BAB 2
LANDASAN TEORI

2.1 Proposal Skripsi

Skripsi adalah karya ilmiah dalam suatu bidang studi yang dibuat oleh para mahasiswa strata satu pada masa akhir studi sebagai persyaratan untuk menyelesaikan program studi, dan studi tersebut harus berdasarkan pada suatu penelitian ilmiah, baik penelitian lapangan, penelitian perpustakaan atau penelitian pengembangan[5]. Kemudian pengertian proposal skripsi adalah rencana penelitian dari seorang mahasiswa yang akan mengadakan penelitian karya ilmiah berupa skripsi, proposal merupakan bukti kemampuan mahasiswa dalam pembuatan rancangan penelitian dan mengembangkan ilmu pada salah satu bidang keilmuan tertentu. Dalam menyusun proposal skripsi harus memperhatikan sistematika penulisannya, sistematika penulisan sendiri disusun untuk memberikan gambaran umum tentang penelitian yang akan dilakukan. Sistematika penulisan proposal tergantung pada jenis penelitiannya, tapi pada umumnya isi dari sistematika penulisan seperti dibawah ini:

1. Halaman Judul

   Berisi judul dari proposal skripsi, logo universitas, dan keterangan fakultas, program studi dan nama universitas beserta tahun penyusunan.
2. Daftar Isi

   Berisi daftar menu dan sub menu yang ada di proposal skipsi.
3. Daftar Tabel

   Berisi daftar tabel yang digunakan pada proposal skripsi.
4. Daftar Gambar

   Berisi daftar gambar yang terdapat pada proposal skripsiDaftar Lampiran

   Berisi daftar data penunjang penelitian.
5. Bab I Pendahuluan

   Berisi latar belakang, rumusan masalah, maksud dan tujuan, batasan masalah, metodologi penelitian dan sistematika penulisan.

6. Daftar Pustaka

Berisi seluruh kepustakaan yang digunakan dalam penyusunan proposal.

Dalam pegajuan proposal di Teknik Informatika UNIKOM, mahasiswa harus menyiapkan data proposal dengan sistematika penulisan seperti yang telah dijelaskan diatas, mahasiswa juga harus membawa data pendukung sebagai penunjang agar proposal yang diajukan dapat diterima oleh dosen penguji dalam sidang proposal.

2.2 *Text Mining*

Text mining menurut Feldman, R. Dan Sanger, J.," *text mining* adalah sebuah proses pengetahuan *intensif* dimana pengguna berinteraksi dan bekerja dengan sekumpulan dokumen dengan menggunakan beberapaalat analisis"[6]. *Text mining* mencoba untuk mengestrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi dari suatu pola menarik dan sumber datanya adalah berupa dokumen.

2.3 Metode *Preprocessing*

Preprocessing adalah proses pengolahan data asli yang akan dipersiapkan untuk diolah pada tahap seanjutnya. Masukan data awal yatitu berupa dokumen tahapan *preprocessing* yang akan dilakukan pada penelitian ini meliputi tokenisasi kalimat, tokenisasi kata, *case folding*, proses *filtering*, *stopword removal* dan *term weighting* atau perhitungan bobot TF-IDF.

2.3.1 Tokenisasi kalimat

Pemisahan kalimat merupakan proses pemisahan teks pada dokumen menjadi kumpulan kalimat. Teknik yang digunakan dalam pemisahan kalimat adalah memisahkan kalimat dengan tanda titik (.) sebagai *delimeter*.

2.3.2 Tokenisasi kata

Proses tokenisasi kata adalah tahap pengecekan dari karakter pertama sampai dengan karakter terakhir. Apabila ditemukan karakter pemenggal atau

delimeter seperti spasi atau tanda baca, maka teks akan dipisahkan dan dimasukkan dalam kumpulan kalimat[7]. Contoh dari tahap ini adalah sebagai berikut :

Teks *Input*:

unikom adalah sebuah universitas di kota bandung.

Hasil Token:

unikom adalah sebuah universitas di kota bandung

2.3.3 *Case folding*

Case folding adalah proses menyeragamkan semua teks ke dalam case yang sama menjadi huruf kecil (lowercase) atau huruf capital (uppercase). Pada penelitian ini huruf dalam teks diseragamkan menjadi huruf kecil. Contoh dari tahap ini adalah sebagai berikut :

Teks *Input*:

UNIKOM adalah sebuah Universitas di kota Bandung.

Hasil

unikom adalah sebuah universitas di kota bandung.

2.3.4 *Filtering*

Tahap *Filtering* adalah tahap yang mengacu pada proses untuk menentukan istilah yanag akan mempresentasikan atau menggambarkan isi dokumen dan membedakan dokumen satu dan yang lainnya pada suatu koleksi. Jadi *Filtering* merupakan proses dimana teks selain karakter “a” sampai “z” dan spasi akan dihilangkan dan hanya menerima spasi. Tahap-tahapan yang dilakukan yaitu:

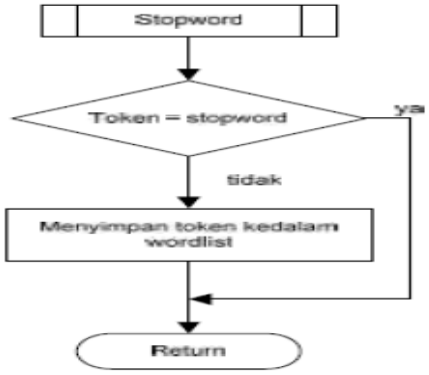
1. Membca data masukan yang dilakukan.
2. Mengecek apakah ada huruf selain “a” sampai ”z” dan “ ” (spasi). Jika tidak ditemukan, maka algoritma berakhir. Jika ada.
3. Hapus karakter yang bukan termasuk “a” sampai “z” dan “ ” (spasi).
4. Algoritma selesai.

2.3.5 Stopword Removal

Tahap *Stopword Removal* adalah tahap mengambil kata – kata penting dari hasil token, menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *word list* (menyimpan kata penting). Proses pembuangan *stopword* dimaksudkan untuk mengetahui suatu kata masuk ke dalam *stopword* atau tidak. Pada tahapan *stopword removal* ini, digunakan *stoplist* yang terdapat pada penelitian Tala, karena pada penelitian tersebut menjelaskan bahwa dengan *stoplist* tersebut dapat meningkatkan presisi kata meskipun tidak signifikan, dengan R-precision yang didapatkan sebesar 0,6649[8]. Kemudian pada penelitian ini, akan menambah dan membuang kata-kata yang ada pada penelitian tala. Proses membuang kata-kata yang ada pada penelitian tala dilakukan agar kata-kata penting yang ada pada stoplist talam tidak menghilang kata-kata yang penting yang dapat menggambarkan dokumen dan proses menambahkan kata dalam stoplist, dilakukan untuk membantu menghilangkan kata yang tidak menggambarkan tentang isi dokumen yang akan di proses. *Stopword removal* adalah proses menghilangkan kata yang 'tidak relevan' dari sebuah dokumen teks dengan cara membandingkannya dengan *stoplist* yang ada, Berikut adalah tahapan proses *stopword* [7] :

a. Kata hasil token *stemming* dibandingkan dengan tabel *stopword.*
b. Dilakukan pengecekan apakah token sama dengan tabel *stopword* atau tidak.
c. Jika token sama dengan tabel *stopword* mak*a akan di-remove.*
d. Jika tidak maka akan langsung ditampilkan.

Untuk lebih jelas dapat dilihat pada gambar 2.1.



Gambar 2.1 *Flowchart Stopword*[7]

Berikut adalah contoh dari tahap *Stopword Removal* :

Hasil token :

unikom	universitas	bandung	
adalah	di	sebuah	kota

Hasil *Filter* :

unikom
universitas
kota
bandung

2.3.6 Term Weighting / *TF-IDF*

Tahap *Term weighting/ TF-IDF* adalah metode yang digunakan dalam menghitung bobot setiap kata. Metode yang sering digunakan yaitu menggunakan

metode TF-IDF (*Term Frequency - Inverse Document Frequency*). Pada metode ini, perhitungan bobot *term*(kata) dalam sebuah dokumen dilakukan dengan mengitung nilai *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) degan cara mengalikannya[9].

Term Frequency (TF) adalah faktor yang menentukan bobot *term* pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan suatu kata (*term frequency*) diperhitungkan dalam pemberian bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu term dalam dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuian yang semakin besar.

Inverse Document Frequency (IDF) adalah pengurangan dominansi *term* yang sering muncul di berbagai dokumen. Hal ini diperlukan karena term yang banyak muncul di berbagai dokumen, dapat dianggap sebagai term umum (*common term*) sehingga tidak penting nilainya. Sebaliknya faktor kejarangmunculan kata (*term scarcity*) dalam koleksi dokumen harus diperhatikan dalam pemberian bobot. Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon term*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebal ikan secara proposional dengan urutannya[9].

Untuk menghitung bobot kata (W_i) dalam dokumen dihitung dengan menggunakan rumus:

$$\boldsymbol{W_i = DF(t_i, d) \times IDF(t_i)} \qquad (2.1)$$

Dimana:

W_i = bobot kata *(term)* dalam dokumen d

i = indeks kata (*term).*

$TF(t_i, d)$ = banyaknya kata (*term*) t_i yang muncul dalam dokumen d.

$IDF(t_i)$ = *inverse document frequency* dari kata (*term*) t_i.

Untuk menghitung $IDF(t_i)$ dapat menggunakan rumus:

$$\boldsymbol{IDF(t_i) = \log\left(\frac{D}{DF(t_i)}\right)} \quad (2.2)$$

Dimana :

$IDF(t_i)$ = *inverse document frequency* dari kata (*term*) t_i.

D = jumlah dokumen keseluruhan.

$DF(t_i)$ = jumlah dokumen yang memiliki kata (*term*) (t_i).

2.4 Peringkasan Teks

Peringkasan teks adalah suatu kegiatan merangkum teks dari sebuah dokumen, dimana tujuan rangkuman ini untuk menyajikan gagasan utama dalam sebuah dokumen dalam ruang/penyimpanan yang lebih sedikit[10]. Pada peringkasan teks terbagi menjadi dua cara yaitu secara manual dan otomatis.Pada peringkasan teks otomatis terdapat dua prosedur yang dapat dilakukan yaitu ekstraksi dan abstraksi. Dalam proses ekstraksi menggunakan algoritma ekstraksi kata kunci dimana algoritma ini menyalin unit-unit teks yang dianggap penting atau paling informatif dati teks yang akan diringkas, sedangkan yang dilakukan pada abtraksi adalah membangun sebuah representasi senabtik internal seperti mengambil intisari dari teks, kemudian membuat ringkasan dengan menciptakan kalimat-kalimat aru yang mempresentasikan intisari dari teks. Peringkasan teks dapat dilakukan pada satu dokumen dan banyak dokumen(*multy document*).

2.5 Keyword Extraction

Pada tahap *keyword extraction* ini, dilakukan untuk untuk memperkecil matriks yang akan di simpan sebagai data latih maupun yang akan digunakan sebagai data uji. Tahap peringkasan metode *keyword extraction* terdiri dari dua tahap yaitu pertama dengan menggunakan algoritma ekstraksi kunci, lalu *selector* rasio kompresi [10]. Pada tahap pertama dokumen diberi tolak ukur dengan algoritma ekstraksi kata kunci dan kemudian menghitung Tf/Idf seperti yang telah dilakukan sebelumnya, yaitu pada tahap penghitungan Tf/Idf kalimat. Dari hasil

Tf/Idf, dijumlah semua Tf/Idf untuk setiap kalimat dan dari hasil tersebut diurutkan dari yang terbesar, setelah diurutkan maka lakukan kompresi sebesar 50% dari dokumen asli. Setelah pemilihan sebanyak 50%, proses selanjutnya adalah mengatur ulang satu kalimat daru nilai yang terbesar sampai yang paling rendah, dan teks baru ini dengan susunan kalimat baru akan menjadi teks akhir yang diringkas Kompresi dilakukan sebanyak 50% karena pada penelitian sebelumnya, didapatkan akurasi yang lebih baik dibandingkan dengan kompresi sebesar 20 dan 30%. [10].

2.6 Algoritma K- Nearest Neighbour

K-Nearest Neighbor adalah sebuah metode dengan tujuan utuk mengklasifikasi objek baru berdasarkan atribut dan training samples dimana data objek yang di klasifikasi berdasarkan data pembelajara yang jarak paling dekat dengan objek tersebut, dimana metode knn adalah salah satu metode yang sederhana dan mudah diimplementasikan[2]. *K-Nearest* Neighbor (K-NN) merupakan teknik klasifikasi yang melakukan prediksi secara tegas pada data uji berdasarkan perbandingan K tetangga terdekat Pada algoritma K-NN , data berdimensi *N*, dapat dihitung jarak dari data tersebut ke data yang lain, nilai jarak ini yang digunakan sebagai nilai kedekatan/ ketidakmiripan antara data latih dan data uji. Pada algoritma K-NN, sebuah data uji z = (x',y'), dimana x' ada vektor/atribut data uji, sedangkan y' adalah label kelas data uji yang belum diketahui, kemudian menghitung jarak (atau kemiripan) data uji ke setiap data latih d(x',x), kemudian mengambil K tetangga terdekat pertama dalam D_Z. Setelah itu dihitung jumlah data yang mengikuti kelas yang ada dari K tetangga tersebut[8].

2.7 Fuzzy K- Nearest Neighbour

Algoritma ini memberikan nilai membership sebagai fungsi pola jarak dari sejumlah himpunan *K-nearest neighbour* dan pemberian nilai keanggotaan *neighbour* pada kelas tertentu. Sehingga pada algoritma ini data ui yang akan diklasifikasikan akan memiliki nilai keanggotaan pada semua kelas uji. Klasifikasi

algoritma ini nantinya akan memilih nilai keanggotaan kelas pada data uji yang paling tinggi[1]. Berikut merupakan persamaan untuk memberikan nilai keanggotaan pada data uji W_j dengan menggunkan metode *euclidean distance*. Jarak *eucidean distance* antara vektor data uji W_j dengan vektor data latih W_i dihitung menggunakan rumus dibawah ini:

$$\|\boldsymbol{W_i - W_j}\| = \left(\sum_{i=1}^{n} |\boldsymbol{W_i - W_j}|^2\right)^{1/2} \quad (2.3)$$

Dimana :

$\|W_i - W_j\|$= jarak e*uclidean* vektor data uji dan dokumen latih.

W_j = bobot hasil TF-IDF ringkasan data uji.

W_i = bobot hasil TF-IDF ringkasan data latih ke-i.

i = index *term* ke-*i*.

n = jumlah *term* keseluruhan hasil *text preprocessing*.

Untuk penentuan nilai keanggotaan kelas ke-*i* pada tetangga ke-*j* dihitung menggunakan rumus dibawah ini:

$$\boldsymbol{\mu}_{\text{ij}} = \begin{cases} \mathbf{1}, & \boldsymbol{j = i} \\ \mathbf{0}, & \boldsymbol{j \neq i} \end{cases} \quad (2.4)$$

Dimana :

μij = nilai keanggotaan kelas ke-*i* pada tetangga ke-*j*.

i = kelas *genre*.

j = Tetangga terdekat (K).

Penentuan nilai *membership* kelas atau pemberian nilai keanggotaan kelas pada data uji x dapat dihitung, dengan menggunakan rumus dibawah ini:

$$\boldsymbol{\mu(x)} = \frac{\sum_{i=1}^{k} \mu_{ij} \left(\frac{1}{\|x - x_j\|^{2/(m-1)}}\right)}{\sum_{i=1}^{k} \left(\frac{1}{\|x - x_j\|^{2/(m-1)}}\right)} \quad (2.5)$$

Dimana:

$\mu(x)$ = nilai keanggotaan kelas ke-*i* pada data uji *x*.

m = bobot pangkat (*weight exponent*).

Jika jarak yang digunakan adalah *Manhattan* maka m = 1.

Jika jarak yang digunakan adalah *Euclidean* maka m = 2.

Jika jarak yang digunakan adalah *Chebyshev* m = ∞ maka.

2.8 Fuzzy K- Nearest Neighbour in Every Class

Metode *Fuzzy K-Nearest Neighbor in every Class* (FK-NNC) adalah sebuah metode pengklasifikasian dengan menggunakan K tetangga terdekat pada setiap kelas dari sebuah data uji dan akan dijumlahkan dengan tetangga dikelas x yang ditemukan dengan menggunakan metode FK-NNC[11]. Kerangka dari FK-NNC masih menggunakan kerangka FK-NN dimana nilai keanggotaannya berinterval [0.1] dimana Jumlah nilai keanggotaan sebuah data pada semua kelas sama dengan 1, seperti pada rumus 2.8, berikut :

$$\sum_{j=1}^{c} \boldsymbol{U_i = 1},\ \boldsymbol{0 \le U_j \le 1} \qquad (2.6)$$

Dimana :

U_j = nilai keanggotaan data uji U_i ke kelas ke- j.

Setiap data uji harus dicarikan K tetangga terdekat pada setiap kelas menggunakan rumus (2.4), sehingga untuk setiap kelas akan dikalikan sebanyak kelas dengan K tetangga yang didapat. Selanjutnya jarak uji U_j ke semua K tetangga dari setiap kelas ke-j dijumlahkan, dengan rumus :

$$\boldsymbol{S_i = \sum_{j=1}^{K} d(W_i, W_j)^{-2/m-1}} \qquad (2.7)$$

Dimana:

S = jarak kumulatif ke-i.

i = 1, 2, 3, 4, 5.

d = akumulasi jarak data uji W_j ke tetangga dalam kelas ke-$_j$ sebanyak C kelas.

m = pangkat bobot .

Jika jarak yang digunakan adalah *Manhattan* maka m = 1.

Jika jarak yang digunakan adalah *Euclidean* maka m = 2.

Jika jarak yang digunakan adalah *Chebyshev* m = ∞ maka.

Selanjutnya, akumulasi jarak data uji W_j ke setiap kelas dibangunkan, simbolkan D, rumusnya sebagai berikut :

$$\boldsymbol{D = \sum_{i=1}^{c}(S_i)} \quad (2.8)$$

Dimana :

D = jumlah akumulasi dari semua kelas.

Untuk mendapatkan nilai keanggotaan data uji W_j pada setiap kelas ke-i (ada C kelas), dengan menggunakan rumus sebagai berikut :

$$\boldsymbol{U_i = \frac{S_i}{D}} \quad (2.9)$$

Dimana :

U_i = Keanggotaan kelas k-i.

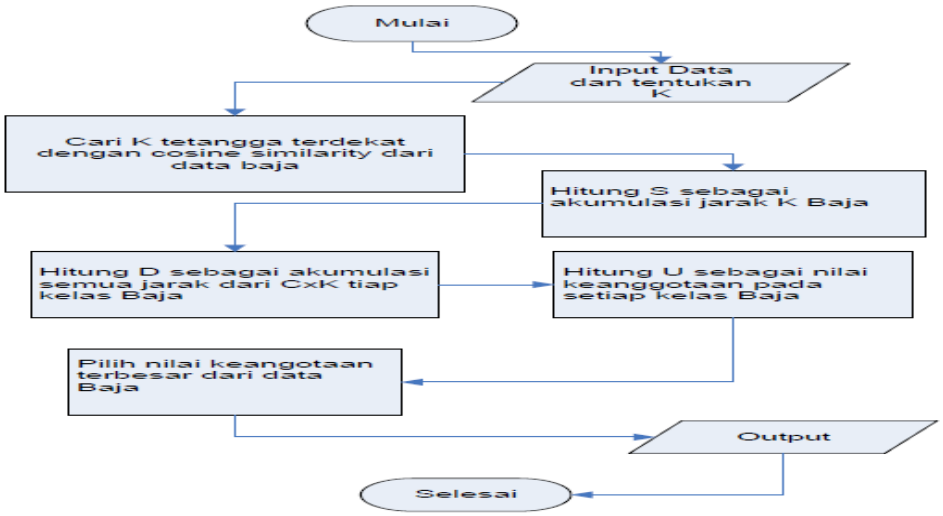
Kemudian untuk menentukan kelas hasil prediksi data uji W_j, dipilih kelas dengan nilai keanggotaan terbesar dari data U_i. Dengan rumus :

$$\boldsymbol{y' = \arg\max(U_i)} \quad (2.10)$$

Algoritma FK-NNC untuk sebuah data uji, disajikan sebagai berikut :

1. Cari K tetangga terdekat pada setiap kelas menggunakan rumus (2.3)
2. Hitung S sebagai akumulasi jarak K tetangga pada setiap kelas, menggunkan rumus (2.7)
3. Hitung D sebagai akumulasi semua jarak dari CxK tetangga, menggunakan rumus (2.8)
4. Hitung U sebagai nilai keanggotaan data pada setiap kelas, menggunakan rumus (2.9)
5. Pilih nilai keanggotaan terbesar dengan menggunakan rumus (2.10), kelas dengan nilai keanggotaan terbesar menjadi kelas hasil prediksi untuk data uji tersebut.

Tahapan proses *Fuzzy K-Nearest Neighbor in every Class* (FK-NNC) dapat dilihat pada gambar 2.2.

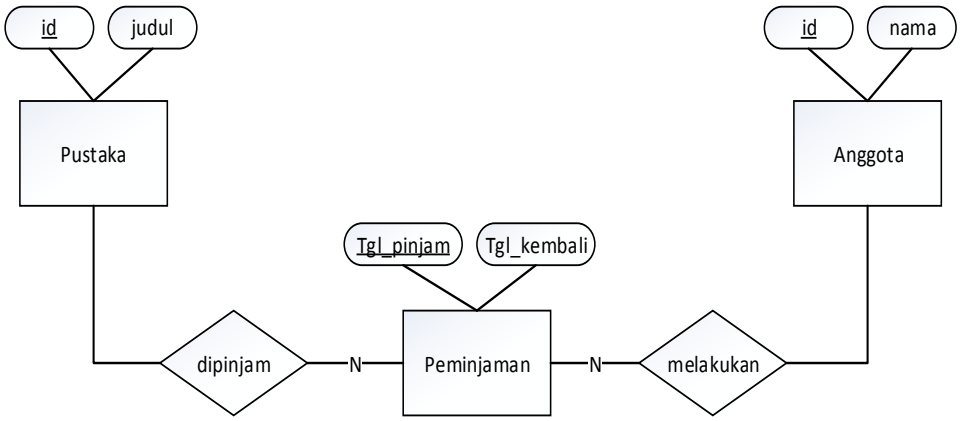


Gambar 2.2 Proses *Fuzzy K-Nearest Neighbor in every Class*[11]

2.9 Pemprograman Terstruktur

2.9.1 ERD

ERD adalah diagram yang menggambarkan keterhubungan antar dara secara konseptual[12]. ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Dengan ERD, model dapat diuji dengan mengabaikan proses yang dilakukan. Contoh dari ERD dapat dilihat pada gambar 2.3.



Gambar 2.3 Contoh ERD

ERD pertama kali dideskripsikan oleh Peter Chen yang dibuat sebagai bagian dari perangkat lunak *CASE* Komponen komponen yang termasuk dalam ERD antara lain[11], adalah :

1. Entitas (*Entity*)

   Sebuah baramg atau obyek yang dapat dibedakan dari *obyek* lain.

2. Relasi (*Relationship*)

   Asosiasi dua tau lebih entitas dan berupa kata kerja.

3. Garis penghubung

   Properti yang dimiliki untuk menghubungkan setiap data yang berelasi.

4. *Atribut*

   Berupa data yang dimiliki suatu entitas.

Angka yang menunjukan banyaknya kemunculan suatu objek terkait dengan kemunculan objek lain pada suatu relasi. Gambar kardinalitas dapat dilihat pada Gambar 2.1.

NO	NOTASI	DERAJAT RELASI MAKS-MIN	NAMA RELASI	KETERANGAN
1		(0 , N)	Zero to Many	Relasi banyak dengan salah satu atribut dapat tidak berelasi (relasi kosong)
2		(1 , N)	One to Many	Relasi banyak dengan salah satu atribut harusmempunyai relasi
3		(1 , 1)	One To One	Relasi satu (one) dengan salah satu atribut harus berelasi
4		(0 , 1)	Zero To One	Relasi satu dengan salah satu atribut boleh tidak berelasi (relasi kosong)

Gambar 2.1 Tabel Kardinalitas

2.9.2 DFD

Data flow diagram adalah menggambarkan aliran data dalam sistem, sumber dan tujuan data, proses yang mengolah data tersebut, dan tempat penyimpanan data[12]. Data *flow* diagram ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya untuk fungsi-fungsi sistem yang merupakan bagian yang lebih penting dan *kompleks* dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, data *flow* diagram adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. Data *flow* diagram ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa

maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program

Beberapa komponen yang digunakan di DFD adalah:

a. Entitas Luar (*Boundary*)

   Entitas luar merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem.

b. Arus Data (*Data Flow*)

   Arus data diberi simbol suatu panah. Arus data mengalir diantara proses (*process*), simpanan data (*data store*) dan kesatuan luar (*external entity*).

c. Sumber/Tujuan Data

   Menggambarkan entitas yang berinteraksi dengan sistem yang berada di luar ruang lingkup sistem atau entitas yang berfungsi sebagai *producer/consumer* dari sistem.

d. Penyimpanan Data (*Data Store*)

   Simpanan data merupakan simpanan dari data yang dapat berupa *file* atau *database* di sistem komputer, arsip atau catatan manual, agenda atau buku. Simpanan data DFD dapat di simbolkan dengan sepasang garis horizontal.

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. Untuk memudahkan analisa dimulai dengan:

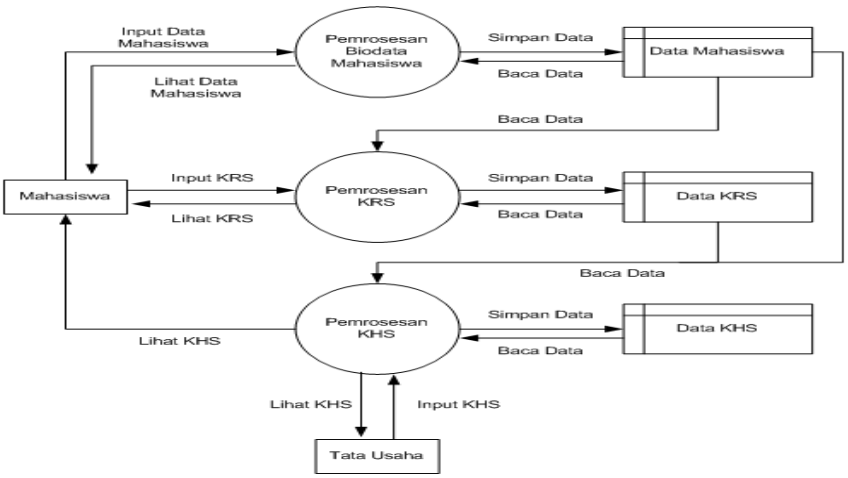
a. Diagram Konteks (*Context Diagram)*

   Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. *Diagram konteks* merupakan level tertinggi dari DFD yang menggambarkan seluruh *input* ke sistem atau *output* dari sistem.

b. Diagram Rinci (*Level Diagram*)

   Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram *zero* atau diagram level atasnya.

Berikut contoh dari DFD dapat dilihat pada agambar 2.4.



Gambar 2.4 Contoh DFD Sistem Akademik

2.10 Software Aplikasi

Software aplikasi adalah penjelasan aplikasi-aplikasi yag digunakan dalam pembentukan program yang akan dibangun, berikut adalah aplikasi yang digunakan.

2.10.1 PHP

PHP adalah bahasa pemrograman *script server-side* yang didesain untuk pengembangan web[13]. Kependekan dari PHP *Hypertext Preprocessor*, bahasa *interpreter* yang mirip dengan bahasa C dan Perl yang memiliki kesederhanaan dalam perintah, PHP dapat digunakan bersama dengan HTML sehingga memudahkan dalam pembangunan aplikasi web dengan cepat. PHP dapat digunakan untuk meng-*update* basis data dan menciptakan basis data. *Interpreter* adalah sebuah program yang digunakan untuk membaca *file* yang berisi kode program yang akan dijalankan kemuadian *interpreter* tersebut akan meminta CPU untuk menlakukan perintah yang diterimanya.

Seperti halnya program *open source* lainnya, PHP dibuat di bawah lisensi GNU, *General Public License,* yang dapat di *download* gratis melalui situs http://www.php.net. Awalnya, PHP diciptakan oleh Andi Gutmans untuk

menghitung jumlah pengunjung yang mengakses *homepage* yang dbuatnya. PHP banyak mendukung basis data, seperti MYSQL, PostgresSQL, Interbase, ODBC, mSQL, Oracle, dan Sybase.

Pada penelitian ini PHP digunakan sebagai bahasa pemograman utama dalam membangun sistem *Word Sense Disambiguation* dimana kode-kode PHP hanya akan dieksekusi di server kemudian hasilnya dikirim ke browser dalam format HTML.

2.10.2 JavaScript

Javascript adalah bahasa skrip yang populer di internet dan dapat di sebagian besar *web browser* seperti *mozilla firefox* dan lain-lain, *javascript* merupakan bahasa pemrograman web yang bersifat *clientside programming language*. Dengan adanya bahasa ini, kemampuan dokumen HTML menjadi semakin luas. Sebagai contoh, dengan menggunakan JavaScript dimungkinkan untuk memvalidasi masukan- masukan pada formulir sebelum formulir dikirimkan ke server. Javascript bukanlah bahasa Java dan merupakan dua bahasa yang berbeda. Javascript diinterpretasikan oleh klien (kodenya bisa dilihat pada sisi klien), sedangkan kode Java dikompilasi oleh pemrogram dan hasil kompilasinyalah yang dijalankan oleh klien.

Pada penelitian ini *Javascript* diimplementasikan pada pembuatan antarmuka tambah data, apabila *user* ingin menambah data cerpen baru sehingga tampilan lebih interaktif dan dinamis.

2.10.3 Xampp

XAMPP adalah perangkat lunak (*free software*) bebas, yang mendukung untuk banyak sistem operasi, yang merupakan kompilasi dari beberapa program. Fungsi XAMPP sendiri adalah sebagai server yang berdiri sendiri (localhost), yang terdiri beberapa program antara lain : Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam

GNU *General Public License* dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis[14]. Untuk mendapatkanya XAMPP anda dapat mendownload langsung dari web resminya. Pada penelitian ini XAMPP berguna sebagai web server.

2.10.4 MySql

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multithread, multi user, dengan sekitar 6 juta instalasi di seluruh dunia. Pengaksesan Sql dapat dilakukan apabila komputer telah terhubung dengan *server*, berbeda dengan database desktop, dimana segala pemrosesan data harus dilakukan pada komputer yang bersangkutan.

MySQL dapat juga dikatakan sebagai *Relational Database Management System* (RDBMS), yaitu hubungan antar tabel yang berisi data-data pada suatu database, dengan demikian dapat mempercepat pencarian suatu data. Tabel-tabel tersebut di-*link* oleh suatu relasi yang memungkinkan kombinasi data dari beberapa tabel ketika *user* menginginkan tampilannya informasi dari suatu database.

MySQL merupakan database yang dikembangkan dari bahasa SQL (*Structured Query Language*), SQL merupakan bahasa terstruktur yang digunakan untuk interaksi antara script program dengan database server dalam hal pengolahan data. Dengan SQL, maka dapat dibuat tabel yang akan diisi data, memanipulasi data seperti menambah, menghapus dan mengupdate data.

2.10.5 Confusion Matrix

Confusion matrix merupakan sebuah metode perhitungan yang digunakan untuk mencari keakuratan pada hasil klasifikasi[2]. Berikut merupakan tabel 2.2. *confusion matrix*:

Tabel 2.2 *Confusion Matrix*

Kelas		Prediksi	
		1	0
Kelas sebenarnya	1	TP	FN
	0	FP	TN

Keterangan:

a. *True Positive* (TP), merupakan jumlah dokumen dari kelas 1 yang benar diklasifikasikan sebagai kelas 1.
b. *False Positive* (FP), merupakan jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
c. *False Negative* (FN), merupakan jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.
d. *True Negative* (TN), merupakan jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.

Kemudian akan dihitung akurasinya, akurasi merupakan perhitungan antara jumlah prediksi benar dan dibagikan dengan jumlah prediksi, untuk menghitung akurasi digunakan rumus sebagai berikut:

$$akurasi = \frac{jumlah\ prediksi\ benar}{jumlah\ seluruh\ prediksi\ yang\ dilakukan} \times 100\ \%$$