berikut :

Tabel 2.6 Tabel *Rule* Algoritma Idris

Kelas Affiks	*Affixes*	Contoh kata
Prefiks	di-	dirantau
	ke-	kemana
	ber-	berlari
	men-	mencari
	ter-	terdapat
	pen-	pendapat
	per-	pertanda
Suffiks	-an	jalanan
	-i	jalani
	-kan	jalankan
	-nya	punyanya
	-lah	apalah
	-kah	inikah
Prefiks - Suffiks	ber – an	bertaburan
	ber – kan	berdasarkan
	di – i	dijauhi
	ke – an	kedatangan
	men – kan	mendengarkan
	men – i	mendapati
	memper – i	mempertanyai
	memper – kan	memperdengarkan
	per – an	perdagangan
	se – nya	seharusnya
Infiks	el	telapak
	em	gelembung

	er	serabut
	in	kinerja

Langkah-langkah yang dilakukan oleh algoritma Idris adalah :

(1) a. Kata yang belum di-*stemming* dicari pada kamus umum atau KBBI. Jika kata itu langsung ditemukan, berarti kata tersebut adalah kata dasar. Kata tersebut dikembalikan dan algoritma dihentikan.

b. Kata yang belum di-*stemming* dicari pada kamus lokal. Jika kata itu langsung ditemukan, berarti kata tersebut adalah kata dasar. Kata tersebut dikembalikan dan algoritma dihentikan.

(2) *Derivational prefix* kemudian dihilangkan. Langkah ini terus dilakukan sampai tidak ada lagi *derivational prefix*. Jika tidak ada lagi, maka lanjutkan ke langkah berikutnya. Untuk beberapa kombinasi imbuhan, dilakuan penghilangan prefiks terlebih dahulu. Yaitu pada kombinasi imbuhan “berlah”, “ber-an”, “men-i”, “di-i”, “pe-i”, “ter-i”.

(3) Hilangkan *Inflectional suffix* terlebih dahulu. Jika hal ini berhasil dan suffiks adalah partikel (“lah” dan ”kah”), langkah ini dilakukan lagi untuk menghilangkan *inflectional possessive pronoun suffix* (“ku”, “mu” atau ”nya”).

(4) *Derivational suffix* kemudian dihilangkan. Langkah ini terus dilakukan sampai tidak ada lagi *derivational suffix*. Jika tidak ada lagi, maka lanjutkan ke langkah berikutnya.

(5) Setelah setiap penghilangan imbuhan dilakukan, maka lakukan pengecekan menggunakan kamus. Jika kata ditemukan, maka algoritma berhenti dan tidak perlu dilakukan pengecekan terhadap imbuhan lainnya. Jika sampai selesai penghilangan imbuhan masih belum menemukan kata dasar, maka dilakukan *recoding*.

(6) Jika semua langkah sudah dilakukan termasuk *recoding* dan tidak juga ditemukan dalam kamus, maka algoritma ini akan mengembalikan kata yang

2.4.3 Algoritma *Enhanced Confix Stripping (ECS) Stemmer*

Stemming dapat dimanfaatkan untuk berbagai sistem yang membutuhkan *term preprocessing* sebagai awal prosesnya. Perkembangan *stemming* bermula pada algortima Porter Stemmer tahun 1980, lalu Algoritma Nazief-Adriani tahun 1996, *Confix Stripping Stemmer* tahun 2007 dan terakhir *Enhanced Confix Stripping Stemmer* tahun 2008. Algoritma *Enhanced Confix Stripping Stemmer* dikembangkan oleh Putu Adhi Kerta Mahendra pada tahun 2008 sebagai hasil evaluasi penelitian sebelumnya, yaitu memperbaiki kekurangan pada algoritma *Confix Stripping Stemmer*. Perbaikan dilakukan dengan menambah dan memodifikasi tabel aturan pemenggalan, lalu menambah fungsionalitas pengembalian akhiran untuk mengantisipasi kesalahan pemenggalan yang seharusnya tidak dilakukan. Aturan morfologi Bahasa Indonesia mengelompokan imbuhan menjadi beberapa kategori sebagai berikut[5]:

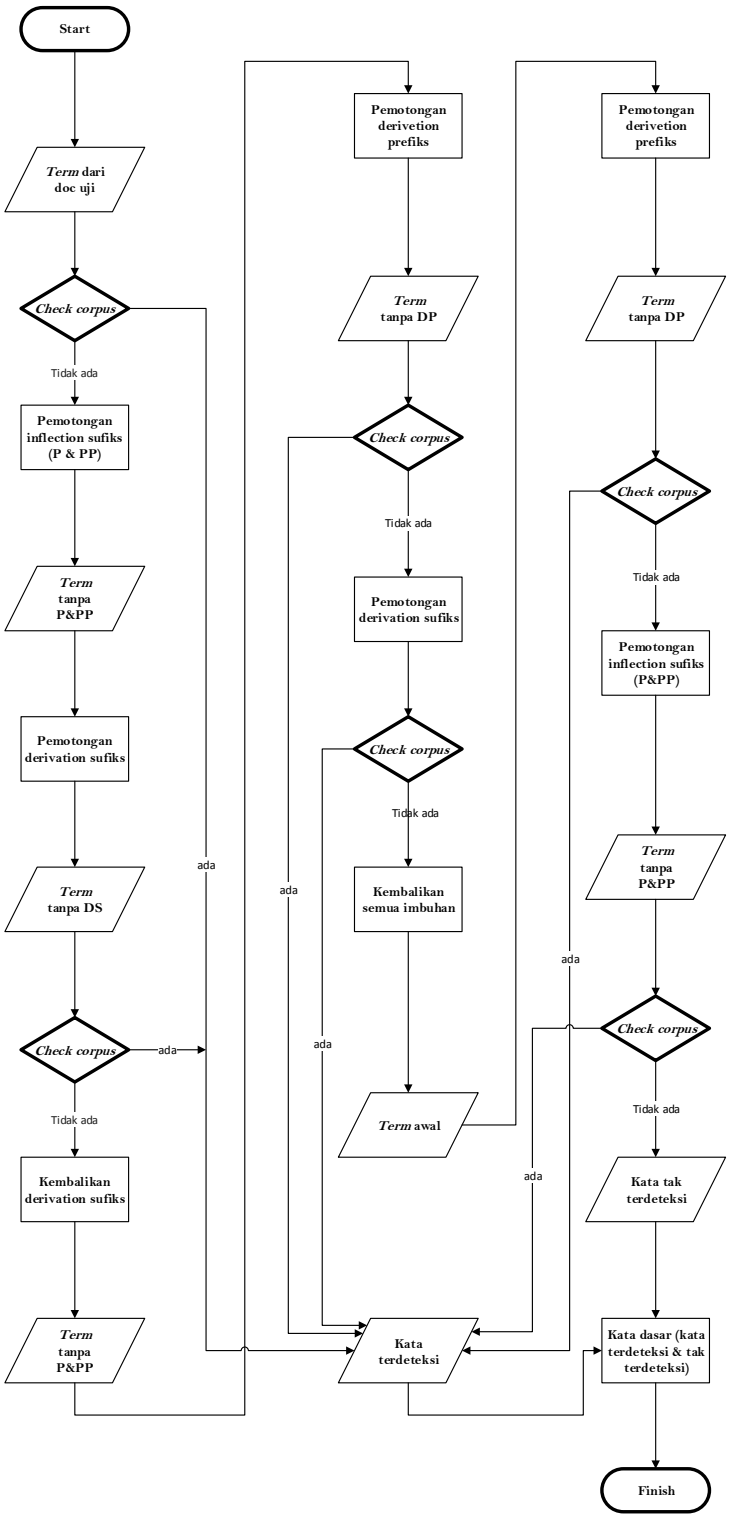
1. *Inflection Suffixes* (IS), yaitu kelompok-kelompok akhiran yang tidak mengubah kata dasar. Kelompok akhiran ini dapat dibagi lagi menjadi 2, yaitu :
   a. *Particle* (P), atau partikel, diantaranya "-lah", "-kah", "-tah" dan "-pun".
   b. *Possessive Pronoun* (PP), atau akhiran yang menambahkan arti sebagai kepunyaan, diantaranya "-ku", "-mu" dan "-nya".
2. *Derivation Suffixes* (DS), yaitu kumpulan akhiran yang secara langsung ditambahkan pada kata dasar, diantaranya "-i", "-kan", dan "-an".
3. *Derivation Prefixes* (DP), yaitu kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Berdasarkan perubahan bentuknya awalan dibagi menjadi 2 :
   a. Awalan yang bermorfologi, diantaranya "me-", "be-", "pe-", dan "te-"
   b. Awalan yang tidak bermorfologi , diantaranya "di-", "ke-" dan "se-".

Gabungan awalan dan akhiran menjadi imbuhan membentuk kata dasar menjadi kata baru yang memiliki makna berbeda dari kata dasar sebelumnya, tetapi tidak semua kombinasi diperbolehkan. Kombinasi yang tidak diperbolehkan diantaranya "be- -

i”,”di- -an”,”ke- -i”, “ke- -kan”, “me- -an”, “se- -i”, “se- -kan”, dan “te- -an”[5]. Kombinasi terlarang menyebabkan kata dasar menjadi kata baru yang tidak baku. Dengan model kata bahasa Indonesia, aturan-aturan dasar morfologi pada bahasa Indonesia serta pengklasifikasian imbuhan-imbuhan seperti di atas, maka bentuk kata berimbuhan dalam bahasa Indonesia dapat dimodelkan sebagai berikut :

[DP+ [DP+ [DP+]]] Kata Dasar [[+DS] [+PP] [+P]]

Berikut adalah tahapan algoritma *Enhanced Confix Stripping Stemmer* yang dapat dilihat pada gambar 2.5.



Gambar 2.5 Skema Algoritma *Enhanced Confix Stripping Stemmer*[5]

Dari gambar 2.5 dapat diuraikan bahwa skema algortima *Enhanced Confix Stripping Stemmer* adalah sebagai berikut[5] :

1. Kata dalam dokumen uji sudah mengalami proses *preprocessing* dan berbentuk kata yang tunggal. Lakukan pengecekan terhadap kata, cari kata dalam kamus, jika kata ditemukan sebagai kata yang valid dalam kamus maka kata tesebut dianggap sudah berbentuk kata dasar, jika tidak maka lakukan langkah 2.
2. Hapus *Inflection Suffixes* . Akhiran yang pertama dihilangkan adalah akhiran yang termasuk *Particles* (P) yaitu "-lah", "-kah", "-tah" atau "-pun". Setelah itu langkah selanjutnya penghilangan akhiran yang termasuk *Possesive Pronouns* (PP) yaitu "-ku", "-mu" atau "-nya".
3. Hapus *Derivation Suffixes* (DS) yaitu "-i", "-kan", atau "-an".
4. Hapus *Derivation Prefixes* (DP) yaitu "di-", "ke-", "se-","me-", "be-", "pe-", atau "te-". Langkah empat memiliki keadaan khusus dimana jika mengalami keadaan khusus maka langkah tambahan dilakukan. Langkah tambahan adalah sebagai berikut:

a. Langkah 4 akan terhenti jika penghilangan awalan sudah 3 kali.
b. Proses penghilangan DP:

- Standar, penghilangan awalan yang tidak bermorfologi, awalan dapat langsung dihilangkan karena tidak merubah kata dasar.
- Kompleks, penghilangan awalan yang bermorfologi, awalan yang mengalami perubahan bentuk sesuai dengan huruf awal kata dasar, penghilangan awalan sesuai dengan tabel aturan pemenggalan imbuhan.

c. Cek kombinasi terlarang. Jika kata memiliki kombinasi terlarang maka proses stemming berhenti dan kata dikelompokkan menjadi kata tak terdeteksi karena kata tersebut tidak baku.
d. Cari kata dalam kamus, apabila ditemukan maka proses dihentikan dengan status kata sudah ditemukan, apabila tidak ditemukan maka langkah 4 diulang kembali

sebanyak 3 kali. Jika proses berhenti tetapi kata belum ditemukan maka dilanjutkan pada langkah 5.

5. Lakukan *Recoding,* proses mengacu pada tabel aturan pemenggalan imbuhan. *Recoding* adalah penambahan karakter di awal kata yang dipenggal, karakter *recoding* adalah karakter serelah tanda hubung (“-“) atau bentuk lain setelah tanda (“ | “). Sebagai contoh kata “menunda” setelah dipenggal menjadi “nunda”, karena kata “nunda” tidak valid maka di*recoding* sehingga kata “nunda” menjadi “tunda” dan valid ketika dicek pada kamus kata dasar.
6. Menambahkan suatu fungsionalitas tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Fungsionalitas ini disebut *loop Pengembalian Akhiran*. Fungsionalitas *Loop Pengembalian Akhiran* dideskripsikan sebagai berikut:

a. Kembalikan seluruh awalan yang telah dihilangkan sebelumnya, sehingga menghasilkan model kata seperti berikut:

   [DP+[DP+[DP]]] + Kata Dasar

   Pemenggalan awalan dilanjutkan dengan proses pencarian di kamus kemudian dilakukan pada kata yang telah dikembalikan menjadi model tersebut.
b. Kembalikan akhiran sesuai dengan urutan model pada bahasa Indonesia. Ini berarti bahwa pengembalian dimulai dari DS (“-i”, “- kan”, “-an”), lalu PP (“-ku”, “-mu”, “-nya”), dan terakhir adalah P (“- lah”, “-kah”, “-tah”, “-pun”). Khusus untuk akhiran “-kan”, pengembalian pertama dimulai dengan “k”, baru kemudian dilanjutkan dengan “an”.
c. Lakukan pengecekan di kamus kata dasar. Apabila ditemukan, proses dihentikan. Apabila gagal, maka lakukan proses pemenggalan awalan berdasarkan tabel aturan pemenggalan imbuhan.
d. Lakukan *recoding* apabila diperlukan.

e. Apabila semua langkah diatas telah dilakukan lalu kata tidak ditemukan setelah pengecekan pada kamus, maka kata dikembalikan dengan semua imbuhan dan kata dimasukkan dalam kelompok kata tak terdeteksi.

7. Jika kata tidak ditemukan setelah semua langkah dilakukan maka kata tersebut dikelompokan pada kata tak terdeteksi. Pada tabel 2.6 dapat dilihat aturan ECS.

Tabel 2.7 Aturan Pemenggalan Awalan *Stemming ECS Stemmer* [5]

Aturan	Format kata	Pemenggalan
1.	berV...	ber-V... | be-rV...
2.	berCAP...	ber-CAP... dimana C!=”r”& P!=”er”
3.	berCAerV...	ber-CaerV... dimana C!=”r”
4.	Beajar	bel-ajar
5.	beClerC2...	be-C!erV... dimana C1!={“r”|”1”}
6.	terV...	ter-V... | te-rV...
7.	terCerV...	ter-CerV.. dimana C!=”r”
8.	terCP...	terCP... dimana C!=”r” dan P!=”er”
9.	terC!erC2...	te-ClerC2... dimana C1!=”r”
10.	me{1|r|w|y|}V...	me-{1|r|w|y|}V...
11.	mem{b|f|v}...	mem-{b|f|v}...
12.	mempe...	mem-pe...
13.	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14.	men{c|d|j|t|z}...	men-{c|d|j|t|z}...
15.	menV...	me-nV... | me-tV...
16.	meng{g|h|q|k}...	meng-{g|h|q|k}...
17.	mengV...	me-ngV.. | meng-V... | meng-kV... | mengV-…jika v=”e”
18.	menyV...	me-nyV... | meny-sV...
19.	mempA...	mem-pA dimana A!=”e”
20.	pe{w|y}V...	pe-{w|y}V...
21.	perV...	per-V... | pe-rV...
22.	perCAP...	per-CAP... dimana C!=”r” dan P!=”er”

23.	perCAerV...	per-CAerV... dimana C!=”r”
24.	pem{b|f|v}...	pem-{b|f|v}...
25.	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26.	pen{c|d|j|z}...	pen-{c|d|j|z}...
27.	penV...	pen-V... | pe-tV...
28.	pengC...	peng-C...
29.	pengV...	Pe-ngV..| peng-V…| peng-kV…| pengV-…jika V=”e”
30	penyV...	pe-nyV...| peny-sV...
31.	pelV...	pe-1V... kecuali “pelajar” yang menghasilkan “ajar”
32.	peCerV...	peC-erV... dimana C!={r|w|y|1|m|n}
33.	peCP...	pe-CP... dimana C!={r|w|y|1|m|n} dan P!=”er”
34.	terC1erC2..	ter-C1erC2.. dimana C1!=”r”
35.	peC1erC2..	pe-C1erC2.. dimana C!={r|w|y|1|m|n}
36.	CerV...	CerV...| CV...
37.	CelV...	CelV...| CV...
38.	CemV...	CemV...|CV...

Keterangan dari tabel 2.7 adalah :

Huruf “C” menandakan huruf konsonan

Huruf “V” menandakan huruf vokal

Huruf “A” menandakan huruf apa saja

Huruf “P” menandakan fragmen kata yang pendek seperti “er”

2.5 Kekuatan *Stemmer*

Untuk mengukur performa kompresinya yang digunakan adalah nilai icf (*index compression factor*) dan persentase. Icf merepresentasikan tingkat dimana sebuah koleksi kata direduksi oleh *stemming*. Persentase adalah ukuran dari keberhasilan *stem* kata dalam bentuk % . Jika *conflation class* ada 10000 kata yang berbeda dihasilkan 2500 *distinct stem*, maka icf = 10000-2500/10000 = 0,75 (75%). Semakin rendah nilai-

nilai icf mengindikasikan sebuah *stemmer* yang bagus dalam melakukan kompresi *term* yang akan dimasukkan ke dalam *index database*[10].

Berikut rumus matematisnya[10] :

$$Persentase = \frac{N1}{S1} * 100$$

$$Icf = \frac{N-S}{N}$$

Keterangan :

N = menunjukkan jumlah kata sebelum dilakukan *stemming.*

S = menunjukkan jumlah kata setelah dilakukan *stemming.*

N1= menunjukkan jumlah huruf pada saat akan dilakukan *stemming.*

S1 = menunjukkan jumlah kata pada saat akan dilakukan *stemming*.

2.6 UML (*Unified Modeling Language*)

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. *Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dam mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat, ini merupakan standar

terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

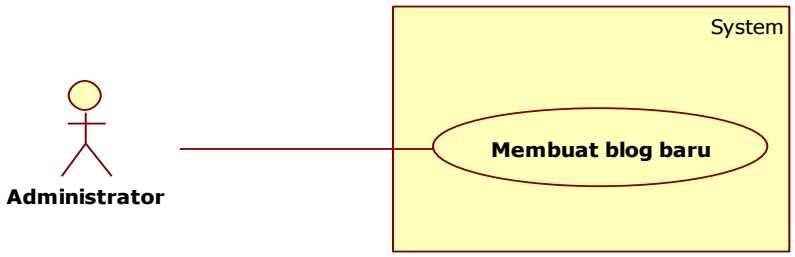
Dalam UML terdapat banyak diagram yang digunakan untuk melakukan pemodelan, disini penulis hanya menjelaskan beberapa diagram sesuai yang digunakan pada tahap perancangan. Berikut diagram-diagram yang ada pada UML, antara lain :

a. *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk u*se case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*. Berikut contoh *use case diagram* :

Gambar 2.6 Contoh *Use Case Diagram*



Pada gambar 2.6 tersebut *administrator* merupakan *actor*, bisa dibilang *user* atau pengguna sistem. Kemudian garis tersebut merupakan garis asosiasi yang menghubungkan *user* dengan *fungsional* sistem. Gambar yang membentuk seperti persegi yang bertuliskan sistem disudut kanan atas merupakan sistem yang dimiliki dan yang berbentuk oval bertuliskan “membuat blog baru” merupakan *use case* atau *fungsiona*l atau menu yang terdapat pada sistem.

b. *Activity Diagram*

Diagram aktivitas atau dalam bahasa inggris *Activity Diagram* adalah representasi grafis dari seluruh tahapan alur kerja. Diagram ini mengandung aktivitas, pilihan tindakan, perulangan dan hasil dari aktivitas tersebut.

c. *Class Diagram*

Class Diagram adalah diagram yang menunjukan kelas - kelas yang ada dari sebuah sistem dan hubungannya secara logika. *Class diagram* menggambarkan struktur statis dari sebuah sistem. Karena itu *class diagram* merupakan tulang punggung atau kekuatan dasar dari hampir setiap metode berorientasi objek termasuk UML.

d. *Sequence Diagram*

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek – obyek yang melakukan suatu tugas atau aksi tertentu. Obyek – obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya diletakkan dipaling kiri dari diagram. Pada diagram ini, dimensi vertikal merepresentasikan waktu. Bagian paling atas dari diagram menjadi titik awal dan waktu berjalan ke bawah sampai dengan bagian dasar dari diagram. Garis vertikal disebut *lifeline,* dilekatkan pada setiap obyek atau aktor. Kemudian *lifeline* tersebut digambarkan menjadi kotak ketika obyek melakukan suatu operasi , kotak tersebut disebut *activation box*. Obyek dikatakan mempunyai *live activation* pada saat tersebut. Pesan yang dipertukarkan antar obyek digambarkan sebagai sebuah anak panah antara *activation box* pengirim dan penerima. Kemudian diatasnya diberikan label pesan[13].

2.7 C# (C Sharp)

C# (dibaca "*See-Sharp*") adalah bahasa pemrograman baru yang diciptakan oleh Microsoft (dikembangkan dibawah kepemimpinan Anders Hejlsberg yang notabene juga telah menciptakan berbagai macam bahasa pemrograman termasuk Borland Turbo C++ dan Borland Delphi). Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa

digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *windows (desktop)* dan aplikasi berbasis *web* serta aplikasi berbasis *web services*[12].

2.8 Microsoft Visual Studio

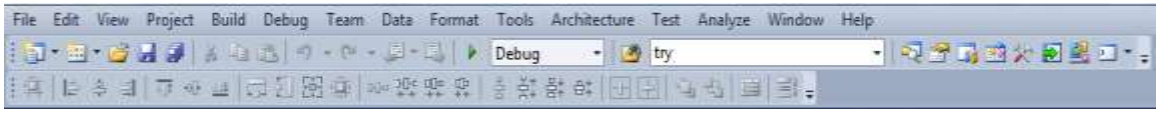
Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi Windows, ataupun aplikasi *Web*. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN *Library*). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe[12].

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas *Windows*) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*).

Untuk membuat suatu program aplikasi pada visual studio, maka diperlukan suatu struktur aplikasi atau komponen yang digunakan, berikut beberapa aplikasi yang penulis gunakan pada visual studio antara lain :

a. *Toolbar*

Terdiri atas beberapa komponen yang digunakan untuk membuat objek pada *form* dan dapat mempercepat pengaksesan perintah-perintah yang ada dalam pemrograman, berikut gambar *toolbar visual studio* seperti gambar 2.7 dibawah ini.



Gambar 2.7 *ToolBar Visual Studio*

b. *Toolbox*

Toolbox Adalah sebuah jendela yang berisi tombol kontrol yang akan *user* gunakan untuk mendesain atau membangun sebuah *form* atau *report*. *Toolbox* terdiri atas beberapa tombol untuk mengendalikan tampilan, berikut gambar *toolbox* pada *visual studio* seperti gambar 2.8 dibawah ini.



Gambar 2.8 Tampilan *Toolbox Visual Studio*

c. Jendela *Properties* Jendela *properties* adalah jendela yang berfungsi sebagai tempat meng*edit* properti pada suatu objek yang terpilih pada saat mendesain atau membangun aplikasi bagi *user*. Berikut gambar jendela *properties*, seperti pada gambar 2.9 dibawah ini.



Gambar 2.9 Tampilan Jendela *Properties*

d. *Form*

Form merupakan tempat untuk mendesain atau membangun sebuah program, didalamnya bisa ditanamkan sebuah koding agar *form* tersebut dapat menjalankan *program* yang telah didesain oleh *user*. Berikut gambar *form*, seperti gambar 2.10 dibawah ini.



Gambar 2.10 Tampilan *Form*

2.9 MySQL

MySQL merupakan *software* yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *open source. Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat MySQL), selain tentu saja bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi[10].

Sebagai software DBMS, MySQL memiliki sejumlah fitur seperti :

a. Multiplatform

MySQL tersedia pada beberapa *platform* yaitu Windows, Linux, Unix, dan lain-lain.

b. Andal, cepat, dan mudah digunakan

MySQL termasuk sebagai *database server* yang andal, dapat menangani *database* yang besar dengan kecepatan tinggi, mendukung banyak sekali fungsi untuk

mengakses *database*, dan sekaligus mudah digunakan. MySQL dapat menangani sebuah tabel yang berukuran dalam *terabyte* (1 *terabyte* = 1024 *gigabyte*).

c. Jaminan keamanan akses

MySQL mendukung pengamanan database dengan berbagai kriteria pengaksesan. Sebagai gambaran, dimungkinkan untuk mengatur user tertentu agar bias mengkases data yang bersifat rahasia, sedangkan user lain tidak boleh. MySQL juga mendukung konektivitas ke berbagai *software*, ODBC (*Open Database Connectivity*), JDBC (*Java Database Connectivity*), PHP dan lain sebagainya.

d. Dukungan SQL

MySQL mendukung perintah SQL (*Structured Query Language*). Sebagaimana diketahui, SQL merupakan standar dalam pengaksesan *database* relasional. Pengetahuan akan SQL akan memudahkan siapa pun untuk menggunakan MySQL.