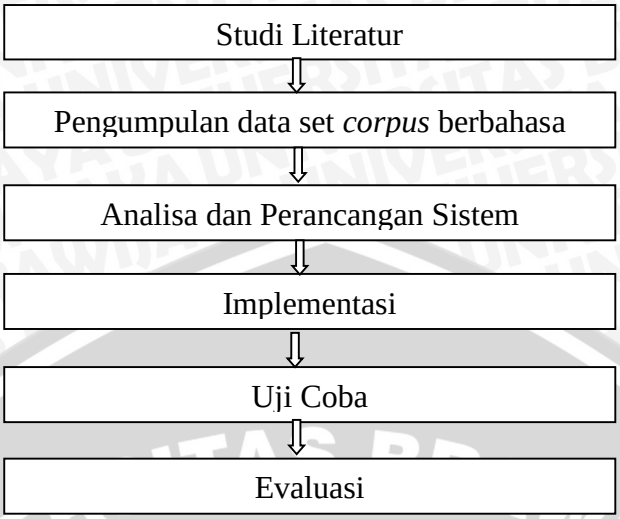
BAB III
METODOLOGI DAN PERANCANGAN

Bab ini berisi metodologi dan perancangan sistem klasifikasi berita berbahasa Inggris menngunakan algoritma *KNN* berbasis ontologi. Berikut ini adalah langkah-langkah yang dilakukan dalam penelitian ini :

1. Melakukan studi literatur yang mengenai klasifikasi teks khususnya untuk teks berbahasa Inggris dan ontologi serta metode klasfikasi teks yaitu algoritma *K-Nearest Neighbor*.
2. Mengumpulkan dataset berupa *corpus* berita berbahasa Inggris yang disimpan dalam format **.txt* yang akan digunakan sebagai dokumen uji dan dokumen latih.
3. Melakukan analisa dan perancangan sistem klasifikasi berita berbahasa Inggris dengan menggunakan algoritma *KNN* berbasis ontologi.
4. Melakukan implementasi sistem berdasarkan analisis dan perancangan yang telah dilakukan sebelumnya .
5. Melakukan uji coba sistem yang telah dibuat menggunakan dokumen uji.
6. Melakukan evaluasi dan menganalisa hasil klasifikasi berita yang dilakukan sistem berdasarkan hasil *precision*, *recall* dan *F1 measure*.

Langkah-langkah penelitian tersebut dapat dilihat pada gambar 3.1.



Gambar 3.1 Langkah-langkah Penelitian

Sumber : [Metodologi dan Perancangan]

3.1 Deskripsi Data

Data yang digunakan dalam penelitian ini adalah berita berbahasa Inggris yang dikeluarkan oleh Reuters, sebuah kantor berita di Inggris. Pada tahun 2000 Reuters merilis *corpus* yang berisi berita-berita yang telah dihimpun oleh Reuters untuk kepentingan pengembangan *Natural Language Processing, Information Retrieval*. Pada penelitian ini, *corpus* yang digunakan adalah Reuters-21578 90 kategori. Reuters-21578 memiliki dua jenis yaitu yang berisi 90 kategori dan 115 kategori.

Reuters-21578 terdiri dari 12.902 dokumen yang terbagi ke dalam 90 kelas. Setiap kategori memiliki sedikitnya 1 dokumen uji dan 1 dokumen latih. Pada penelitian ini kategori yang digunakan berjumlah empat yaitu *interest, money-fx, trade, crude.* Jumlah dokumen yang digunakan sebagai data latih berjumlah 400 dokumen dengan jumlah dokumen masing-masing kelas berjumlah 100 dokumen. Data uji berjumlah 20 dokumen dimana seluruh data uji bukan termasuk dari 400 dokumen latih. Keterangan jumlah kategori dapat dilihat pada tabel 3.1.

Tabel 3.1 Kategori dan jumlah dokumen pada *corpus*

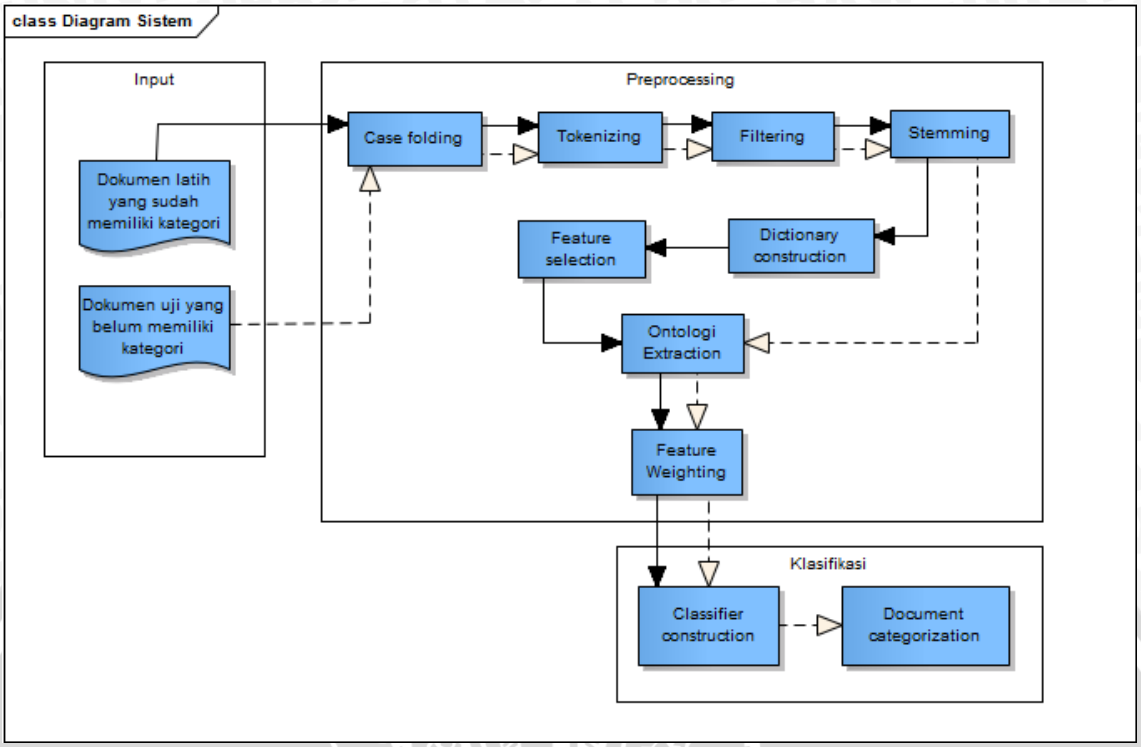
Kategori	**Jumlah Dokumen**
Interest	100
Money-fx	100
Trade	100
Crude	100
Total	400

3.2. Perancangan Sistem

Perancangan sistem ini merupakan deskripsi sistem yang digambarkan dengan diagram blok dan diagram UML (*Unified Model Language)*. Diagram blok menggambarkan proses yang akan dilalui oleh input sistem yaitu dokumen latih dan dokumen uji. Dokumen uji dan dokumen latih akan melalui proses *preprocessing* dan klasifikasi. Sistem yang akan dirancang akan menggunakan pendekatan berbasis objek. Diagram UML digunakan untuk menggambarkan hubungan antar objek di dalam sistem.

3.2.1. Diagram blok sistem

Sistem yang akan dibuat merupakan sistem untuk melakukan klasifikasi berita berbahasa Inggris. Sistem ini mengimplementasikan algoritma *K-Nearest Neighbor* (KNN) untuk melakukan klasifikasi pada berita berbahasa Inggris. Sistem terbagi menjadi dua proses utama secara garis besar yaitu *preprocessing data* dan klasifikasi. Arsitektur sistem dapat ditunjukkan pada gambar 3.2.



Gambar 3.2 Diagram Blok Sistem

Sumber : [Metodologi dan Perancangan]

Penjelasan mengenai diagram blok sistem pada Gambar 3.2. adalah sebagai berikut:

Input sistem terdiri dari dua jenis dokumen yaitu dokumen latih yang sudah memiliki kategori serta dokumen uji yang belum memiliki kategori. Penggunaan dua tanda panah yang berbeda pada gambar arsitektur sistem menunjukkan bahwa dokumen uji dan dokumen latih mendapatkan perlakuan berbeda pada saat *preprocessing*.

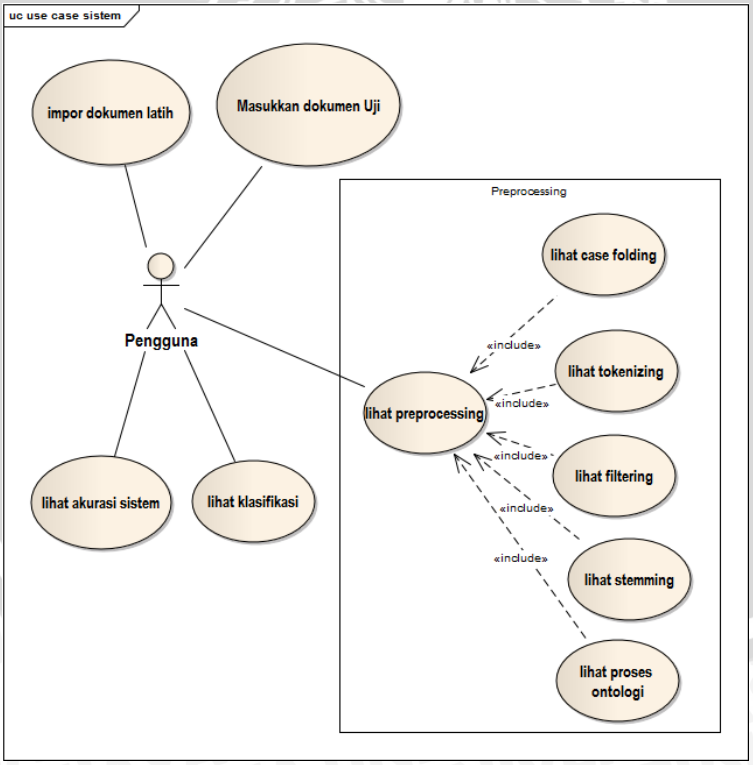
Kedua jenis dokumen tersebut melalui proses *preprocessing* yaitu *case folding*, *tokenizing*, *filtering*, *stemming*, *ontology extraction*, *dictionary construction*, *feature selection*, dan *feature weighting*.

Proses ekstraksi ontologi dilakukan setelah proses *stemming* pada dokumen latih dan uji. *Term-term* dari dokumen uji dicocokkan sinonim nya dengan kata pada database WordNet kemudian *term-term* yang saling bersinonim akan digabungkan frekuensi kemunculannya.

Pada tahap *feature wighting*, *term-term* dari dokumen uji dan dokumen latih dihitung bobotnya. Bobot dari dokumen latih dan dokumen uji ini digunakan untuk membangun *classifier* pada proses klasifikasi. *Classifier* dibentuk menggunakan metode KNN. Proses menghitung kesamaan antar dokumen menggunakan *cosine similarity*. Hasil dari proses *classifier* adalah kelas atau kategori dari dokumen uji.

3.2.2. Diagram *Unified Model Language* (UML)

Sistem akan dibangun menggunakan konsep *Object Oriented Programming* (OOP). UML digunakan untuk memodelkan sistem yang dibangun menggunakan konsep OOP. Diagram UML yang akan digunakan adalah diagram *use case* dan *class diagram*. *Use case* adalah deskripsi kejadian yang dilakukan oleh sistem dan dapat dilihat oleh pengguna. Diagram *use case* dari sistem klasifikasi berita berbahasa Inggris menggunakan algoritma KNN berbasis ontologi dapat dilihat pada gambar 3.3.



Gambar 3.3 Diagram *Use Case* Sistem

Sumber : [Metodologi dan Perancangan]

Pada Gambar 3.3 aktor pada sistem yaitu *user* dapat menggunakan fungsi-fungsi yang terdapat dalam sistem. Skenario-skenario tersebut akan dijelaskan pada tabel 3.2.

Tabel 3.2 Skenario *Use Case*

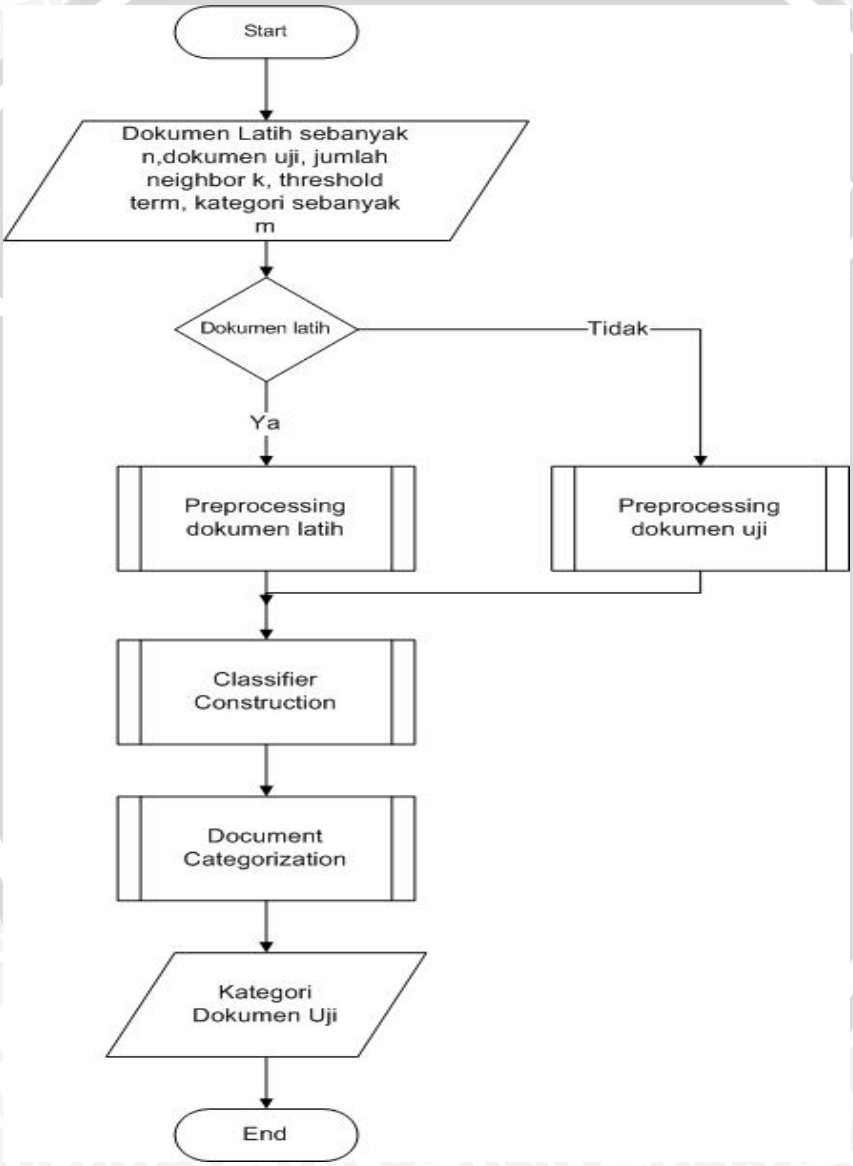
Impor Dokumen Latih	
Deskripsi Singkat	Pengguna mengimporkan dokumen latih dari suatu direktori
Aktor	Pengguna
Pre-Condition	Sistem menampilkan tombol impor dokumen uji
Post-Condition	Sistem menampilkan daftar dokumen latih beserta isi dari setiap dokumen latih
Masukkan Dokumen Uji	
Deskripsi Singkat	Penggunan memasukkan dokumen uji yang belum diketahui kategorinya
Aktor	Pengguna
Pre-Condition	Sistem menampilkan kotak dialog direktori dari dokumen uji
Post-Condition	Sistem menampilkan dokumen latih beserta isinya.
Lihat *Preprocessing*	
Deskripsi Singkat	Menampilkan hasil dari *preprocessing* dokumen latih dan dokumen uji
Aktor	Pengguna
Relationship	Lihat *case folding*, lihat *Tokenizing*, lihat *filtering*, lihat *stemming*, lihat *proses ontologi*
Pre-Condition	Sistem menampilkan form hasil *preprocessing*
Post-Condition	Sistem menampilkan nilai TFIDF dari setiap *term*
Lihat *Case Folding*	
Deskripsi Singkat	Merubah string berita menjadi karakter huruf kecil
Pre-Condition	Sistem menampilkan form hasil *case folding*
Post-Condition	Sistem menampilkan string berita hasil dari proses *case folding*
Lihat *Tokenizing*	
Deskripsi Singkat	Memecah string berita menjadi kumpulan token

Pre-Condition	Sistem melakukan *tokenizing* setelah proses *case folding*
Post-Condition	Sistem menampilkan kumpulan token *term* berita
Lihat *Filtering*	
Deskripsi Singkat	Proses pembuangan *term* yang termasuk *stopwords lists*
Pre-Condition	Sistem telah melakukan *tokenizing*
Post-Condition	Sistem menampilkan *term-term* yang tidak termasuk *stopwords*
Lihat *Stemming*	
Deskripsi Singkat	Mengembalikan *term* ke dalam bentuk kata dasar
Pre-Condition	Sistem telah melakukan proses *tokenizing*
Post-Condition	Sistem menampilkan kata dasar dari setiap *term*
Lihat proses ontologi	
Deskripsi Singkat	Mencocokan sinonim setiap *term* dengan database WordNet
Pre-Condition	Sistem sudah terhubung dengan database WordNet
Post-Condition	Sistem menampilkan hasil penggabungan *term* yang bersinonim
Lihat klasifikasi	
Deskripsi Singkat	Melakukan klasifikasi dokumen uji ke dalam kategori
Aktor	Pengguna
Pre-Condition	Sistem sudah mendapatkan nilai TFIDF dari setiap *term*
Post-Condition	Sistem menampilkan hasil klasifikasi dokumen uji
Lihat akurasi sistem	
Deskripsi Singkat	Melakukan perhitungan akurasi sistem secara keseluruhan
Aktor	Pengguna
Pre-Condition	Sistem sudah melakukan klasifikasi dokumen sesuai skenario pengujian
Post-Condition	Sistem menampilkan hasil akurasi sistem secara keseluruhan

3.3 Perancangan Algoritma

Perancangan algoritma berisi algoritma-algoritma yang digunakan untuk sistem klasifikasi berita berbahasa Inggris menggunakan algoritma *K-Nearest Neighbor* berbasis ontologi. *Input* dari sistem adalah dokumen yang terdiri dari

dokumen latih dan dokumen uji. Dokumen-dokumen tersebut disimpan dalam *file* berformata **.txt*. Dokumen latih dan dokumen uji pertamama melalui proses *preprocessing* tetapi dokumen latih dan dokumen uji akan melalui proses *preprocessing* yang berbeda. Hasil dari *preprocessing* dari dokumen latih dan dokumen uji akan digunakan untuk membentuk *classifier*. Hasil dari pembentukan *classifier* akan digunakan untuk menentukan kategori dari dokumen uji. *Flowchart* sistem secara umum dapat dilihat pada gambar 3.4.



Gambar 3.4. *Flowchart* Sistem Secara Umum

Sumber : [Metodologi dan Perancangan]

3.3.1. *Preprocessing* Dokumen Latih

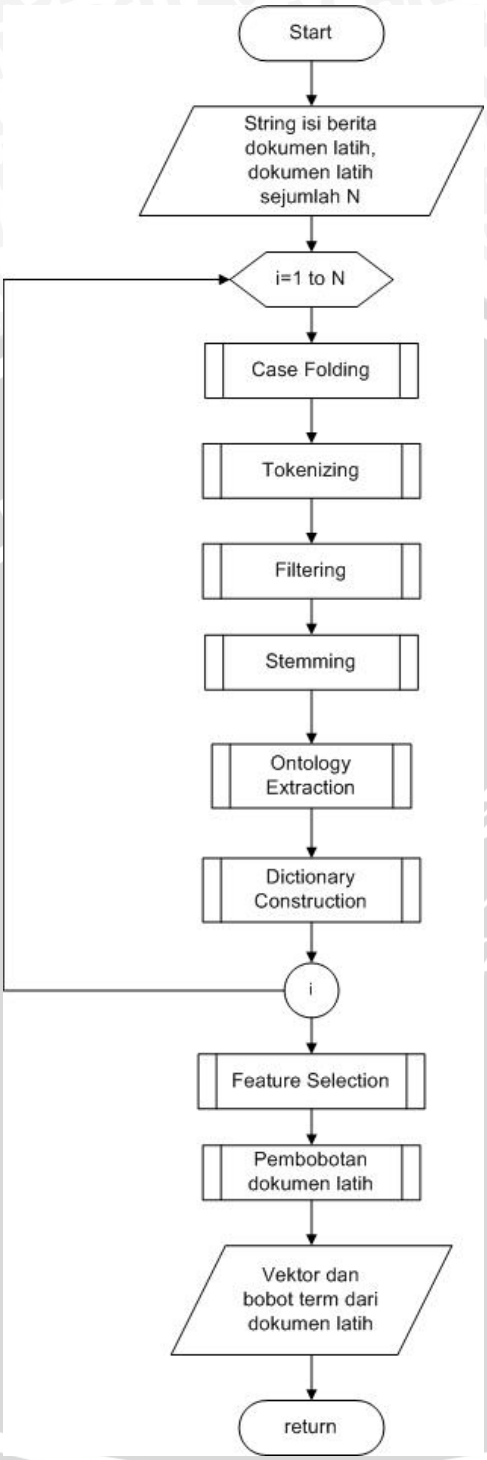
Preprocessing dokumen latih adalah proses perubahan dokumen latih menjadi vektor numerik yang akan digunakan untuk proses klasifikasi dokumen. Proses *preprocessing* dokumen latih terdiri dari beberapa sub proses yaitu *case folding, tokenizing, filtering, stemming, ontology extraction, feature selection,* dan pembobotan dokumen latih. *Flowchart* dari *preprocessing* dokumen latih dapat dilihat pada gambar 3.5.

1. ***Case folding***

Proses ini merupakan pembacaan setiap dokumen yaitu isi berita. Langkah-langkah yang dilakukan adalah mengubah karakter huruf yang berada di dalam berita menjadi huruf kecil semua. Selain itu, apabila terdapat karakter lain selain karakter huruf a sampai z, maka karakter tersebut akan dihilangkan dan diganti menjadi spasi. *flowchart* dari proses *case folding* dapat dilihat pada gambar 3.6.

Gambar 3.6. *Flowchart* Proses *case folding*

Sumber : [Metodologi dan Perancangan

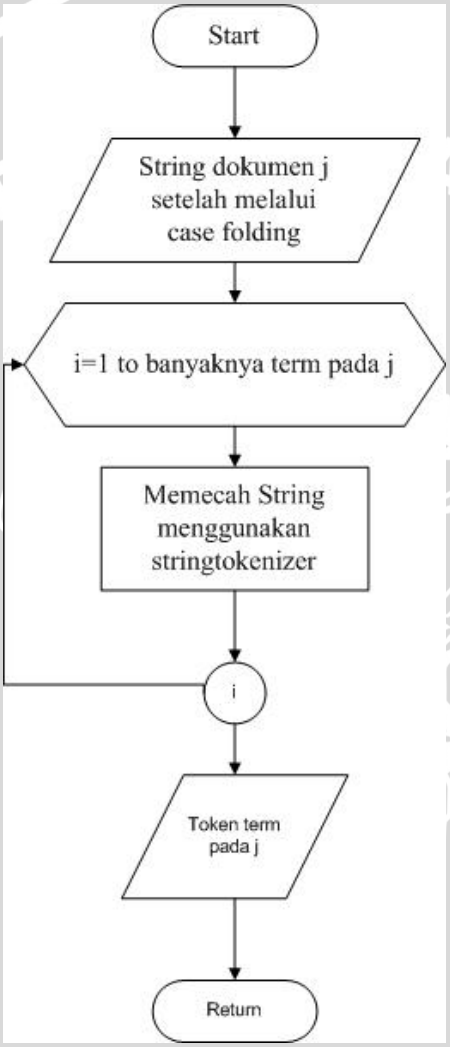


Gambar 3.5 *Flowchart Preprocessing* Dokumen Latih

Sumber : [Metodologi dan Perancangan]

2. *Tokenizing*

Setelah melalui proses *case folding*, proses selanjutnya adalah memecah *string*isi berita menjadi kata per kata. Pada bahasa pemrograman *Java* pemecahan stirng menjadi kata bisa dilakukkan dengan menggunakan fungsi *stringtokenizer* dari kelas *String*. *Flowchart* dari proses *tokenizing* dapat dilihat pada gambar 3.7.

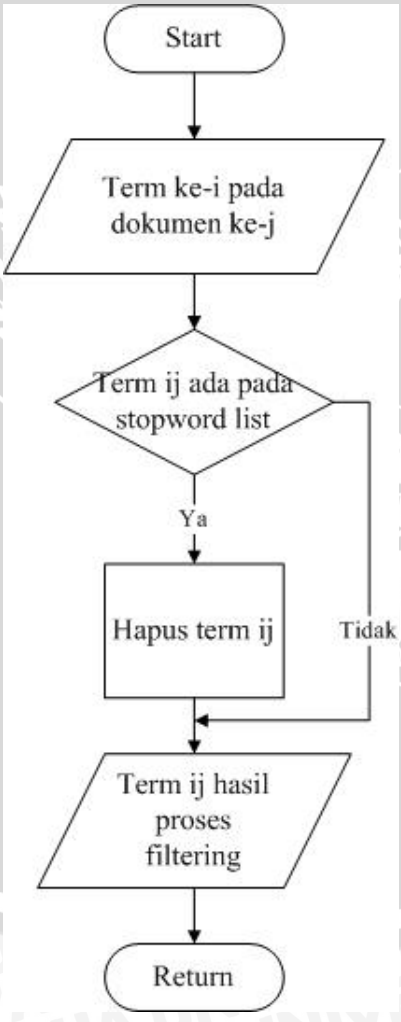


Gambar 3.7 *Flowchart* Proses *tokenizing*

Sumber : [Metodologi dan Perancangan]

3. *Filtering*

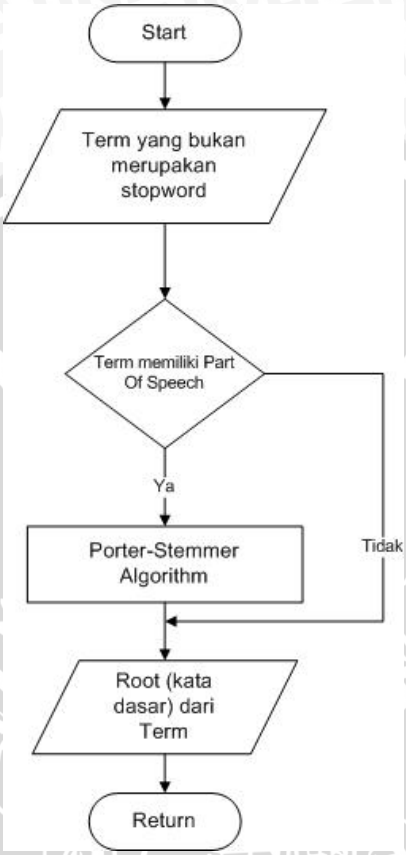
Filtering merupakan proses penghilangan kata-kata yang tidak penting dari dalam dokumen. Metode *filtering* yang digunakan adalah *stopword removal* yaitu membuang kata-kata yang dianggap tidak penting. Pada metode *filtering* ini digunakan *stopwords list* yaitu daftar kata-kata yang dianggap tidak bisa mendeskripsikan isi dokumen. Setiap *term* yang ada pada dokumen dicek apakah *term* tersebut ada terdapat pada *stopwords list* atau tidak. Apabila *term* termasuk ke dalam kata dalam *stopwords list*, maka *term* tersebut akan dibuang. *Flowchart filtering* dapat dilihat pada gambar 3.8.



Gambar 3.8. *Flowchart* Proses *filtering*

Sumber : [Metodologi dan Perancangan]

4. *Stemming*



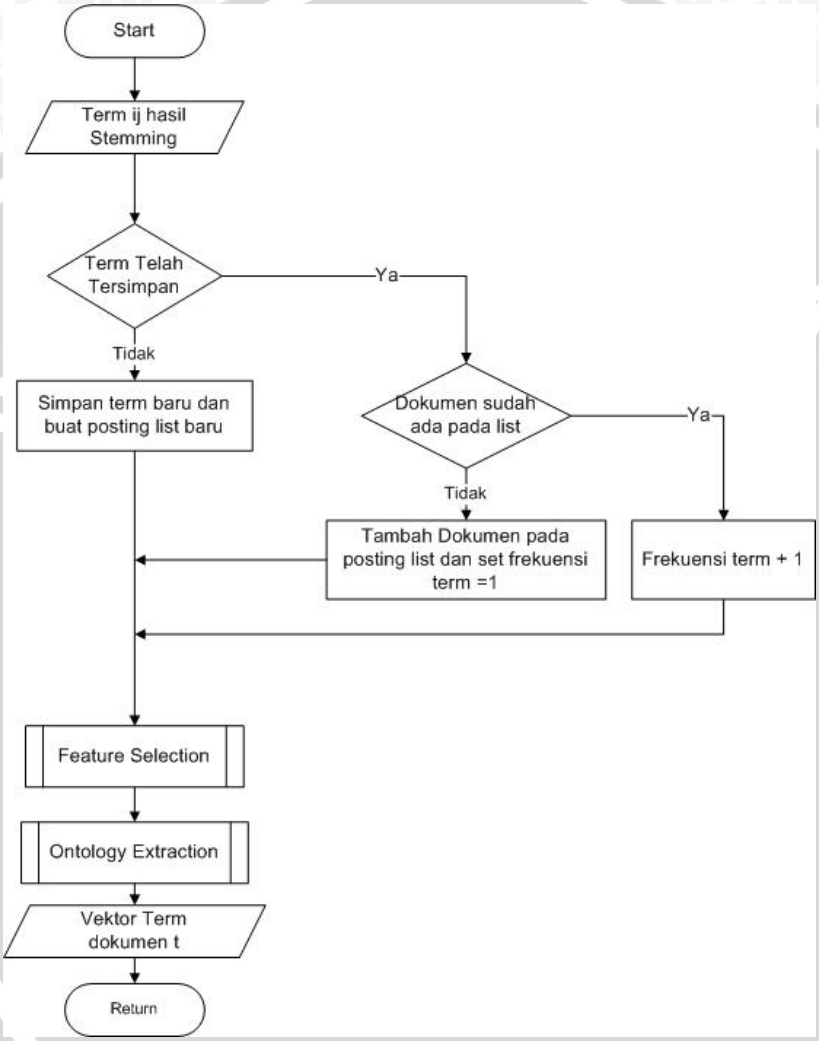
Gambar 3.9 *Flowchart* Proses *Stemming*

Sumber : [Metodologi dan Perancangan]

Term yang bukan merupakan kata dalam *stopwords list* kemudian akan melalui proses selanjutnya yaitu proses *stemming*. *Stemming* adalah proses pengembalian *term* menjadi bentuk kata dasar nya. Untuk bahasa Inggris, algoritma *Stemming* yang digunakan adalah algoritma *Porter-stemmer*. *Term-term* yang melalui proses *stemming* ini adalah *term* yang memiliki *Part of Speech* (POS). POS ini diantaranya adalah *noun, adjective, verb,* dan *adverb*. Untuk mengetahui POS dari suatu *term* dapat diketahui dari *WordNet* apakah *term* tersebut memiliki POS atau tidak. Jika *term* tidak memiliki POS, maka *term* tidak akan melalui proses *stemming*. Algoritma ini melakukan reduksi kata hingga menjadi bentuk kata dasar nya. Algoritma ini memiliki aturan-aturan untuk mereduksi akhiran seperti yang

telah digambarkan pada tabel 2.1 sampai dengan tabel 2.9. Algoritma *Porter-stemmer* terdiri dari 5 langkah pereduksian kata seperti yang telah dipaparkan pada bab sebelumnya. *Flowchart* dari algoritma *Porter-Stemmer* dapat dilihat pada gambar 3.9.

5. Dictionary Construction



Gambar 3.10. *Flowchart* Proses *Dictionary Construction*

Sumber : [Metodologi dan Perancangan]

Term-term hasil dari proses *stemming* akan disimpan pada *Inverted Index.* Proses penyimpanan *term* ini adalah proses *Dictionary Construction* (pembentukan kamus). Implementasi dari *Inverted Index* adalah menggunakan struktur data *hashmap. Hashmap* terdiri *key* (kunci) dan *value* (nilai). *Hashmap inverted index*

terdiri dari *term-term* yang merupakan *key* serta *posting list* merupakan *value*. *Posting list* berisi frekuensi tiap *term* pada setiap dokumen. Apabila *term* belum ada pada *hashmap* maka *term* akan disimpan dan membuat *posting list* baru. Apabila *term* sudah ada tetapi nama dokumen belum ada, maka akan dibuat *posting list* baru untuk dokumen tersebut beserta nilai frekuensi nya. *Flowchart* dari proses *dictionary construction* dapat dilihat pada gambar 3.10.

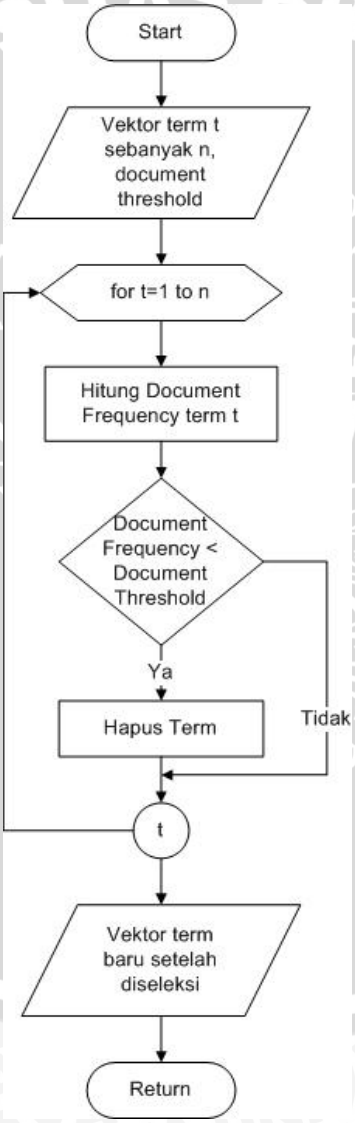
6. Feature Selection

Term-term pada dokumen latih telah disimpan ketika proses *dictionary construction.* Setelah *term* disimpan dalam *inverted index* maka proses selanjutnya adalah *Feature Selection* untuk mengurangi dimensi dari vektor dokumen agar dapat meningkatkan performa klasifikasi. Metode *feature selection* yang digunakan adalah *Document Frequency*. *Document Frequency* menghitung pada berapa dokumen *term* tersebut muncul. Nilai dari *Document Frequency* ditentukan dari nilai *threshold* yang dimasukkan ke dalam sistem. Nilai *threshold* merupakan bilangan bulat lebih dari 0. *Term* yang memiliki nilai *Document Frequency* dibawah dari nilai *threshold* yang telah ditentukan maka *term* tersebut akan dihapus. Jika nilai *threshold* sama dengan 1 maka tidak terjadi pembuangan *term* karena *term* yang disimpan pada proses *Dictionary Construction* minimal memiliki *Document Frequency* sama dengan 1. *Flowchart* dari *feature selection* dapat dilihat pada gambar 3.11.

7. Ontology Extraction

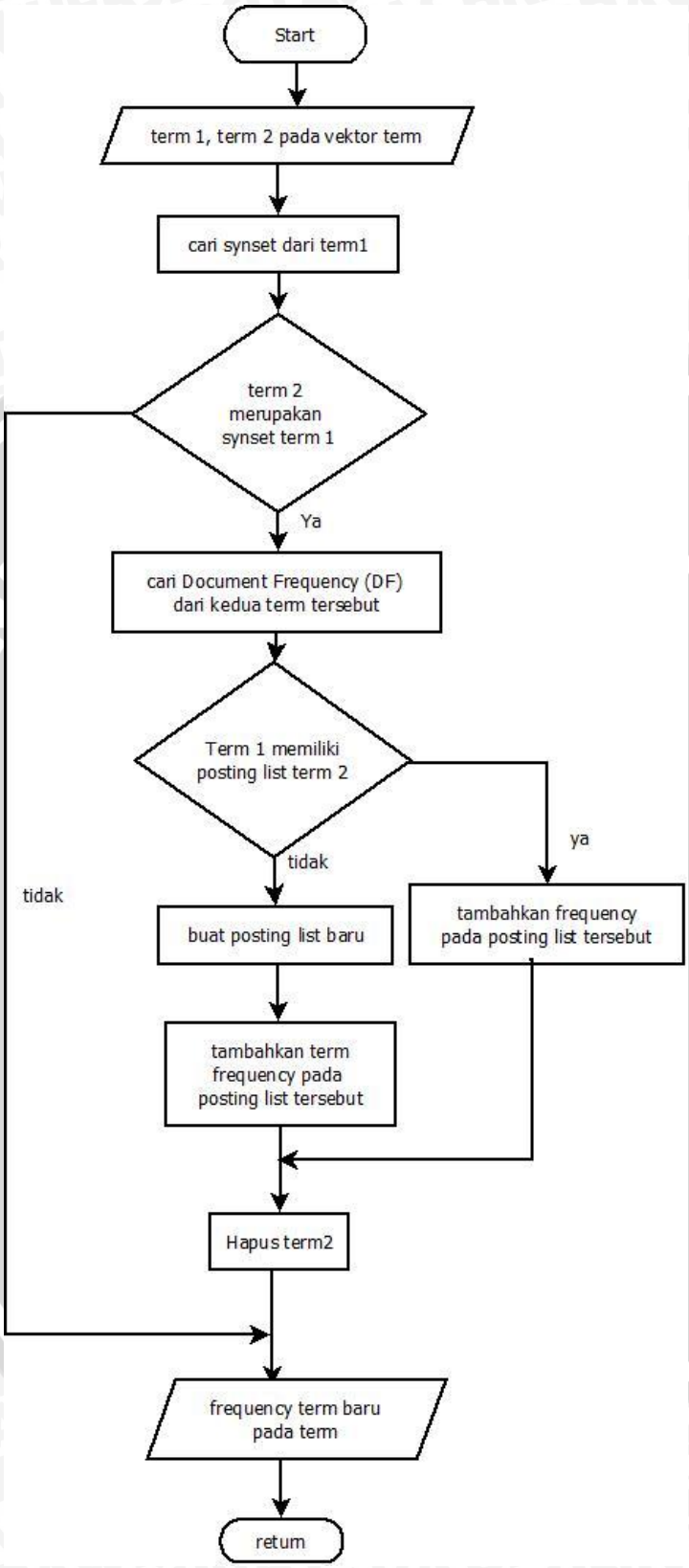
Setelah *term* melalui proses *feature selection,* langkah selanjutnya adalah mencari *term-term* yang saling bersinonim pada kamus dengan menggunakan bantuan *wordnet*. Setelah *term* yang saling bersinonim pada kamus ditemukan, sebelum *term* tersebut digabungkan, *term-term* tersebut dicari terlebih dahulu *Document Frequency* (DF) dari masing-masing *term*. *Term* yang memiliki DF lebih kecil akan digabungkan dengan *term* yang memiliki DF lebih besar. Jika DF *term* tersebut bernilai sama, maka *term* yang akan digabungkan adalah salah satu saja. *Term* yang sudah digabungkan akan dihapus dari kamus. Aturan penggabungan dari

term tersebut adalah jika *posting list* dari term yang dimiliki DF lebih kecil sudah ada pada *posting list* term yang memiliki DF lebih besar, maka frekuensi *term* dari dokumen yang memiliki DF lebih kecil akan ditambahkan ke *posting list* dokumen yang memiliki DF lebih besar. Jika *posting list* pada dokumen yang memiliki DF lebih besar belum ada, maka akan dibuat posting list baru sesuai dengan *posting list* yang dimiliki oleh term yang akan digabungkan. *Flowchart* dari proses *ontology extraction* dapat dilihat pada gambar 3.12.



Gambar 3.11. *Flowchart* Proses *Feature Selection*

Sumber : [Metodologi dan Perancangan]

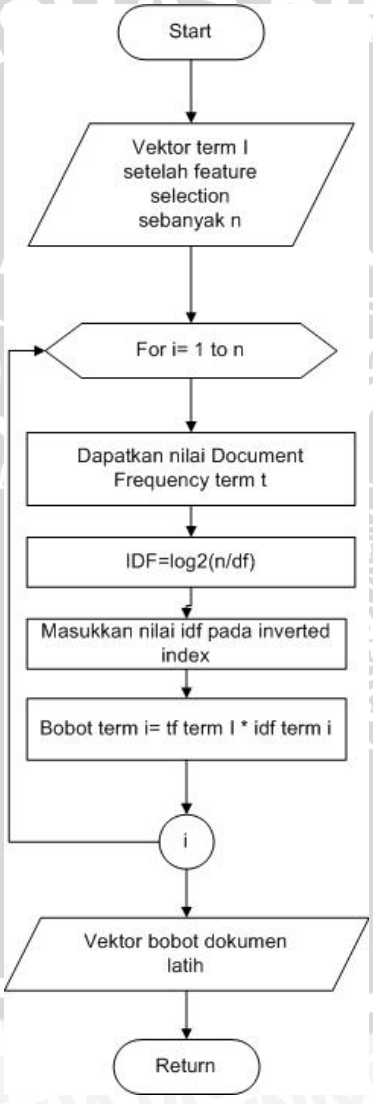


Gambar 3.12. *Flowchart* Proses *Ontology Extraction*

Sumber : [Metodologi dan Perancangan]

8. Pembobotan Dokumen Latih

Term-term yang telah diseleksi akan dihitung bobotnya pada setiap dokumen. Metode pembobotan yang digunakan adalah metode TF-IDF. Rumus dari TF-IDF dapat dilihat pada persamaan 2-1. Nilai dari TF tiap *term* telah diketahui sebelumnya pada saat proses *Dictionary Construction*. Nilai TF dari setiap *term* disimpan pada *inverted index* lalu kemudian dikalikan dengan nilai IDF *term* tersebut. *Flowchart* dari pembobotan dokumen latih menggunakan metode TF-IDF dapat dilihat pada gambar 3.13.

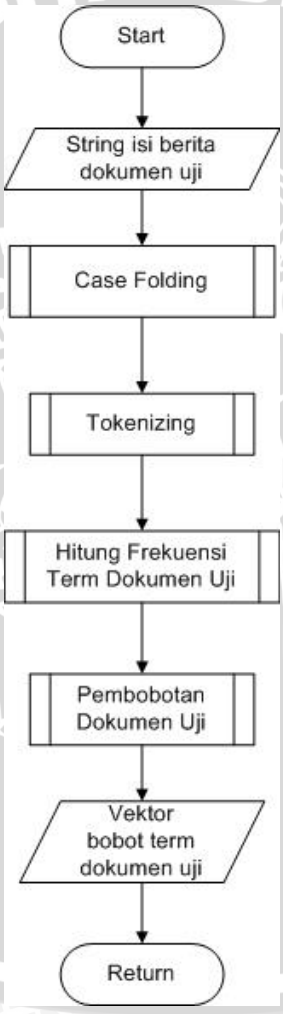


Gambar 3.13. *Flowchart* Proses Pembobotan Dokumen Latih

Sumber : [Metodologi dan Perancangan]

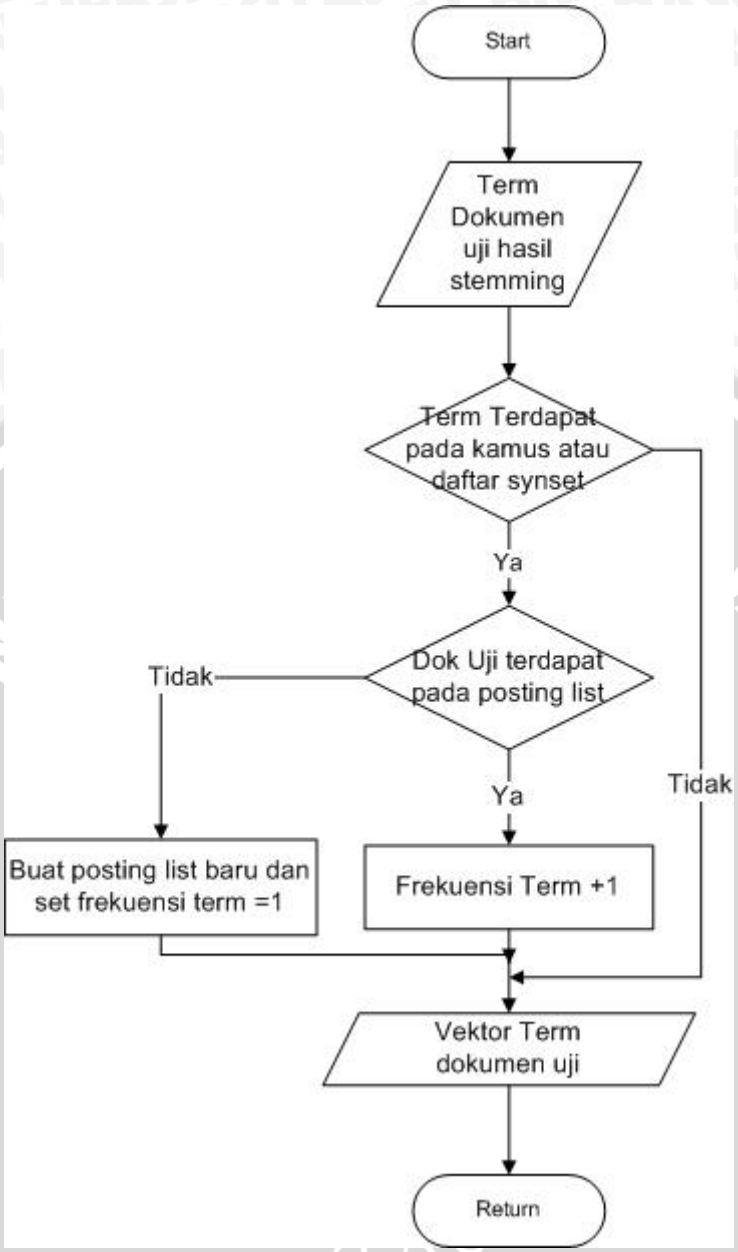
3.3.2. *Preprocessing* Dokumen Uji

Preprocessing juga dilakukan pada dokumen uji. Tujuan dari *preprocessing* dokumen uji adalah merubah dokumen uji menjadi representasi vektor. *Preprocessing* dokumen latih dengan dokumen uji memiliki perbedaan dimana pada *preprocessing* dokumen uji tidak terdapat proses *dictionary construction* dan *feature selection* karena *term* yang digunakan pada dokumen uji mengacu pada *term* yang ada pada kamus yang telah dibentuk pada saat *preprocessing* dokumen latih. Proses dari *preprocessing* dokumen uji dapat dilihat pada gambar 3.14.



Gambar 3.14. *Flowchart Preprocessing* Dokumen Uji

Sumber : [Metodologi dan Perancangan]

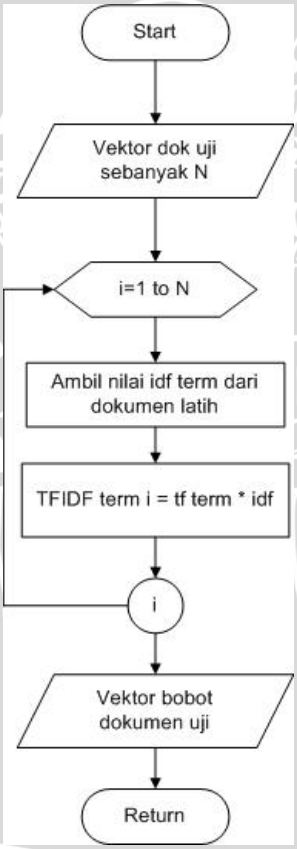


Gambar 3.15. *Flowchart* Proses Hitung Frekuensi *Term* uji

Sumber : [Metodologi dan Perancangan]

Proses *case folding*, *tokenizing*, *filtering* dan *stemming* pada *preprocessing* dokumen uji sama dengan yang dilakukan pada *preprocessing* dokumen latih. Setelah melalui proses *stemming*, *term* pada dokumen uji akan melalui proses *ontology extraction*. Proses *ontology extraction* dari *term* pada dokumen uji ini adalah mencocokkan apakah *synset* dari *term* tersebut ada pada daftar *synset* pada

dokumen latih. Jika *term* pada dokumen uji tidak terdapat pada kamus dokumen latih tetapi *term* tersebut ada didalam daftar *synset* maka *term* tersebut dianggap sama dengan *term* pada dokumen uji yang bersinonim dengan *synset* tersebut. Setelah melalui proses *ontology extraction*, *term* yang ada pada dokumen uji dicocokkan dengan *inverted index* atau kamus yang telah dibuat pada proses *preprocessing* dokumen latih. *Term* yang tidak ada pada *inverted index* dan juga tidak terdapat pada daftar *synset* tidak akan dihitung frekuensi nya atau diberikan nilai 0 dan tidak akan dilakukan proses lebih lanjut. *Flowchart* perhitungan frekuensi *term* dokumen uji dapat dilihat pada gambar 3.15.



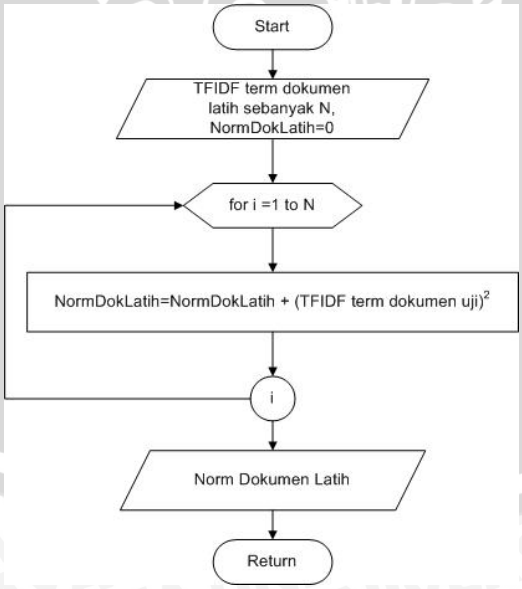
Gambar 3.16. *Flowchart* Pembobotan Dokumen Uji

Sumber : [Metodologi dan Perancangan]

Setelah frekuensi dari *term* dokumen uji dihitung, langkah selanjutnya adalah menghitung bobot dokumen uji dengan menggunakan metode TFIDF. Nilai TF yang digunakan adalah nilai TF *term* pada dokumen uji sedangkan nilai IDF dari *term* adalah nilai IDF yang sudah dihitung pada *preprocessing* dokumen latih. *Flowchart* pembobotan dokumen latih dapat dilihat pada gambar 3.16.

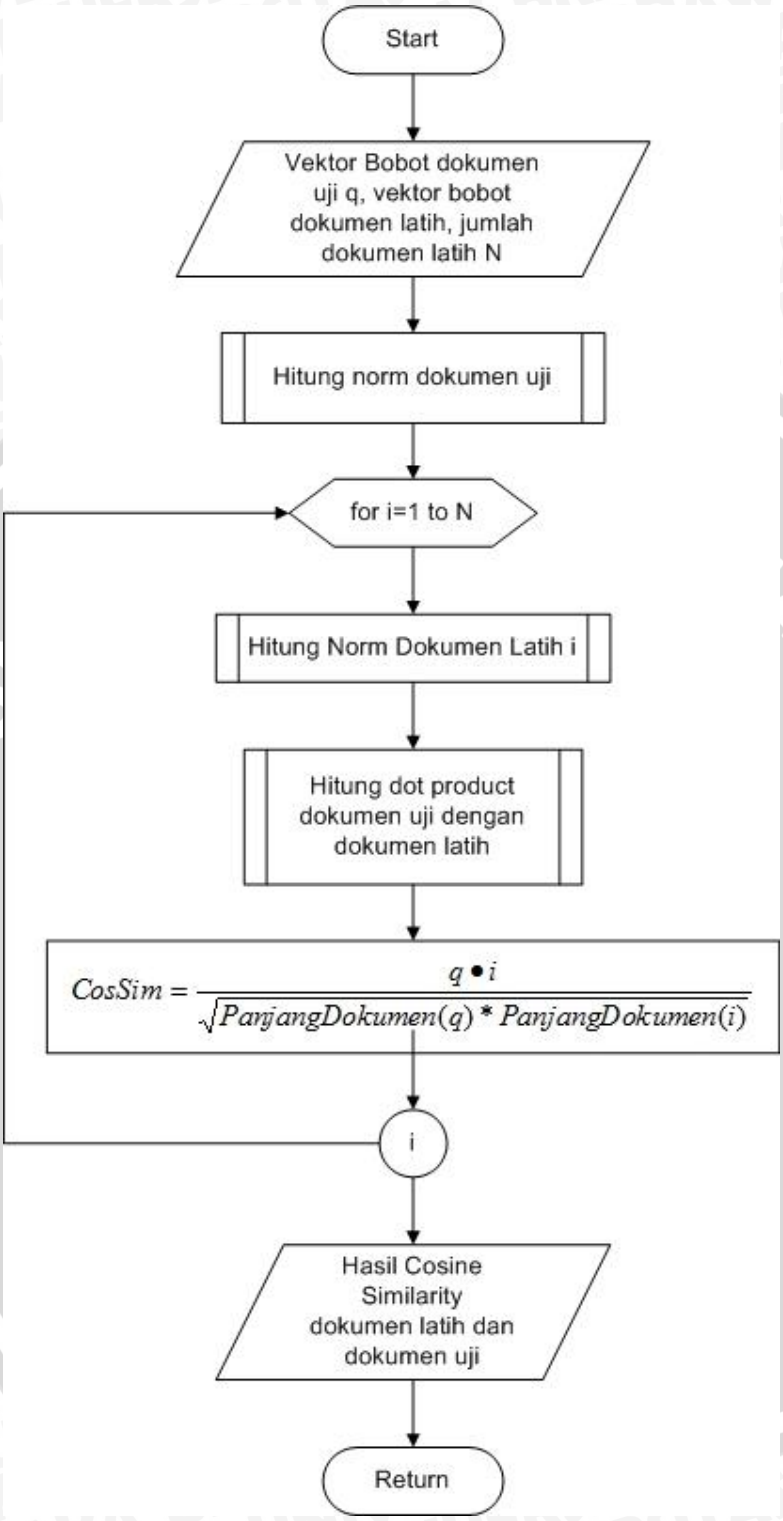
3.3.3. *Classifier Construction*

Proses *classifier construction* atau pembentukan *classifier* adalah proses pengklasifikasian dokumen uji ke dalam kategori yang ada pada dokumen latih. Proses ini terdiri dari dua proses utama yaitu *cosine similarity* yaitu perhitungan kemiripan dokumen uji dengan setiap dokumen latih. *Flowchart* dari *cosine similarity* dapat dilihat pada gambar 3.17. Pada gambar 3.18 menggambarkan proses perhitungan panjang dokumen uji sementara gambar 3.19 menggambarkan proses perhitungan panjang dokumen latih. Setelah dihitung panjang dari kedua dokumen maka langkah selanjutnya adalah menghitung *dot product* dari dokumen latih dengan dokumen uji. Proses penghitungan *dot product* dari dokumen latih dan dokumen uji dapat dilihat pada gambar 3.20.



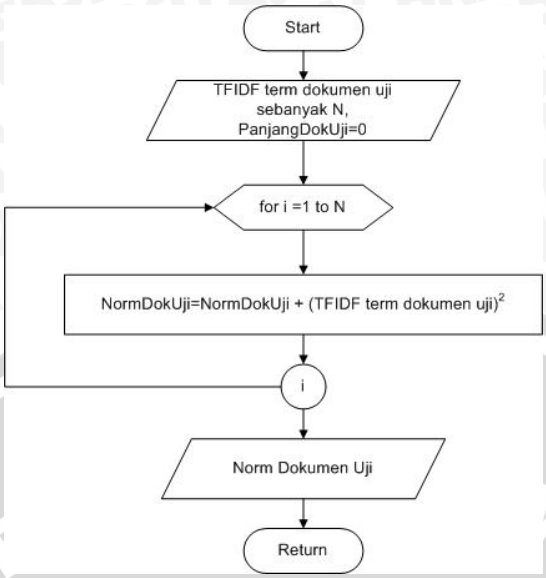
Gambar 3.19. *Flowchart* Proses *Norm* Dokumen Latih

Sumber : [Metodologi dan Perancangan]



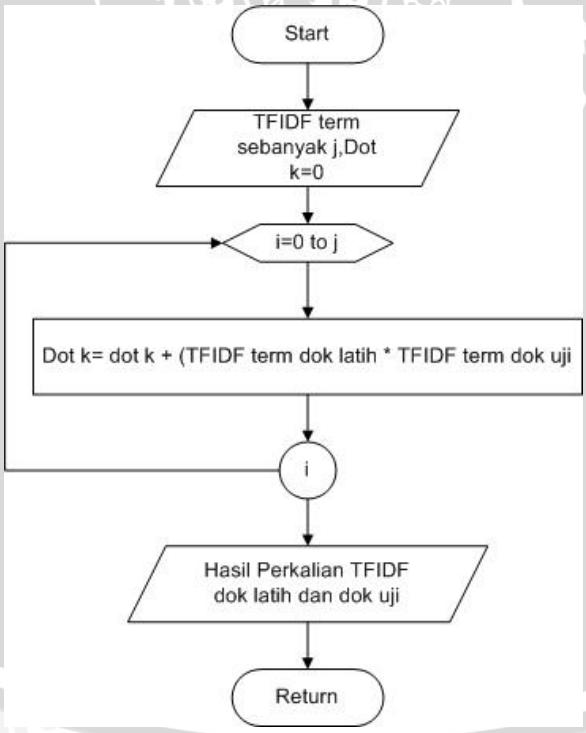
Gambar 3.17. *Flowchart* Proses *Cosine Similarity*

Sumber : [Metodologi dan Perancangan]



Gambar 3.18. *Flowchart* Proses *Norm Dokumen Uji*

Sumber : [Metodologi dan Perancangan]

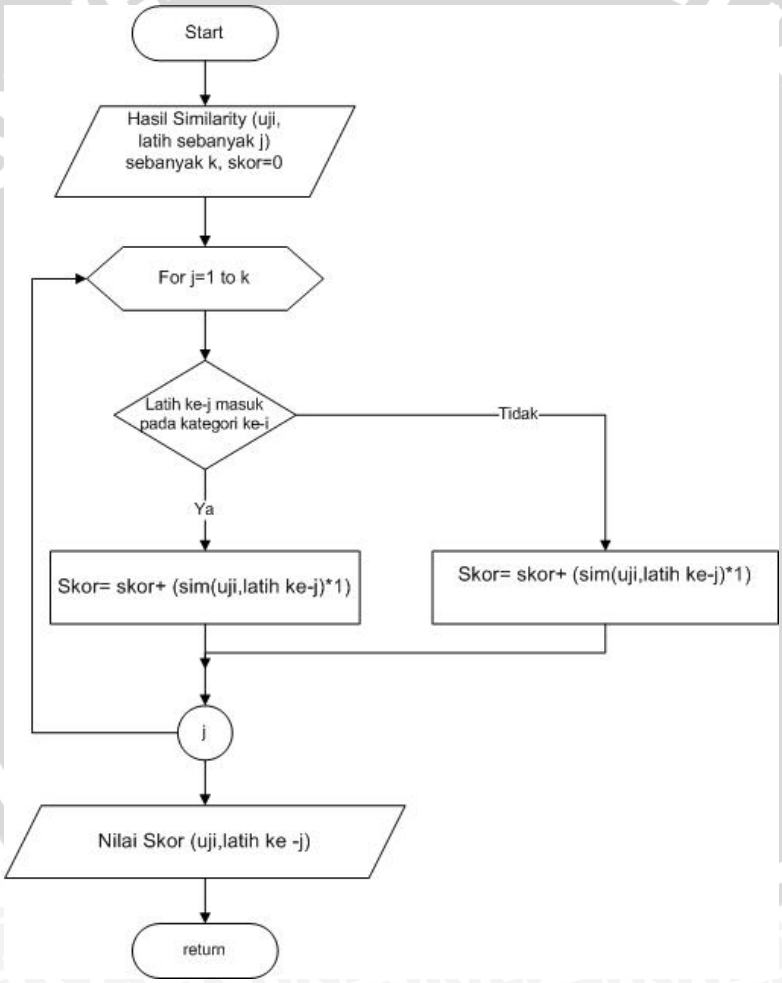


Gambar 3.20. *Flowchart* Hitung Dot Produk Dokumen Uji dan Dokumen Latih

Sumber : [Metodologi dan Perancangan]

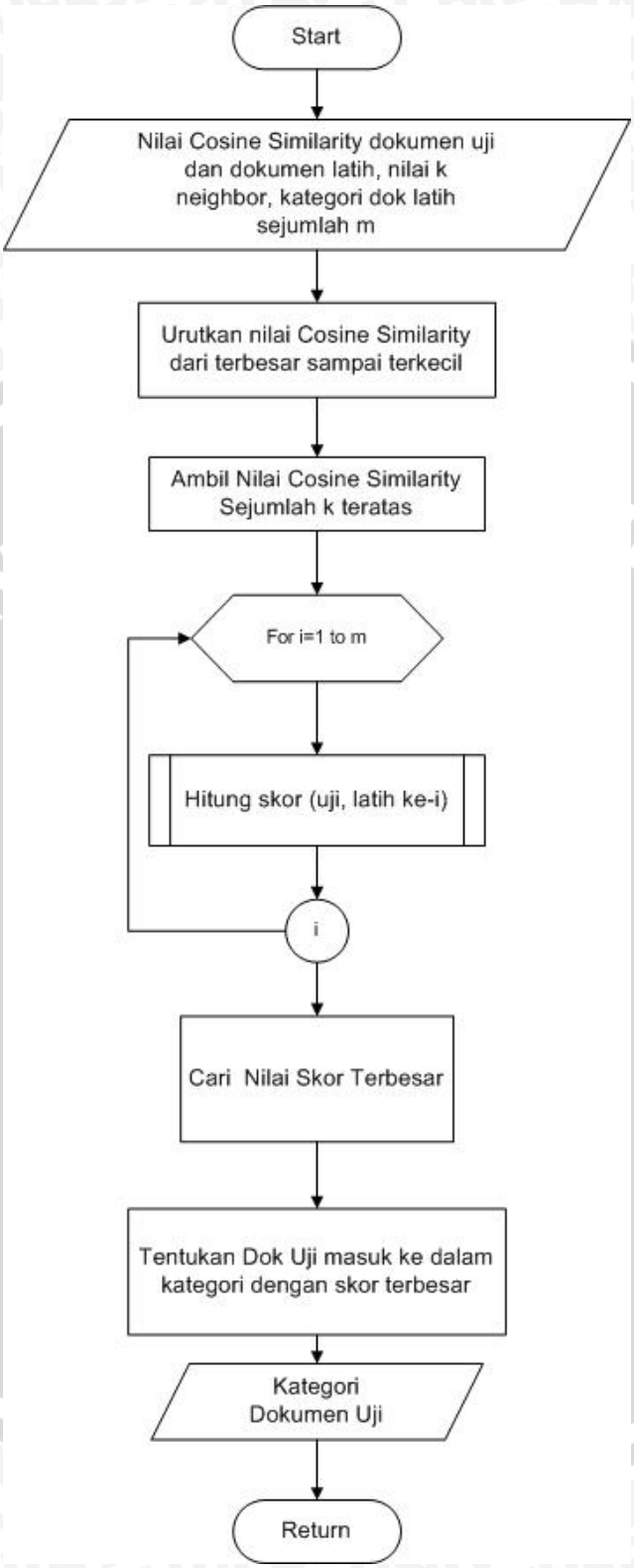
3.3.4. *Document Categorization*

Proses terakhir adalah proses *document categorization* atau klasifikasi dokumen. Proses ini menggunakan algoritma *K-Nearest Neighbor* (KNN). Algoritma KNN akan memilih sejumlah *k* dokumen latih beserta kategorinya yang dianggap berdekatan dengan dokumen uji. Setelah diambil sejumlah *k* dokumen latih maka dilakukan proses *scoring* untuk memilih kategori dari dokumen uji yang prosesnya dapat dilihat pada gambar 3.22. *Flowchart* dari proses *Document Categorization* dengan menggunakan algoritma KNN dapat dilihat pada gambar 3.21.



Gambar 3.22. *Flowchart Scoring* Kategori Dokumen Uji

Sumber : [Metodologi dan Perancangan]



Gambar 3.21. ***Flowchart Document Categorization***

Sumber : [Metodologi dan Perancangan]

3.4 Perhitungan Manual

Bagian perhitungan manual menjelaskan setiap langkah proses klasifikasi berita berbahasa Inggris menggunakan algoritma KNN berbasis ontologi. Data latih yang diambil sebanyak 12 dokumen yang terdiri dari 4 kelas. Untuk setiap dokumen latih lalu diambil beberapa kalimat inti dari dokumen untuk menyederhanakan penjabaran perhitungan manual. Persebaran dokumen pada tiap kategori dapat dilihat pada tabel 3.3. Isi dari setiap dokumen latih dan dokumen uji dapat dilihat pada tabel 3.4.

Tabel 3.3 Persebaran Dokumen Latih Perhitungan Manual

Kategori	Jumlah Dokumen
Interest	3
Money-fx	3
Trade	3
Crude	3

Tabel 3.4 Keterangan kelas dan isi dokumen latih

No	Kelas Dokumen	Isi Dokumen
1	*Crude*	Ghana will import 15.000 tones of crude oil annualy from Iran under an agreement reached in Teheran today.
2	*Crude*	Nerco inc said its oil and gas unit closed the acquisition of a 47 pct working interest in the Broussard oil and gas field.

3	*Crude*	Triton energy corp said proven reserves of the Villespedue oil field in France's Paris were estimated at a total of 67.5 mln barrels on march one.
4	*Interest*	Some also look for a permanent reserve injection to offset seasonal pressures via an outright purchase of bills or coupons this afternoon.
5	*Interest*	The Federal Reserve entered the U.S Government securuities market to arrange overnight System repurchase agreement.
6	*Interest*	The Bank of France said it left its intervention rate unchanged at 7-3/4 pct when it injected funds in the market against first category paper in today's money market intervention tender.
7	*Money-fx*	A meeting of finance minsters and central bankers of the group of five ended after nearly three and half hour.
8	*Money-fx*	The bank of japan bought a modest amount of dollars at around 145.10 yen just after the market here opened
9	*Money-fx*	The Swiss Federal Government will launch a new series of three month money market certificates.
10	*Trade*	Presidential sopkesman Marlin Fitzwater said U.S trade sanctions against Japan were likely take affect on April 17.
11	*Trade*	Japanese officials are to meet under the emergency provisions of a July 1986 semiconductor pact to discuss trade and the punitive tariffs.
12	*Trade*	Canada had a trade surplus of 1.25 billion dollars in February compared with an upward revised 623 million dollars surplus in January.

13	Tidak Diketahui	Diamond Shamrock Corp said that effective today it had cut its contract proces for crude oil by 1.50 dollars a barrel.

Langkah pertama adalah *preprocessing* dokumen latih. Setiap dokumen latih akan melalui proses *case folding, tokenizing, stopword filtering*, dan *stemming*. Contoh penerapan *case folding* misalnya diambil contoh dokumen nomor satu dari daftar dokumen latih diatas.

Dokumen sebelum proses case folding :

Ghana will import 15.000 tones of crude oil annualy from Iran under an agreement reached in Teheran today.

Dokumen setelah proses *case folding* akan menjadi :

ghana will import tonnes of crude oil annualy from iran under an agreement reached in teheran today

Setelah melalui proses *case folding*, langkah selanjutnya adalah proses pemecahan dokumen menjadi token *term-term*. Hasil dari proses *tokenizing* dapat dilihat pada tabel 3.5

Tabel 3.5 Hasil *Tokenizing*

ghana	crude	under	tehran
will	oil	an	today
import	annualy	agreement	
tonnes	from	reached	
of	iran	in	

Proses selanjutnya adalah *filtering* yaitu membuang *term* yang merupakan *stopword*. Hasil dari proses *filtering* dapat dilihat pada tabel 3.6

Tabel 3.6 Hasil *Filtering*

ghana	crude	agreement
will	oil	reached
import	annualy	tehran
tonnes	iran	today

Setelah melalui proses *filtering*, langkah selanjutnya adalah melakukan proses *stemming* untuk setiap *term*. Proses *stemming* adalah proses mengubah *term* menjadi bentuk kata dasarnya. Algoritma yang digunakan untuk proses *stemming* adalah algoritma *Porter Stemmer*. Algoritma ini merupakan algoritma penghilangan akhiran dari suatu term. Hasil dari proses *Stemming* dapat dilihat pada tabel 3.7

Tabel 3.7 Hasil *Stemming*

ghana→ ghana	crude→crude	agreement→agree
will→will	oil→oil	reached→reach
import→import	annualy→annual	tehran→tehran
tonnes→ton	iran→iran	today→today

Setelah melalui proses *stemming*, langkah selanjutnya adalah membentuk *inverted index*. Sebelum membentuk *inverted index*, sebelumnya akan menghitung *term frequency* (TF) dari setiap *term* pada setiap dokumen. Nilai dari *term frequency* dari dokumen diatas dapat dilihat pada tabel 3.8

Tabel 3.8 Hasil Perhitungan *Term Frequency*

ghana=1	crude=1	agree=1
will=1	oil=1	reach=1
import=1	annual=1	tehran=1
ton=1	iran=1	today=1

Inverted index yang terbentuk pada *preprocessing* dokumen latih berisi kumpulan *term* dari seluruh dokumen latih. Oleh karena itu proses *case folding*, *tokenizing*, *stemming* dan perhitungan *Term Frequency* juga dilakukan untuk semua dokumen latih. Setelah semua *term* dari dokumen latih dihitung *term frequency* nya, langkah selanjutnya adalah menentukan *document frequency* dari setiap *term*. *Document frequency* adalah jumlah dokumen dimana *term* tersebut muncul. *Inverted index* serta *document frequency* dari dokumen latih dapat dilihat pada tabel 3.9.

Tabel 3.9 *Inverted Index* dan *Document Frequency term-term* pada Dokumen Latih

Term	**D 1**	**D 2**	**D 3**	**D 4**	**D 5**	**D 6**	**D 7**	**D 8**	**D 9**	**D1 0**	**D1 1**	**D1 2**	**D F**
ghana	1												1
will	1								1				2
import	1												1
ton	1												1
crude	1												1
oil	1	4	1										3
annual	1												1
iran	1												1

agree	1				1							2
reach	1											1
tehran	1											1
today	1					1						2
nerce		1										1
gas		2										1
unit		1										1
close		1										1
acquisi		1										1
pct		1	2			1						3
work		1										1
interest		1										1
broussar d		1										1
field		1	1									2
davis		1										1
mln		1	2								1	3
dollar		1						1				2
cash		1										1
triton			1									1
energi			1									1

corp			1									1
prove			1									1
reserve			1	1	1							3
villesped u			1									1
france			1	1								2
paris			1									1
basin			1									1
estimate			1									1
total			1									1
barrel			2									1
march			1									1
perman				1								1
injec				1								1
offset				1								1
season				1								1
pressur				1								1
….												
….												
….												
….												

statist											1	1

Inverted index telah terbentuk dan *document frequency* dari setiap *term* juga telah dihitung. Langkah selanjutnya adalah melakukan proses *feature selection*. Proses *feature selection* ini adalah membuang *term* yang memiliki nilai *document frequency* dibawah batas frekuensi (*frequency threshold*) yang telah ditentukan. Jika nilai *frequency threshold* yang ditentukan adalah 1, maka tidak ada pembuangan *term* karena *term* pada *inverted index* sedikitnya bernilai 1. Pada penjelasan perhitungan manual ini *frequency threshold* yang digunakan adalah 1 jadi tidak ada *term* yang dibuang.

Langkah selanjutnya adalah *ontology extraction*. Proses ini adalah proses menggabungkan *term* yang saling bersinonim pada *inverted index.* Proses pencarian sinoinim ini menggunakan bantuan database leksikal *WordNet*.*Term* yang saling bersinonim akan dihapus salah satu nya lalu frekuensi dari kedua *term* tersebut digabungkan. Setelah dilakukkan pencocokan untuk setiap *term*, terdapat beberapa *term* yang saling bersinonim, diantaranya adalah sebagai berikut:

- *term crude* bersinonim dengan *term oil.*

- *term close* bersinonim dengan *term end.*

- *term bought* bersinonim dengan *term purchase.*

Tabel 3.10 *Inverted index term* sebelum *ontology extraction*

Term	**D 1**	**D 2**	**D 3**	**D 4**	**D 5**	**D 6**	**D 7**	**D 8**	**D 9**	**D1 0**	**D1 1**	**D1 2**	**D F**
crude	1												1
oil	1	4	1										3
close		1											1
end							1						1

bought							1					1
purchase				1								1

Tabel 3.11 *Inverted index term* sesudah *ontology extraction*

Term	**D 1**	**D 2**	**D 3**	**D 4**	**D 5**	**D 6**	**D 7**	**D 8**	**D 9**	**D1 0**	**D1 1**	**D1 2**	**D F**
oil	2	4	1										3
end		1					1						2
purchase				1			1						2

Setalah proses *ontology extraction,* maka *inverted index* yang baru telah terbentuk dengan pengurangan *term* pada proses *feature selection* dan *ontology construction*. Langkah selanjutnya adalah menentukan *Inverse Document Frequency* (IDF) dari setiap *term* pada *inverted index*. Rumus menghitung IDF dapat dilihat pada persamaan 2-2. Contoh perhitungan salah IDF salah satu *term* adalah sebagai berikut:

Menghitung *term ghana*. Rumus dari IDF adalah sebagai berikut

$$idf_i = log_2\left(\frac{N}{df_i}\right)$$

Dengan :

N = jumlah keseluruhan dokumen yaitu 12 dokumen.

df_i = Nilai *document frequency* dari *term* tersebut.

Term ghana memiliki *document frequency* 1, maka nilai IDF dari *term ghana* sesuai rumus adalah sebagai berikut

$$log_2{}^{\frac{12}{1}} = 3.584963$$

Nilai IDF dari setiap *term* pada dokumen latihdapat dilihat pada tabel 3.12

Tabel 3.12 Nilai IDF *Term-Term* pada Dokumen Latih

Term	DF	IDF
ghana	1	3.584963
will	2	2.584963
import	1	3.584963
ton	1	3.584963
oil	3	2
annual	1	3.584963
iran	1	3.584963
agree	2	2.584963
reach	1	3.584963
tehran	1	3.584963
today	1	2.584963
nerce	1	3.584963
gas	1	3.584963
unit	1	3.584963
acquisi	1	3.584963
pct	3	2
work	1	3.584963
interest	1	3.584963
broussard	1	3.584963
field	2	2.584963
davis	1	3.584963
mln	3	2
dollar	2	2.584963
cash	1	3.584963

triton	1	3.584963
energi	1	3.584963
corp	1	3.584963
prove	1	3.584963
reserve	3	2
villespedu	1	3.584963
france	2	2.584963
paris	1	3.584963
basin	1	3.584963
estimate	1	3.584963
total	1	3.584963
barrel	1	3.584963
march	1	3.584963
perman	1	3.584963
injec	1	3.584963
offset	1	3.584963
season	1	3.584963
….		
…..		
statist	1	3.584963

Setelah diperoleh nilai TF dan IDF untuk setiap *term*, langkah selanjutnya adalah perhitungan bobot tiap *term* dengan menggunakan persamaan 2-3. Nilai *term frequency* dari setiap *term* pada *Inverted Index* masing-masing dikalikan dengan nilai IDF *term* tersebut. Pada tabel 3.9, kolom *Term Frequency* yang kosong dianggap bernilai 0. Sebagai contoh misalkan untuk menghitung nilai dari TFIDF dari *term oil*, maka langkah-langkahnya adalah sebagai berikut:

- *Posting List* dari *term oil* adalah sebagai berikut

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**	**D10**	**D11**	**D12**	**IDF**
oil	2	4	1	0	0	0	0	0	0	0	0	0	2

Maka nilai TFIDF dari *term oil* pada setiap dokumen latih adalah sebagai berikut

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**	**D10**	**D11**	**D12**	**IDF**
oil	2*2= **4**	4 *2=**8**	1*2=**2**	0	0	0	0	0	0	0	0	0	2

Perhitungan diatas juga dilakukan untuk semua *term* dokumen latih.Hasil perhitungan TFIDF tersebut dapat dilihat pada tabel 3.13.*Inverted Index* selengkapnya beserta nilai TFIDF setiap *term* dapat dilihat pada lampiran 2.

Tabel 3.13 Hasil Perhitungan TFIDF *Term* Dokumen Latih

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**	**D10**	**D11**	**D12**	**IDF**
ghana	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
will	2.58	0	0	0	0	0	0	0	2.58	0	0	0	2.58
import	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
ton	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
oil	4	8	2	0	0	0	0	0	0	0	0	0	2
annual	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
iran	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
agree	2.58	0	0	0	2.58	0	0	0	0	0	0	0	2.58
reach	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
tehran	3.58	0	0	0	0	0	0	0	0	0	0	0	3.58
today	2.58	0	0	0	0	0	2.58	0	0	0	0	0	2.58
nerce	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
gas	0	7.16	0	0	0	0	0	0	0	0	0	0	3.58

unit	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
acquisi	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
pct	0	2	4	0	0	2	0	0	0	0	0	0	2
work	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
interest	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
broussard	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
field	0	2.58	2.58	0	0	0	0	0	0	0	0	0	2.58
davis	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
mln	0	2	4	0	0	0	0	0	0	0	0	2	2
dollar	0	2.58	0	0	0	0	0	2.58	0	0	0	0	2.58
cash	0	3.58	0	0	0	0	0	0	0	0	0	0	3.58
triton	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
energi	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
corp	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
prove	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
reserve	0	0	2	2	2	0	0	0	0	0	0	0	2
villespedu	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
france	0	0	2.58	2.58	0	0	0	0	0	0	0	0	2.58
paris	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
basin	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
estimate	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
total	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
barrel	0	0	7.16	0	0	0	0	0	0	0	0	0	3.58
march	0	0	3.58	0	0	0	0	0	0	0	0	0	3.58
perman	0	0	0	3.58	0	0	0	0	0	0	0	0	3.58
injec	0	0	0	3.58	0	0	0	0	0	0	0	0	3.58

offset	0	0	0	3.58	0	0	0	0	0	0	0	0	3.58
season	0	0	0	3.58	0	0	0	0	0	0	0	0	3.58
pressur	0	0	0	3.58	0	0	0	0	0	0	0	0	3.58
….													
….													
statist	0	0	0	0	0	0	0	0	0	0	0	3.58	3.58

Klasifikasi Dokumen Uji

Preprocessing dokumen latih telah selesai dilakukan lalu langkah selanjutnya adalah mencari frekuensi *term* dari dokumen uji. Sebelumnya setiap dokumen uji juga dilakukan proses *preprocessing* seperti pada dokumen latih tetapi terdapat perbedaan pada salah satu proses nya. Pada *preprocessing* dokumen uji tidak terdapat proses *dictionary construction* atau pembentukan *inverted index* karena *term* yang diambil adalah *term* yang sudah ada di dalam *inverted index* dokumen latih. Misalkan terdapat dokumen uji pada tabel 3.4 yaitu dokumen nomor 13, dokumen tersebut pertama melalui proses *case folding, tokenizing,filtering,* dan *stemming,* dan penghtiungan frekuensi tiap *term*. Hasil *preprocessing* awal dokumen uji tersebut dapat dilihat pada tabel 3.14

Tabel 3.14 Hasil *Preprocessing* Awal Dokumen Uji

diamond =1	cut=1	dollar=1
shamrock=1	contract=1	barrel=1
corp=1	price=1	oil=1
effect=1	crude=1	today=1

Setelah mengetahui *term frequency* dari dokumen uji, langkah selanjutnya adalah melakukan proses *ontology extraction*. Proses *ontology extraction* pada dokumen latih sedikit berbeda dengan proses yang dilakukan pada dokumen latih. Setiap *term* pada dokumen uji dicari *synset* nya kemudian dicek apakah *term*

tersebut ada pada kamus dokumen latih. Jika *term* tersebut tidak muncul pada kamus dokumen latih maka dicek kembali apakah *term* pada dokumen uji tersebut ada pada daftar *synset* dokumen latih. Jika muncul, maka *term* tersebut dianggap sama dengan *term* pada dokumen latih yang bersinonim dengan *synset.* Contoh *term* ***oil*** dan ***crude.*** *Term oil* terdapat pada kamus dokumen latih maka frekuensi term nya adalah 1 sedangkan pada *term crude* tidak terdapat pada kamus dokumen latih tetapi terdapat pada daftar *synset* maka *term crude* pada dokumen uji tersebut dianggap sama dengan *term oil* maka frekuensi term oil bertambah satu sementara frekuensi term crude adalah 0. Frekuensi term dari dokumen uji setelah proses ontology extraction dapat dilihat pada tabel 3.15

Tabel 3.15 *Term Frequency* dari *Dokumen Uji Setelah Ontolgy Extraction*

diamond =1	cut=1	dollar=1
shamrock=1	contract=1	barrel=1
corp=1	price=1	oil=2
effect=1	today=1	

Langkah berikutnya adalah mencari frekuensi *term* dari dokumen uji. *Term* yang diambil dari dokumen uji adalah *term* yang sudah ada pada *inverted index.* Nilai IDF dari *term-term* tersebut diambil dari nilai IDF *term* yang telah disimpan di *inverted index*. Daftar *term,* nilai TF serta nilai bobot TFIDF dari dokumen uji dapat dilihat pada tabel 3.16

Tabel 3.16 Daftar *term* nilai TF serta nilai bobot TFIDF dari dokumen uji

DOKUMEN UJI					
TERM	**TF**	**TFIDF**	**TERM**	**TF**	**TFIDF**
ghana		0	bank		0
will		0	left		0
import		0	interven		0

ton		0	rate		0
oil	2	4	unchange		0
annual		0	inject		0
iran		0	fund		0
agree		0	categori		0
reach		0	paper		0
tehran		0	money		0
today	1	2.58	tender		0
nerce		0	meet		0
gas		0	finance		0
unit		0	minis		0
acquisi		0	central		0
pct		0	banker		0
work		0	group		0
interest		0	end		0
broussard		0	three		0
field		0	half		0
davis		0	hour		0
mln		0	japan		0
dollar	1	2.58	modest		0
cash		0	amount		0
triton		0	yen		0
energi		0	open		0
corp	1	3.58	swiss		0
prove		0	launch		0
reserve		0	seri		0

villespedu		0	month		0
france		0	certif		0
paris		0	president		0
basin		0	spokesman		0
estimate		0	marlin		0
total		0	fitzwalter		0
barrel	1	3.58	trade		0
march		0	sanction		0
perman		0	april		0
injec		0	offici		0
offset		0	emerge		0
season		0	provision		0
pressur		0	juli		0
outright		0	semiconductor		0
purchase		0	pact		0
bill		0	discuss		0
coupon		0	punit		0
afternoon		0	tariff		0
federal		0	canada		0
enter		0	surplu		0
govern		0	billion		0
secur		0	februari		0
market		0	compare		0
arrange		0	upward		0
overnight		0	revis		0
system		0	januari		0
repurchase		0	statist		0

Pembobotan dokumen uji telah selesai dilakukan dan dokumen uj tersebut akan melakukan pelatihan (*learning)* terhadap setiap dokumen latih untuk menentukan kelas dari dokumen uji. Proses pelatihan tersebut adalah menghitung kemiripan (*similarity*) antara dokumen uji dengan semua dokumen latih. Perhitungan *similarity* ini dihitung menggunakan persamaan 2-4. Contoh perhitungan *similarity* antara dokumen latih 1 dengan dokumen uji adalah sebagai berikut:

$$Sim(d_1, q) = \frac{\sum_{i=1}^{t}(w_{i1} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{i1}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

$$= \frac{(3.58 * 0) + (2.58 * 0) + (3.58 * 0) + (3.58 * 0) + (4 * 4) + \cdots + (0 * 0)}{\sqrt{(3.58^2 + 2.58^2 + 3.58^2 + 3.58 + 4^2 + \cdots + 0^2) * (0^2 + 0^2 + \cdots + 0^2 + 2.58^2)}}$$

$$= \frac{22.65}{83.10} = 0.272$$

Nilai *similarity* antara dokumen uji dengan dokumen latih nomor 1 adalah 0,272. Proses penghitungan *kemiripan* antar dokumen tersebut dilakukan lagi terhadap semua dokumen latih. Hasil perhitungan *similarity* seluruh dokumen dapat dilihat pada tabel 3.16 dn pada tabel 3.17 menggambarkan nilai *similarity* dari seluruh dokumen setelah diurutkan dari yang terbesar hingga yang terkecil.

Tabel 3.17 Hasil *Similarity* Dokumen Uji dengan Seluruh Dokumen Latih

	Hasil *Similarity*
sim(d1,uji)	0.272
sim (d2,uji)	0.332
sim (d3,uji)	0.410
sim (d4,uji)	0
sim (d5,uji)	0
sim (d6,uji)	0
sim(d7,uji)	0.082

sim(d8,uji)	0.081
sim(d9,uji)	0
sim(d10,uji)	0
sim(d11,uji)	0
sim(d12,uji)	0

Tabel 3.18 Hasil *Similarity* Dokumen Uji setelah diurutkan

	Hasil *Similarity*
sim(d3,uji)	0.410
sim (d2,uji)	0.332
sim (d1,uji)	0.272
sim (d7,uji)	0.082
sim (d8,uji)	0.082
sim (d4,uji)	0
sim(d5,uji)	0
sim(d6,uji)	0
sim(d9,uji)	0
sim(d10,uji)	0
sim(d11,uji)	0
sim(d12,uji)	0

Langkah selanjutnya adalah mengambil sebanyak *k* terbesar dari nilai *similarity*. Misalkan nilai *k* yang ditentukan sebanyak 5 maka nilai *similarity* yang digunakan adalah seperti yang digambarkan pada tabel 3.18

Tabel 3.19 Kategori Dokumen Latih yang Telah Diambil Sesuai Jumlah *K*

Dokumen	Nilai *Similarity*	Kategori
D3	0.410	*Crude*
D2	0.332	*Crude*
D1	0.272	*Crude*
D7	0.082	*Money-Fx*
D8	0.082	*Money-Fx*

Proses selanjutnya adalah melakukan proses *scoring* pada setiap kategori untuk menentukan kategori dari dokumen uji. Terdapat empat kategori pada dokumen latih maka terdapat empat skor untuk setiap kategori. Berdasarkan persamaan 2-5 skor setiap kategori adalah

Score (uji,*interest*)= ((0.0410*0) + (0.332*0) + (0.272*0) + (0.082*0) + (0.082*0))= 0

Score (uji,*trade*)= ((0.0410*0) + (0.332*0) + (0.272*0) + (0.082*0) + (0.082*0))= 0

Score (uji,*crude*)= ((0.0410*1) + (0.332*1) + (0.272*1) + (0.082*0) + (0.082*0))= **0.6441**

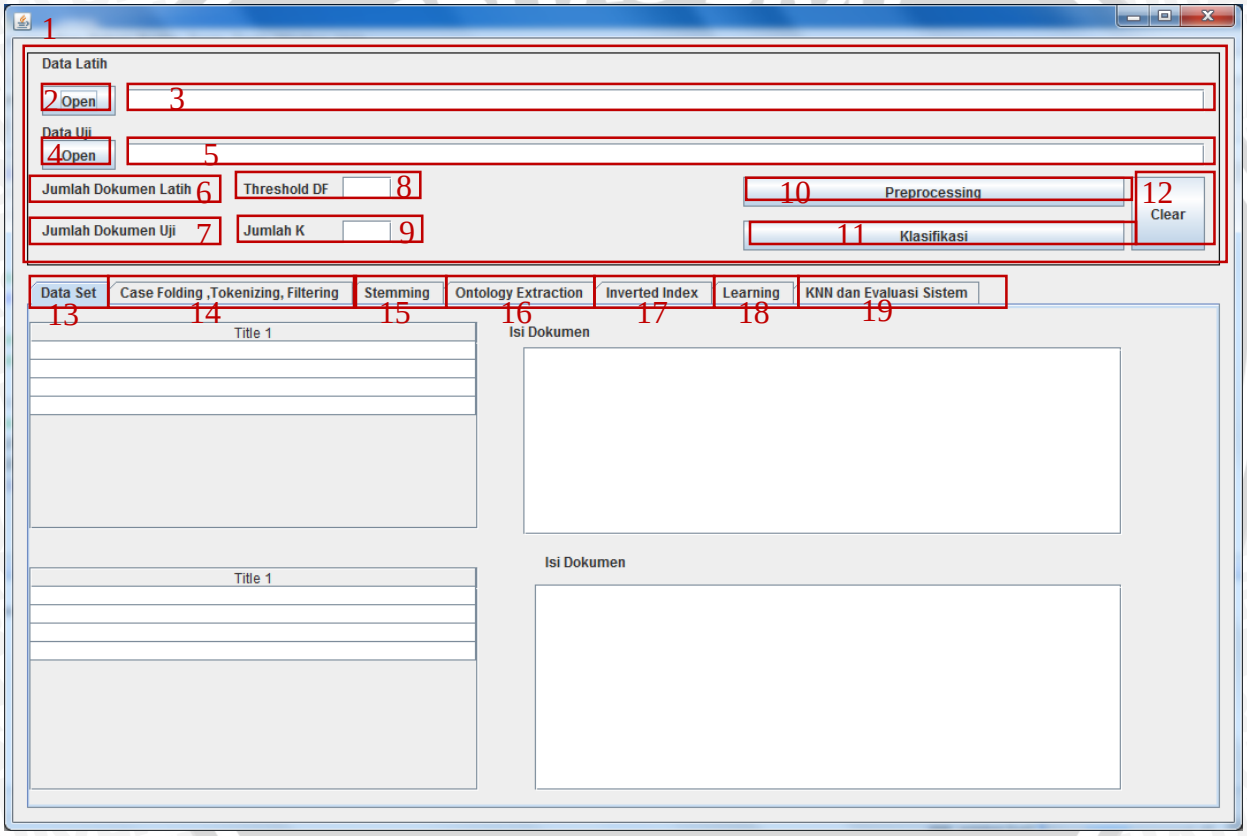
Score (uji,*money-Fx*)= ((0.0410*0) + (0.332*0) + (0.272*0) + (0.082*1) + (0.082*1))= 0.164

Berdasarkan hasil penghitungan skor setiap kategori, kategori dari dokumen uji ditentukan oleh nilai skor kategori yang paling tinggi. Pada hasil perhitungan, kategori *crude* memiliki nilai paling tinggi maka dokumen uji tersebut diklasifikasikan pada kategori *crude.*

Setelah melakukan beberapa pengujian,langkah selanjutnya adalah menghitung *precision, recall, F-Measure* dan *Accuracy* berdasarkan persamaan 2-7 sampai dengan 2-10.

3.5 Rancangan Antar Muka

Rancangan antar muka dari sistem klasifikasi berita berbahasa Inggris menggunakan algoritma KNN berbasis ontologi secara umum terdiri dari bagian *input*, *output*, tombol *preprocessing* dan tombol klasifikasi. Input dari sistem berupa dokumen latih dan dokumen uji, *threshold* DF untuk proses *feature selection* dan jumlah *k neighbor*. Output dari sistem adalah kategori dokumen uji. Setiap proses pada sistem ini ditampilkan pada setiap *tab*. Gambar 3.21 menunjukkan rancangan antar muka secara umum.



Gambar 3.23. Rancangan Antarmuka Sistem

Sumber : [Metodologi dan Perancangan]

Penjelasan dari setiap bagian rancangan antar muka sistem yang telah ditandai oleh label angka berwarna merah adalah sebagai berikut:

1. Panel Input data beserta tombol proses.
2. Tombol untuk membuka dan memilih *folder* atau direktori dari dokumen-dokumen latih.
3. *Field* untuk menampilkan *folder* atau direktori dari dokumen-dokumen latih.
4. *Button* untuk membuka dan memilih *folder* atau direktori dari dokumen-dokumen uji.
5. *Field* untuk menampilkan *folder* atau direktori dari dokumen-dokumen latih.
6. *Text Field* untuk menampilkan keterangan jumlah dokumen latih.
7. *Text Field* untuk menampilkan keterangan jumlah dokumen latih.
8. *Field* untuk memasukkan nilai *threshold document frequency* untuk proses *feature selection.*
9. *Field* untuk memasukkan nilai jumlah *k-neighbor* untuk proses klasifikasi.
10. Tombol untuk mengeksekusi proses *preprocessing* pada dokumen latih dan dokumen uji.
11. Tombil untuk mengeksekusi proses klasifikasi pada dokumen-dokumen uji.
12. Tombol untuk menghapus semua *field*.
13. *Tab* dataset berisi tabel dokumen latih dan dokumen uji beserta isi dari setiap dokumen.
14. *Tab case folding, tokenizing, dan filtering* untuk menampilkan hasil dari proses *case folding, tokenizing,* dan *filtering* dari setiap dokumen.
15. *Tab stemming* menampilkan hasil proses *stemming* dari dokumen latih dan dokumen uji.
16. *Tab ontology construction* menampilkan hasil pencarian sinonim setiap kata pada *inverted index.*

17. *Tab inverted index* menampilkan isi dari *nverted index* yang dibangun pada saat proses *Dictionary Construction.*
18. *Tab learning* menampilkan hasil dari *cosine similarity* dari dokumen latih terhadap setiap dokumen uji. Hasil dari *cosine similarity* juga ditampilkan secara berurut serta akan mengambil sejumlah k teratas dari nilai *cosine similarity* tersebut.
19. *Tab KNN* menampilkan kategori dari dokumen uji serta hasil evaluasi dan rata-rata evaluasi dari algoritma KNN berbasis ontologi.

3.6 Rancangan Uji Coba

Setelah sistem selesai diimplementasikan maka langkah selanjutnya adalah melakukan serangkaian uji coba untuk mengetahui kinerja dari sistem ini. Pengukuran evaluasi sistem yang digunakan adalah *macroaverage precision, macroaverage recall* dan *macroaverage F1-measure.*

Pengujian pertama adalah mengetahui pengaruh pereduksian dimensi ruang vektor teks menggunakan *document frequency* (DF) *threshold* terhadap akurasi sistem. Pengujian kedua adalah mengetahui pengaruh jumlah "*k*" tetangga pada algoritma *K-Nearest Neighbor* (KNN) terhadap performa klasifikasi akurasi sistem. Pengujian sejumlah dokumen uji akan dilakukan beberapa kali dengan mengganti nilai parameter dari *document frequency* dan jumlah "*k*" tetangga. Hasil evaluasinya akan ditampilkan dalam bentuk tabel seperti pada yang ditunjukkan pada tabel 3.19.

Tabel 3.20 Contoh Tabel Hasil Evaluasi Terhadap Perubahan Parameter

Uji Coba	k	*Recall*	*Precision*	*F1-Measure*
1	1			
2	2			
3	3			
…	…			
25	25			

Pengujian selanjutnya adalah pengaruh jumlah dokumen latih dan dokumen uji terhadap akurasi sistem. *Corpus* berita berbahasa Inggris akan dibagi menjadi dokumen latih dan dokumen uji dengan rasio yang berbeda. Hasil dari pengujian rasio dokumen ini dapat dilihat pada tabel 3.20.

Tabel 3.21 Contoh Hasil Evaluasi Terhadap Rasio Dokumen

Uji Coba	Jumlah Dok Latih	Jumlah Dok Uji	*Recall*	*Precision*	*F1-Measure*
1	40	20			
2	80	20			
3	120	20			
…					
n					