**PROPOSAL**

**IMPLEMENTASI METODE *ROCCHIO* UNTUK KLASIFIKASI JENIS PELAPORAN DI LAMAN PROPAM POLDA SULTRA**

Diajukan Untuk Memenuhi

Salah Satu Syarat Memperoleh Gelar Sarjana Teknik



**MUH. SAIFUL**

**E1E1 15 075**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALU OLEO**

**KENDARI**

**2019**

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Salah satu pemanfaatan teknologi informasi dalam bidang pemerintahan adalah *Electronic Government (e-Government)*. *E-Government* mengacu pada penggunaan teknologi informasi oleh instansi pemerintah dalam pelayanan administrasi kepada masyarakat. Teknologi ini juga dapat melayani berbagai tujuan yang berbeda seperti pelayanan pemerintah kepada masyarakat, meningkatkan interaksi dengan bisnis dan industri, memperkuat akuntabilitas, dan pemberdayaan warga melalui akses informasi atau manajemen pemerintah yang lebih efisien (Bhatnagar dan Deane, 2004).

Kepolisian Negara Republik Indonesia Daerah (POLDA) Sulawesi Tenggara, yang dikelola oleh Div PROPAM (Divisi Profesi dan Pengamanan) telah mengembangkan situs pengaduan/pelaporan *online* yang bertujuan untuk meningkatkan partisipasi masyarakat dalam melakukan pengawasan polisi dan kinerja kepolisian Sulawesi Tenggara serta menerima berbagai masukan dari masyarakat dalam upaya membina dan menyelenggarakan fungsi pertanggung jawaban profesi dan pengamanan internal termasuk penegakan disiplin dan ketertiban di lingkungan Kepolisian Negara Republik Indonesia khususnya wilayah Sulawesi Tenggara.

Dalam menganalisa jumlah data laporan yang ada di situs Pengaduan Propam Polda Sultra, pemanfaatan metode klasifikasi dapat membantu mengidentifikasi laporan yang masuk secara langsung dari masyarakat berdasarkan jenis kata laporan yang ada. Dalam kasus data pada situs Propam Polda Sultra yang merupakan data teks, *Text Mining* merupakan jenis metode data mining yang digunakan dalam menganalisa data laporan yang bersifat tidak terstruktur seperti data teks dan dalam jumlah yang sangat besar (Xiang, dkk., 2015). *Text Mining* merupakan salah satu bagian dari data mining dalam menganalisa dan memproses data teks yang bersifat semi terstruktur (*semistructured*) dan tidak (*unstructured*). Hal ini berbeda dengan data mining yang berguna dalam menganalisa data yang

1

bersifat ordinal, kategorikal maupun kontinyu (Hwan, dkk., 2010). Pertama kali dikembangkan pada tahun 1980-an, *Text Mining* telah menjadi semakin efektif seiring dengan daya komputasi yang meningkat. *Text Mining* telah sukses dalam menemukan berbagai pola dan koneksi yang tidak terlihat oleh sekilas mata, seperti mengukur tingkat kebahagiaan melalui kata-kata dalam *twitter*, mengetahui hubungan senyawa tertentu dengan enzim tertentu, dan sebagainya (Belsky, 2012).

Salah satu metode pengorganisasian informasi adalah klasifikasi berdasarkan kategori atau kelas yang telah ditentukan. Salah satu tujuan dari klasifikasi teks atau dokumen adalah menggolongkan atau mengelompokkan suatu dokumen ke dalam suatu kategori tertentu (Manning, dkk., 2008).

Salah satu metode statistik yang dapat diterapkan untuk melakukan klasifikasi adalah *Rocchio*. Keunggulan dari memanfaatkan metode ini yaitu dalam tahapan kerjanya, metode klasifikasi *Rocchio* membandingkan kesamaan isi antara data *training* dan data *test* dengan mempresentasikan semua data kedalam vektor. Dimana hanya dokumen yang relevan yang dipilih, titik kueri bergerak lebih dekat ke arah pusat dokumen yang dianggap relevan. Penelitian untuk klasifikasi teks menggunakan metode *Rocchio* sebelumnya sudah pernah dilakukan. Pada penelitian tersebut menyimpulkan bahwa Klasifikasi Jurnal Berbahasa Inggris Berdasarkan Abstrak dengan Algoritma *Rocchio* dapat diterapkan dengan baik dengan hasil akurasi 93% (Hasugian, 2017). Pada penelitian yang lain juga dilakukan oleh Lumbanraja (2013) menyimpulkan bahwa Klasifikasi Dokumen Teks Skripsi Menggunakan Metode *Rocchio* terbukti dapat digunakan secara efektif untuk melakukan klasifikasi dokumen teks Bahasa Indonesia. Metode *Rocchio* berhasil mengklasifikasikan 76,67% dokumen terkelompokan secara benar sesuai dengan bidang keilmuwan.

Melihat permasalahan tersebut, maka peneliti mengembangkan sebuah sistem yang mampu mengklasifikasikan laporan-laporan pengaduan secara otomatis. Atas dasar inilah peneliti mengangkat judul ***“*Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra*”*.**

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam penelitian ini adalah bagaimana mengklasifikasikan laporan pengaduan di laman Propam Polda Sultra secara otomatis dengan *text mining* menggunakan metode *Rocchio*.

## 1.3 Batasan Masalah

Adapun hal-hal yang menjadi batasan masalah pada penelitian ini adalah sebagai berikut:

1. Data yang digunakan adalah laporan pengaduan yang terdapat pada Propam Polda Sulawesi Tenggara.
2. Data yang digunakan adalah data tahun 2017.
3. Laporan yang digunakan dalam penelitian ini hanya laporan berbahasa Indonesia yang tidak dieja.
4. Kata dasar yang dihitung hanya kata-kata yang terdapat pada Kamus Besar Bahasa Indonesia.
5. Program ini tidak dapat mengatasi kesalahan penulisan yang dilakukan oleh penulis.

## 1.4 Tujuan Penelitian

Tujuan penelitian dalam penelitian Tugas Akhir ini adalah sebagai berikut:

1. Mengetahui hasil klasifikasi laporan pengaduan di laman Propam Polda Sultra menggunakan metode *Rocchio*.
2. Mengetahui akurasi klasifikasi laporan pengaduan di laman Propam Polda Sultra menggunakan metode *Rocchio*.

## 1.5 Manfaat Penelitan

Manfaat yang diharapkan dari penelitian ini adalah dapat memberikan kemudahan dan efisiensi waktu bagi pihak Propam Polda Sultra dalam menangani laporan-laporan yang masuk dari masyarakat dan menindak lanjutinya sesuai dengan pelanggaran yang dilakukan oleh oknum polisi yang terlapor.

## 1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir digunakan untuk menjelaskan penulisan perbab sebagai berikut:

|  |  |
| --- | --- |
| **BAB I** | **PENDAHULUAN**  Bab ini menguraikan tentang latar belakang masalah, rumusan masalah, batasan masalah yang menjelaskan tentang batasan-batasan dari sistem yang dibuat agar tidak menyimpang dari ketentuan yang telah ditetapkan. Selain itu, juga terdapat tujuan dan manfaat penelitian Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra dan sistematika penulisan serta tinjauan pustaka. |
| **BAB II** | **LANDASAN TEORI**  Bab ini menjelaskan tentang teori-teori yang berkaitan dalam pembuatan sistem klasifikasi jenis laporan, yaitu penjelasan tentang  *E-Government*, PROPAM, *TF-IDF*, *Text Mining*, *Rocchio, RUP*, *Flowchart, UML , XAMPP* dan *Database*. |
| **BAB III** | **METODOLOGI PENELITIAN** |

Bagian ini menyajikan langkah-langkah pengumpulan data, prosedur pengembangan perangkat lunak yang dilakukan dalam penelitian menggunakan bentuk kalimat pasif yang meliputi prosedur dan pengumpulan data, prosedur pengembangan perangkat lunak, dan waktu serta tempat penelitian.

## BAB IV ANALISIS DAN PERANCANGAN SISTEM

Pada bagian ini diuraikan analisis sistem yang akan dibuat dan kebutuhan sistem yang meliputi kebutuhan fungsional, kebutuhan non fungsional sistem. Rancangan sistem meliputi rancangan arsitektur sistem, rancangan proses, rancangan prosedural, rancangan data, dan rancangan *user* *interface*.

## BAB V IMPLEMENTASI SISTEM

Bagian ini menyajikan secara lengkap setiap langkah yang dilakukan dalam implementasi sistem, analisis hasil dan pembahasan hasil penelitian.

## BAB VI PENUTUP

Bagian ini berisi tentang kesimpulan dan saran-saran mengenai tugas akhir implementasi metode *Rocchio* untuk klasifikasi jenis pelaporan di laman Propam Polda Sultra.

### 1.7 Tinjauan Pustaka

Penelitian yang dilakukan oleh Apriliana (2017) menyimpulkan bahwa Algoritma *Naïve Bayes Classifier* memiliki kinerja yang baik untuk klasifikasi skripsi berdasarkan abstrak. Hal ini dibuktikan pada pengujian menggunakan data skripsi yang diambil dari perpustakaan prodi teknik informatika, kemudian skripsi diklasifikasikan pada tiga kategori yaitu rekayasa perangkat lunak, komputasi berbasis jaringan, dan komputasi cerdas visualisasi. Hasil klasifikasi menggunakan 51 skripsi didapatkan akurasi 94,11%.

Penelitian yang dilakukan oleh Rezki Fidmasari (2017) menyimpulkan bahwa Metode *Term Frequency - Inverse Document Frequency* mampu melakukan proses analisis keterkaitan berita berdasarkan tingkat kemiripannya secara otomatis, dapat memberikan kemudahan dan efisiensi waktu bagi para pembaca berita dalam menemukan berita lain yang saling berkaitan dan memudahkan untuk menyediakan berita tarkait tanpa melakukan proses *tagging* pada saat memasukkan berita.

Penelitian yang dilakukan oleh Muh. Arzanul Manhar (2013) menyimpulkan bahwa Implementasi *Text Mining* dalam Pembobotan Dokumen Berbahasa Indonesia Menggunakan Metode *Term Frequency / Inverse Document Frequency* mampu mengukur tingkat kemiripan dokumen terhadap dokumen lain dengan baik dengan tingkat akurasi diatas 98% dan mampu mendeteksi adanya aksi plagiat.

Dari beberapa penelitian tersebut, peneliti ingin membuat sistem pelaporan yang mampu mengklasifikasian laporan masuk secara otomatis. Sistem ini menggunakan metode pembobotan *TFIDF* dan metode klasifikasi *Rocchio* untuk mengklasifikasikan laporan. Sistem ini memiliki kelebihan dimana sebelum laporan diklasifikasikan kata yang ada di laporan akan dihitung bobot katanya sehingga menghasilkan tingkat akurasi yang lebih baik. Sedangkan pada penelitianpenelitian sebelumnya hanya menggunakan satu metode saja dalam prosesnya.

**BAB II**

**LANDASAN TEORI**

### 2.1 *E-Government*

*E-Government* merupakan kependekan dari elektronik pemerintah. E-Government biasa dikenal *e-gov*, pemerintah digital, *online* pemerintah atau pemerintah *transformasi*. *E-Government* adalah Suatu upaya untuk mengembangkan penyelenggaraan kepemerintahan yang berbasis elektronik. Suatu penataan sistem manajemen dan proses kerja di lingkungan pemerintah dengan mengoptimalkan pemanfaatan teknologi informasi dan komunikasi.

Menurut Rahardjo (2001), E-Government adalah penggunaan teknologi informasi yang dapat meningkatkan hubungan antara Pemerintah dan pihak-pihak lain. Penggunaan teknologi informasi ini kemudian menghasilkan hubungan bentuk baru seperti: *G2C* *(Government to Citizen)*, *G2B (Government to Business Enterprises)*, dan *G2G (inter-agency relationship)*.

### 2.2 PROPAM

PROPAMadalah singkatan dari Profesi dan Pengamanan yang dipakai oleh organisasi POLRI pada salah satu struktur organisasinya sejak 27 Oktober 2002 (Kep KAPOLRI Nomor : Kep/54/X/2002), sebelumnya dikenal Dinas Provos atau Satuan Provos POLRI yang organisasinya masih bersatu dengan TNI/Militer sebagai ABRI, dimana Provost POLRI merupakan satuan fungsi pembinaan dari Polisi Organisasi Militer / POM atau istilah Polisi Militer / PM.

PROPAM adalah salah satu wadah organisasi POLRI berbentuk Divisi yang bertanggung-jawab kepada masalah pembinaan profesi dan pengamanan dilingkungan internal organisasi POLRI disingkat Divisi Propam Polri sebagai salah satu unsur pelaksana staf khusus POLRI di tingkat Markas Besar yang berada di bawah KAPOLRI.

Tugas PROPAM secara umum adalah membina dan menyelenggarakan fungsi pertanggung jawaban profesi dan pengamanan internal termasuk penegakan

7

disiplin dan ketertiban di lingkungan POLRI dan pelayanan pengaduan masyarakat tentang adanya penyimpangan tindakan anggota/PNS POLRI, yang dalam struktur organisasi dan tata cara kerjanya PROPAM terdiri dari 3 (tiga) bidang/wadah fungsi dalam bentuk sub organisasi disebut Biro (Biro Paminal, Biro Wabprof dan Biro Provos) :

1. Fungsi Pengamanan dilingkungan internal organisasi POLRI dipertanggungjawabkan kepada Biro Paminal.
2. Fungsi pertanggung-jawaban profesi dipertanggungjawabkan kepada Biro Wabprof.
3. Fungsi Provos dalam penegakan disiplin dan ketertiban dilingkungan POLRI dipertanggungjawabkan kepada Biro Provos.

### 2.3 Laman *pengaduan-propam-poldasultra.com*

PROPAM Polda Sultra adalah salah satu organisasi POLRI berbentuk divisi yang bertanggung jawab kepada masalah pembinaan profesi dan pengamanan di lingkungan internal organisasi POLRI terkhusus Sulawesi Tenggara. PROPAM Polda Sultra bertugas untuk membina, penegakan disiplin dan ketertiban di lingkungan POLRI dan melayani penggaduan masyarakat tentang adanya penyimpangan anggota/PNS POLRI di daerah Sulawesi Tenggara.

Laman pengaduan-propam-poldasultra.com adalah laman *website* yang dikelola dan dikembangkan oleh PROPAM Polda Sulawesi Tenggara yang berfungsi untuk meningkatkan layanan kepolisian dengan mempermudah masyarakat dalam hal pelayanan pengaduan atau laporan secara *online*. Laman ini telah terintegrasi sesuai standar prosedur yang berlaku di PROPAM Polda Sultra. Laporan dari masyarakat yang masuk terlebih dahulu akan diverifikasi sebelum ditindak lanjuti oleh pihak PROPAM.

### 2.4 *Text Mining*

*Text mining* yang juga disebut sebagai *Teks Data Mining (TDM)* atau *Knowledge Discovery in Text (KDT*), secara umum mengacu pada proses ekstraksi informasi dari dokumen-dokumen teks tak terstruktur *(unstructured). Text mining* dapat didefinisikan sebagai penemuan informasi baru dan tidak diketahui sebelumnya oleh komputer, yang secara otomatis mengekstrak informasi dari sumber-sumber teks tak terstruktur yang berbeda. Kunci dari proses ini adalah menggabungkan informasi yang berhasil diekstraksi dari berbagai sumber.

Tujuan utama *text mining* adalah mendukung proses *knowledge discovery* pada koleksi dokumen yang besar. Pada prinsipnya, *text mining* adalah bidang ilmu *multidisipliner*, melibatkan *information retrieval (IR)*, *text analysis*, *information extraction (IE)*, *clustering*, *categorization*, *visualization*, *database technology*, *natural language processing (NLP)*, *machinelearning*, dan *data mining*. Dapat pula dikatakan bahwa *text mining* merupakan salah satu bentuk aplikasi kecerdasan buatan *(artificial intelligence / AI)*.

*Text mining* mencoba memecahkan masalah *information overload* dengan menggunakan teknik-teknik dari bidang ilmu yang terkait. *Text mining* dapat dipandang sebagai suatu perluasan dari *data mining* atau *knowledge-discovery in database (KDD)*, yang mencoba untuk menemukan pola-pola menarik dari basis data berskala besar. Namun *text mining* memiliki potensi komersil yang lebih tinggi dibandingkan dengan *data mining*, karena kebanyakan format alami dari penyimpanan informasi adalah berupa teks. *Text mining* menggunakan informasi teks tak terstruktur dan mengujinya dalam upaya mengungkap struktur dan arti yang tersembunyi di dalam teks.

Perbedaan mendasar antara *text mining* dan *data mining* terletak pada sumber data yang digunakan. Pada *data mining*, pola-pola diekstrak dari basis data yang terstruktur, sedangkan di *text mining*, pola-pola diekstrak dari data tekstual (*natural language*). Secara umum, basis data didesain untuk program dengan tujuan melakukan pemrosesan secara otomatis, sedangkan teks ditulis untuk dibaca langsung oleh manusia (Eko, 2011).

*Text mining* (penambangan teks) adalah penambangan yang dilakukan oleh komputer untuk mendapatkan sesuatu yang baru, sesuatu yang tidak diketahui sebelumnya atau menemukan kembali informasi yang tersirat secara *implisit*, yang berasal dari informasi yang diekstrak secara otomatis dari sumber-sumber data teks yang berbeda-beda (Feldman & Sanger, 2007). *Text mining* merupakan teknik yang digunakan untuk menangani masalah klasifikasi, *clustering*, *information extraction* dan *information retrival* (Berry & Kogan, 2010)*.*

Pada dasarnya proses kerja dari *text mining* banyak mengapdopsi dari penelitian *Data Mining* namun yang menjadi perbedaan adalah pola yang digunakan oleh *text mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur sedangkan dalam *Data Mining* pola yang diambil dari *database* yang terstruktur (Han & Kamber, 2006).

#### 2.4.1 Ruang Lingkup Text mining

*Text mining* merupakan suatu proses yang melibatkan beberapa area teknologi. Namun secara umum proses-proses pada *text mining* mengadopsi proses *data mining*. Bahkan beberapa teknik dalam proses *text mining* juga menggunakan teknik-teknik *data mining* (Eko, 2011).

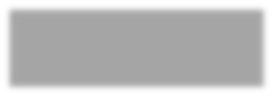
Ada empat tahap proses pokok dalam *text mining*, yaitu pemrosesan awal terhadap teks (*text preprocessing*), transformasi teks (*text transformation*), pemilihan fitur (*feature selection*), dan penemuan pola (*pattern* discovery) (Eko, 2011).

#### 2.4.2 Text Preprocessing

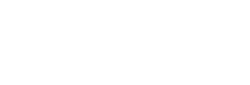
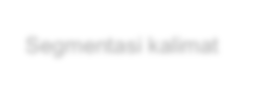
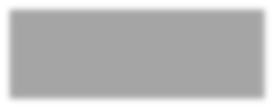
Tahap *text preprocessing* adalah tahap awal dari *text mining*. Tahap ini mencakup semua rutinitas, dan proses untuk mempersiapkan data yang akan digunakan pada operasi *knowledge discovery* sistem *text mining* (Feldman & Sanger, 2007). Tindakan yang dilakukan pada tahap ini adalah *toLowerCase*, yaitu mengubah semua karakter huruf menjadi huruf kecil dan *Tokenizing* yaitu proses penguraian deskripsi yang semula berupa kalimat-kalimat menjadi kata-kata dan menghilangkan *delimiter-delimiter* seperti tanda titik (.), koma (,), spasi dan karakter angka yang ada pada kata tersebut (Weiss, dkk., 2005).

Tahap ini melakukan analisis semantik (kebenaran arti) dan sintaktik (kebenaran susunan) terhadap teks. Tujuan dari pemrosesan awal adalah untuk mempersiapkan teks menjadi data yang akan mengalami pengolahan lebih lanjut. Operasi yang dapat dilakukan pada tahap ini meliputi *part-of-speech (POS) tagging*, menghasilkan *parse tree* untuk tiap-tiap kalimat, dan pembersihan teks.

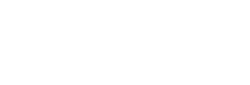
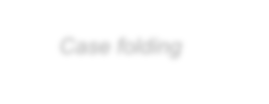
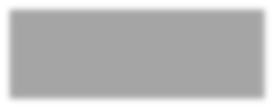
*Text preprocessing* adalah suatu tahap untuk mengolah teks berita yang merupakan bahan mentah menjadi kata-kata yang telah siap dihitung bobot katanya. Beberapa proses dari *text preprocessing*, yaitu segmentasi kalimat, *case folding, tokenizing, filtering,* dan *stemming.* Berikut adalah diagram alirnya terlihat pada gambar 2.1.



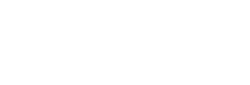
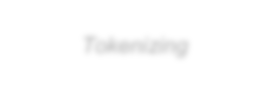
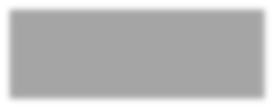
*Stemming*



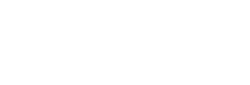
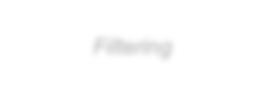
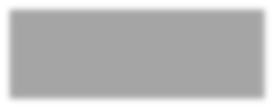
Segmentasi kalimat



*Case folding*



*Tokenizing*



*Filtering*

**Gambar 2.1 Proses *Text Preprocessing***

1. Segmentasi kalimat

Segmentasi kalimat merupakan langkah awal dari proses *text preprocessing.* Dalam proses ini, teks berita yang terdiri dari paragraf yang dipecah menjadi beberapa kalimat. Pemisahan setiap kalimat berdasarkan tanda baca, seperti tanda titik (.), tanda seru (!) dan tanda tanya (?). Contoh pemecahan dokumen menjadi kalimat pada Tabel 2.1.

**Tabel 2.1 Proses Segmentasi**

|  |  |
| --- | --- |
| **Dokumen** | **Hasil segmentasi kalimat** |
| Banjir pada musim hujan. Banjir mengancam kota Surabaya. Pengerukan serta peleburan saluran belum dilakukan secara maksimal. Sekitar 3 dari 5 rumah warga terkena banjir. | * Banjir pada musim hujan. * Banjir mengancam kota Surabaya. - Pengerukan serta peleburan saluran belum dilakukan secara maksimal. * Sekitar 3 dari 5 rumah warga terkena banjir. |

1. *Case Folding*

Paragraf berita yang telah dipotong menjadi kalimat akan menjalankan proses *case folding. Case folding* adalah proses mengubah semua teks menjadi karakter dengan huruf kecil dan membuang semua karakter selain a-z. Jika terdapat tanda baca, bilangan numerik dan simbol akan dihilangkan. Contohnya terlihat pada Tabel 2.2.

**Tabel 2.2 Proses *Case Folding***

|  |  |
| --- | --- |
| **Kalimat** | **Hasil *case folding*** |
| Banjir pada musim hujan. Banjir mengancam Sebagian kota di  Indonesia. Pengerukan serta peleburan saluran belum dilakukan secara maksimal. Sekitar 3 dari 5 rumah warga terkena banjir. | * banjir pada musim hujan * banjir mengancam sebagian kota di indonesia * pengerukan serta peleburan saluran belum dilakukan secara maksimal. * sekitar dari rumah warga terkena banjir |

1. *Tokenizing*

Suatu proses untuk mengubah bentuk kalimat menjadi kata-kata tunggal. Pemotongan kalimat berdasarkan *delimiter* yang menyusunnya. *Tokenization* dapat dilakukan dengan menghilangkan tanda baca dan memisahkannya per spasi. Tahapan ini juga menghilangkan karakterkarakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*). Proses ini bertujuan agar nantinya dapat melakukan proses *stemming*. Contoh hasil *tokenizing* kata terlihat pada Tabel 2.3. **Tabel 2.3 Proses *Tokenizing* Kata**

|  |  |
| --- | --- |
| **Hasil filtering** | **Hasil *tokenizing* kata** |
| * banjir pada musim hujan. * banjir mengancam sebagian kota di Indonesia. * pengerukan serta peleburan saluran belum dilakukan secara maksimal. * sekitar dari rumah warga terkena banjir. | * banjir * pada * musim * hujan * mengancam * sebagian * kota * di * indonesia * pengerukan * serta * peleburan * saluran * belum * dilakukan * secara * maksimal * sekitar * dari * rumah * terkena |

1. *Filtering*

Pada proses *filtering* dilakukan pembuangan *stopword*. *Stopword* adalah kata-kata yang tidak memiliki makna atau kata yang kurang berarti dan sering muncul dalam kumpulan kata-kata. Proses pembuangan kata- kata yang tidak penting dengan mengecek pada kamus *stopword*. Jika kata ada yang sama dengan kata pada kamus *stopword¸* maka kata tersebut akan dibuang atau dihapus. Seandainya *stopword* tidak dihilangkan maka *stopword* akan memiliki bobot yang besar karena sering muncul dalam teks, sehingga akan mempengaruhi hasil ringkasan karena kalimat yang terpilih adalah kalimat yang memiliki bobot paling besar. *Stopword* dapat berupa kata penghubung, kata ganti, preposisi, dll , seperti: dia, antara, akan, demi, karena, atau, bahwa, bila, juga, kalau, hingga, bagi, kecuali, oleh, dan lainlain. Contoh *filtering* terlihat pada Tabel 2.4.

**Tabel 2.4 Proses *Filtering***

|  |  |
| --- | --- |
| **Kalimat** | **Hasil *filtering*** |
| * banjir pada musim hujan. * banjir mengancam sebagian kota di Indonesia. * pengerukan serta peleburan saluran belum dilakukan secara maksimal. * sekitar dari rumah warga terkena banjir. | * banjir * musim * hujan * mengancam * kota * indonesia * pengerukan * peleburan * saluran * maksimal * rumah * terkena |

1. *Stemming*

Proses selanjutnya *stemming*, yaitu mengembalikan suatu kata ke bentuk akarnya (*root word*) dengan aturan-aturan tertentu, sehingga setiap kata memiliki representasi yang sama. *Stemming* dalam penelitian ini menggunakan algoritma Nazief & Adriani (1996). Contoh *stemming* terlihat pada Tabel 2.5.

**Tabel 2.5 Hasil *Stemming***

|  |  |  |  |
| --- | --- | --- | --- |
| **Hasil *tokenizing* kata** | | **Hasil *stemming*** | |
| banjir  musim  hujan mengancam kota indonesia | pengerukan peleburan saluran maksimal rumah terkena | banjir  musim hujan ancam kota indonesia | keruk lebur  salur maksimal rumah terkena |

*Stemming* adalah proses pemetaan variansi morfologikal kata dalam kata dasar atau kata umumnya (*stem*) (Adhitia, 2009). Misalnya kata "perancangan" dan "merancang" akan diubah menjadi sebuah kata yang sama, yaitu "rancang". Proses *stemming* sangat tergantung kepada bahasa dari kata yang akan di-*stem*. Hal ini dikarenakan, dalam melakukan proses *stemming* harus mengaplikasikan aturan morfologikal dari suatu bahasa.

Kebanyakan bahasa memiliki kata fungsi dan kata sambung seperti artikel dan preposisi yang hampir selalu muncul pada dokumen teks. Biasanya kata-kata ini tidak memiliki arti yang lebih di dalam memenuhi kebutuhan seorang pencari di dalam mencari informasi. Kata-kata tersebut (misalnya *a*, *an*, *the*, *on* pada bahasa Inggris) disebut sebagai *Stopwords* (Chakrabarti, dkk., 2003). Pembuangan Stopwords dapat mengurangi besar dari *index space* dan meningkatkan performa dalam pemrosesan lebih lanjut.

Aturan imbuhan yang digunakan pada Bahasa Indonesia lebih kompleks, tidak seperti aturan imbuhan Bahasa Inggris. Pada Bahasa Indonesia terdapat aturan imbuhan yang lebih kompleks yang meliputi awalan, akhiran, sisipan, dan konfiks (kombinasi dari awalan dan akhiran) (Adriani, dkk., 2007).

Banyak penelitian yang dilakukan untuk menemukan algoritma *stemming* yang tepat dan bagus dalam Bahasa Indonesia, antara lain algoritma Nazief & Adriani. Algoritma Nazief & Adriani menyimpulkan sebuah kata dasar dapat ditambahkan imbuhan berupa *derivation prefix* (DP) di awal dan/atau diakhiri secara berurutan oleh *derivation suffix* (DS), *possesive pronoun* (PP), dan *particle* (P) yang masing-masing bersifat *optional*. Keterangan diatas dirumuskan pada Gambar 2.2.

DP + DP + DP + root word + DS + PP + P



## Gambar 2.2 Format Kata Berimbuhan dalam Bahasa Indonesia

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Cari kata yang akan di-*stem* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tersebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang.

Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“- ku”, “-mu”, atau “-nya”), jika ada.

1. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a.
   1. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   2. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
2. Hapus *Derivation Prefix*. Jika pada langkah 3 ada *sufiks* yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   1. Periksa Tabel kombinasi awalan-akhiran yang tidak diijinkan pada Tabel 2.6. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   2. *For* i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
3. Melakukan *Recoding*.
4. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me- ”, atau “pe-” maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “*none*” maka awalan dapat dilihat pada Tabel 2.8. Hapus awalan jika ditemukan.

## Tabel 2.6 Kombinasi Awalan Akhiran Yang Tidak Diizinkan

|  |  |
| --- | --- |
| **Awalan** | **Akhiran yang tidak diizinkan** |
| be- | -i |
| di- | -an |
| ke- | -i, -kan |
| me- | -an |
| se- | -i, -kan |

**Tabel 2.7 Aturan Pemenggalan Awalan *Stemmer* Nazief & Adriani**

|  |  |  |
| --- | --- | --- |
| **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 | Belajar… | bel-ajar |
| 5 | beC l erC2… | be-C lerC2… dimana C1!={‘r’|’I’} |
| 6 | terV… | Ter-V… | te-rV… |
| 7 | terCerV… | Ter-CerV… dimana C!=’r’ |
| 8 | terCP… | Ter-CP… dimana C!=‟r‟ dan P!=‟er‟ |
| 9 | teC1erC2... | te-C1erC2... dimana C1!=‟r‟ |
| 10 | me{l|r|w|y}V... | me-{l|r|w|y}V... |
| 11 | mem{b|f|v}... | mem-{b|f|v}... |
| 12 | mempe{r|l}... | mem-pe... |
| 13 | mem{rV|V}... | me-m{rV|V}... | me-p{rV|V}... |
| 14 | men{c|d|j|z}... | men-{c|d|j|z}... |
| 15 | menV... | me-nV... | me-tV |
| 16 | meng{g|h|q}... | meng-{g|h|q}... |
| 17 | mengV... | meng-V... | meng-kV... |
| 18 | menyV... | meny-sV… |
| 19 | mempV... | mem-pV... dimana V!=„e‟ |
| 20 | pe{w|y}V... | pe-{w|y}V... |
| 21 | perV... | per-V... | pe-rV... |

**Tabel 2.7 Aturan Pemenggalan Awalan *Stemmer* Nazief & Adriani (Lanjutan)**

|  |  |  |
| --- | --- | --- |
| **Aturan** | **Format Kata** | **Pemenggalan** |
| 23 | perCAP | per-CAP... dimana C!=‟r‟ dan P!=‟er‟ |
| 24 | perCAerV... | per-CAerV... dimana C!=‟r‟ |
| 25 | pem{b|f|V}... | pem-{b|f|V}... |
| 26 | pem{rV|V}... | pe-m{rV|V}... | pe-p{rV|V}... |
| 27 | pen{c|d|j|z}... | pen-{c|d|j|z}... |
| 28 | penV... | pe-nV... | pe-tV... |
| 29 | peng{g|h|q}... | peng-{g|h|q}... |
| 30 | pengV... | peng-V... | peng-kV... |
| 31 | penyV... | peny-sV… |
| 32 | pelV... | pe-lV... kecuali “pelajar” yang  menghasilkan “ajar” |
| 33 | peCerV... | per-erV... dimana C!={r|w|y|l|m|n} |
| 34 | peCP... | pe-CP... dimana C!={r|w|y|l|m|n} dan  P!=‟er‟ |

Keterangan simbol huruf:

C : huruf konsonan

V : huruf vokal

A : huruf vokal atau konsonan

P : partikel atau fragmen dari suatu kata, misalnya “er”

## Tabel 2.8 Jenis Awalan Berdasarkan Tipe Awalannya

|  |  |
| --- | --- |
| **Tipe awalan** | **Awalan yang harus dihapus** |
| di- | -di |
| ke- | ke- |
| se- | se- |
| te- | te- |
| ter- | ter- |
| ter-luluh | Ter |

Untuk mengatasi keterbatasan pada algoritma di atas, maka ditambahkan aturan-aturan dibawah ini:

1. Aturan untuk reduplikasi.

1. Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka *root word* adalah bentuk tunggalnya, contoh: “buku-buku” *root word*-nya adalah “buku”.
2. Kata lain, misalnya “bolak-balik”, “berbalas-balasan”, dan ”seolah-olah”. Untuk mendapatkan *root word*-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki *root word* yang sama maka diubah menjadi bentuk tunggal, contoh: kata “berbalasbalasan”, “berbalas” dan “balasan” memiliki *root word* yang sama yaitu “balas”, maka *root word* “berbalas-balasan” adalah “balas”.

Sebaliknya, pada kata “bolak-balik”, “bolak” dan “balik” memiliki *root word* yang berbeda, maka *root word*-nya adalah “bolak-balik” 2. Tambahan bentuk awalan dan akhiran serta aturannya.

1. Untuk tipe awalan “mem-“, kata yang diawali dengan awalan

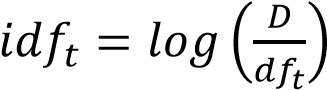
“memp-” memiliki tipe awalan “mem-”.

1. Tipe awalan “meng-“, kata yang diawali dengan awalan “mengk- ” memiliki tipe awalan “meng-”.

### 2.4 Metode *Term Frequency-Inverse Document Frequency* (TF-IDF)

Metode *TF-IDF* (Robertson, 2004) merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebut tinggi di dalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen.

Metode *TF- IDF* merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat. Metode ini akan menghitung nilai *Term Frequency* *(TF)* dan *Inverse Document Frequency (IDF)* pada setiap *token* (kata) di setiap dokumen dalam korpus. Rumus untuk *TF-IDF*:

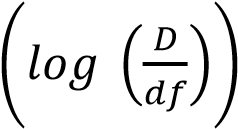
…………………….………………...…(2.1)

𝑊𝑑,𝑡 = 𝑡𝑓𝑑,𝑡 𝑥 𝑖𝑑𝑓𝑑,𝑡…………………….………...…......(2.2)

Keterangan:

𝑡𝑓 = banyaknya kata yang dicari pada sebuah dokumen.

*D* = total dokumen.

𝑑𝑓𝑡 = jumlah dokumen yang mengandung *term* t. 𝑖𝑑𝑓= *Inversed Document Frequency* *.*

*d* = dokumen ke-d. *t =* kata ke-t dari kata kunci.

*W* = bobot dokumen ke-d terhadap kata ke-t.

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses pengurutan dimana semakin besar nilai W, semakin besar tingkat similaritas dokumen tersebut terhadap kata kunci, demikian sebaliknya. Untuk menerapkan perhitungan algoritma *term frequency - inverse document frequency* dengan kalimat yang lain dapat dilihat pada studi kasus:

Dokumen 1 (D1) = Manajemen Keuangan.

Dokumen 2 (D2) = Pengetahuan nilai tukar rupiah.

Dokumen 3 (D3) = Dalam manajemen keuangan pengetahuan tentang manajemen pengeluaran penting.

Maka, jumlah dokumen (D) = 3 dan perhitungan untuk *term frequency - inverse document frequency* dapat dilihat pada Tabel 2.9.

**Tabel 2.9 Hasil Penghitungan Bobot Masing-Masing Kata**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Term |  | Tf | | Df | D/df | IDF Log(D/df) |  | W |  |
| D1 | D2 | D3 | D1 | D2 | D3 |
| manajemen | 1 | 0 | 2 | 2 | 1,5 | 0.176 | 0.176 | 0 | 0.352 |
| keuangan | 1 | 0 | 1 | 2 | 1,5 | 0.176 | 0.176 | 0 | 0.176 |
| pengetahuan | 0 | 1 | 1 | 2 | 1,5 | 0.176 | 0 | 0.176 | 0.176 |
| nilai | 0 | 1 | 0 | 1 | 3 | 0.477 | 0 | 0.477 | 0 |
| tukar | 0 | 1 | 0 | 1 | 3 | 0.477 | 0 | 0.477 | 0 |
| rupiah | 0 | 1 | 0 | 1 | 3 | 0.477 | 0 | 0.477 | 0 |
| pengeluaran | 0 | 0 | 1 | 1 | 3 | 0.477 | 0 | 0 | 0.477 |
| penting | 0 | 0 | 1 | 1 | 3 | 0.477 | 0 | 0 | 0.477 |
|  |  | Total | |  |  |  | 0.352 | 1,607 | 1,658 |

Keterangan:

Term : kata yang akan dihitung bobotnya.

Tf : jumlah kata muncul dalam satu kalimat/dokumen.

Df : jumlah dokumen yang berisi kata tersebut.

Idf : kemunculan kata pada kumpulan kalimat/dokumen.

W : bobot kata.

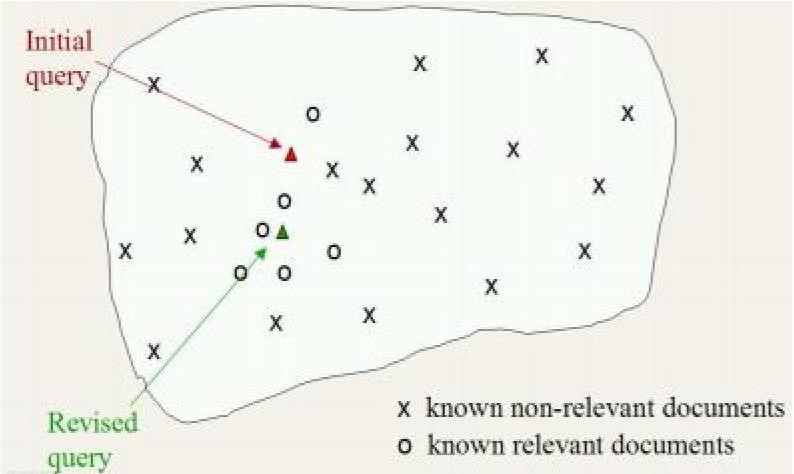
D : dokumen atau kalimat (i=1,2,3).

### 2.5 Metode *Rocchio*

*Rocchio* merupakan suatu metode pembelajaran *supervised document classification*. Metode klasifikasi *Rocchio* membandingkan kesamaan isi antara data *training* dan data *test* dengan mempresentasikan semua data kedalam vektor. Setiap bobot kata merupakan dimensi dalam ruang vektor. Kedekatan kesamaan dihitung dari kedekatan sudut yang terbentuk antara bobot data *training* dan bobot data *test*. Untuk menghitung bobot setiap kata dalam dokumen digunakan skema pembobotan *TFIDF.* Karena komponen *heuristic*/utama klasifikasi *Rocchio* adalah skema pembobotan *TFIDF,* metode pembelajaran *Rocchio* disebut juga dengan *TFIDF Classifer* (Joachihms, 1997).

Dalam membandingkan kesamaan isi antara data *training* dan data *test*, *Rocchio* menggunakan *prototype* *vector* untuk merepresentasikan kategori yang terbentuk dari data *training*, dengan kata lain *prototype vector* merupakan vektor yang mewakili seluruh vektor data *training* dalam setiap kategori. Tiga hal utama yang dapat dipakai pada klasifikasi *Rocchio* adalah menggunakan skema pembobotan *tf-idf* yang berguna untuk merepresentasikan dokumen dalam sebuah vektor, merepresentasikan *prototype* setiap kategori dengan menjumlahkan vektorvektor dalam satu kategori dari data *training*, membandingkan kedekatan sudut antara vektor data *test* dengan semua *prototype vector* (Selvi, dkk., 2016).

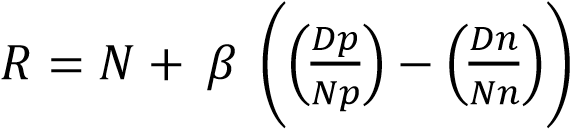
Ketika jawaban kueri dipresentasikan, pengguna akan memilih beberapa dokumen yang relevan dan sebuah himpunan dokumen yang tidak relevan, keduanya direpresentasikan sebagai vektor di dalam ruang (Khan, dkk., 2010). Efek dari umpan balik relevansi diilustrasikan seperti gambar 2.3:



**Gambar 2.3.  *Rocchio Algorithm* (Manning, dkk., 2008).**

Dimana hanya dokumen yang relevan yang dipilih, titik kueri bergerak lebih dekat ke arah pusat dokumen yang dianggap relevan.

Algoritma *Rocchio* mengklasifikasikan data dengan cara melihat keberadaan indeks dari dokumen tersebut. Indeks dokumen tersebut terbagi menjadi 2 yaitu dokumen yang relevan dan tak relevan. Algoritma *Rocchio* mengukur tingkat kemiripan *term* dengan dengan persamaan (Mark van Uden, 2011):

……….……………..(2.3)

|  |  |  |
| --- | --- | --- |
| Keterangan: | | |
| 𝑅 |  | = tingkat kemiripan *term*. |
| 𝑁 |  | = jumlah *term* tiap dokumen. |
| 𝛽 |  | = nilai bobot *term*. |
| 𝐷𝑝 |  | = *term* dari dokumen relevan. |
| 𝑁𝑝 |  | = jumlah dari dokumen relevan. |
| 𝐷𝑛 |  | = *term* dari dokumen tak relevan. |
| 𝑁𝑛 |  | = jumlah dari dokumen tak relevan. |

Contoh perhitungan *Rocchio* sebagai berikut:

Input Pelapor : Saya telah dipukuli dan ditendang pada saat saya melapor

|  |  |
| --- | --- |
| Kategori 1 :  (Penganiayaan) | saya telah dianiaya dipukuli dan ditendang hingga lebam pada saat kasus penangkapan |
| Kategori 2 :  (Pengancaman) | telah melakukan pengancaman /peneroran apabila memberikan kesaksian tentang kasusnya |
| Kategori 3 :  (Penanganan Perkara) | melakukan penganiayaan terhadap pelapor akan tetapi penyidik mengabaikan kasus tersebut. |
| Kategori 4 :  (Pelayanan Buruk) | tidak menyelesaikan pengaduan masyarakat dan mengabaikan laporan yang masuk |

Berikut tahap-tahap perhitungan metode *rocchio*:

1. Tahap *Preprocessing*

Sebelum melakukan proses pembobotan setiap dokumen harus melalui tahap *text preprocessing* yang terdiri dari *case folding*, *tokenizing*, *filtering dan stemming*. Hasil dari tahap *preprocessing* dapat dilihat pada tabel 2.10, 2.11,

2.12 dan 2.13.

## Tabel 2.10 Tahap *Preprocessing* Input Pelapor

|  |  |  |
| --- | --- | --- |
| Input Pelapor | | Keterangan dipukuli -> pukul  ditendang-> tendang melapor->lapor stopword dihapus |
| Term | Jumlah |
| pukul | 1 |
| tending | 1 |
| lebam | 1 |
| lapor | 1 |

## Tabel 2.11 Tahap *Preprocessing* Kategori 1

|  |  |  |
| --- | --- | --- |
| Dokumen 1 | | Keterangan    dianiaya-> aniaya dipukuli -> pukul  ditendang-> tendang  penangkapan->tangkap stopword dihapus |
| Term | Jumlah |
| aniaya | 1 |
| pukul | 1 |
| tending | 1 |
| lebam | 1 |
| kasus | 1 |
| tangkap | 1 |

## Tabel 2.12 Tahap *Preprocessing* Kategori 2

|  |  |  |
| --- | --- | --- |
| Dokumen 2 | | Keterangan    melakukan -> laku  pengancaman -> ancam peneroran -> teror  memberikan -> beri kesaksian -> saksi stopword dihapus. |
| *Term* | Jumlah |
| laku | 1 |
| ancam | 1 |
| teror | 1 |
| beri | 1 |
| saksi | 1 |
| kasus | 1 |

**Tabel 2.13 Tahap *Preprocessing* Kategori 3**

|  |  |  |
| --- | --- | --- |
| Dokumen 3 | | Keterangan    melakukan -> laku  penganiayaan -> aniaya pelapor -> lapor  penyidik -> sidik  mengabaikan -> abai stopword dihapus |
| *Term* | Jumlah |
| laku | 1 |
| aniaya | 1 |
| lapor | 1 |
| sidik | 1 |
| abai | 1 |
| kasus | 1 |

**Tabel 2.14 Tahap *Preprocessing* Kategori 4**

|  |  |  |
| --- | --- | --- |
| Dokumen 4 | | Keterangan    menyelesaikan -> selesai pengaduan -> adu  mengabaikan -> abai pelapor -> lapor |
| *Term* | Jumlah |
| selesai | 1 |
| adu | 1 |
| masyarakat | 1 |
| abai | 1 |
| lapor | 1 |
| masuk | 1 |

2. Tahap pembobotan menggunakan metode *Term Frequency- Inverse Document Frequency* (*TFIDF*).

Setelah dokumen melewati tahap *preprocessing* dokumen akan melalui tahap pembobotan menggunakan metode *TFIDF*. Berikut tabel pembobotannya:

### Tabel 2.15 Pembobotan *Term Frequency- Inverse Document Frequency (TFIDF)*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Jumlah Kategori : | 4 |  | |  |  |  |  |  |  |  |
| **TERM** |  |  | |  | **df** | **idf** |  |  | **β** |  |
|  |  |  |  |  |  |  |  |
|  | **K1** | **K2** | **K3** | **K4** |  |  | **K1** | **K2** | **K3** | **K4** |
| aniaya | 1 |  | 1 |  | 2 | 0,30103 | 0,3010 |  | 0,3010 |  |
| pukul | 1 |  |  |  | 1 | 0,60206 | 0,6021 |  |  |  |
| tendang | 1 |  |  |  | 1 | 0,60206 | 0,6021 |  |  |  |
| lebam | 1 |  |  |  | 1 | 0,60206 | 0,6021 |  |  |  |
| kasus | 1 | 1 | 1 |  | 3 | 0,124939 | 0,1249 | 0,1249 | 0,1249 |  |
| tangkap | 1 |  |  |  | 1 | 0,60206 | 0,6021 |  |  |  |
| laku |  | 1 | 1 |  | 2 | 0,30103 |  | 0,3010 | 0,3010 |  |
| ancam |  | 1 |  |  | 1 | 0,60206 |  | 0,6021 |  |  |
| teror |  | 1 |  |  | 1 | 0,60206 |  | 0,6021 |  |  |
| beri |  | 1 |  |  | 1 | 0,60206 |  | 0,6021 |  |  |
| saksi |  | 1 |  |  | 1 | 0,60206 |  | 0,6021 |  |  |
| lapor |  |  | 1 | 1 | 2 | 0,30103 |  |  | 0,3010 | 0,3010 |
| sidik |  |  | 1 |  | 1 | 0,60206 |  |  | 0,6021 |  |
| abai |  |  | 1 | 1 | 2 | 0,30103 |  |  | 0,3010 | 0,3010 |
| selesai |  |  |  | 1 | 1 | 0,60206 |  |  |  | 0,6021 |
| adu |  |  |  | 1 | 1 | 0,60206 |  |  |  | 0,6021 |
| masyarakat |  |  |  | 1 | 1 | 0,60206 |  |  |  | 0,6021 |
| masuk |  |  |  | 1 | 1 | 0,60206 |  |  |  | 0,6021 |
|  |  | **Jumlah** | |  |  |  | **2,8342** | **2,8342** | **1,9311** | **3,0103** |

Ket : Kuning : dokumen relevan

Hijau : dokumen tidak relevan

3. Tahap Perhitungan Metode *Rocchio*

Setelah melalui proses pembobotan, dokumen tersebut akan diklasifikasikan dengan metode klasifikasi yaitu menggunakan metode *Rocchio*. Perhitungan dan hasil perhitungan klasifikasi ditampilkan pada tabel

2.15 dan 2.16.

**Tabel 2.16 Perhitungan Metode *Rocchio***

|  |
| --- |
|  |
|  |
|  |

## Tabel 2.17 Hasil Perhitungan Metode Rocchio

|  |  |
| --- | --- |
| Kategori 1 (Penganiayaan) | 7,25 |
| Kategori 2 (Pengancaman) | 5,10 |
| Kategori 3 (Penanganan Perkara) | 5,88 |
| Kategori 4 (Pelayanan Buruk) | 5,81 |

Dari hasil perhitungan di atas didapatkan nilai paling besar yaitu 7,25 yang merupakan data kategori 1 yaitu kategori penganiayaan, maka didapatkan bahwa laporan yang telah diinput diklasifikasikan ke dalam kategori penganiayaan.

### 2.6 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan pada penelitian ini adalah metode *Rational Unified Process* (RUP). RUP merupakan salah satu proses rekayasa perangkat lunak yang menyediakan pendekatan untuk menentukan tugas dan tanggung jawab dalam pengembangan suatu organisasi, tujuannya adalah untuk memastikan produksi kualitas tinggi, *software* memenuhi dengan kebutuhan *user* sesuai dengan jadwal dan biaya yang telah dirancang. Dalam metode RUP ini, terdiri dari 4 tahap, yaitu:

1. *Inception*

Tahap ini membangun *business case* untuk sistem dan membatasi ruang lingkupnya, untuk melakukan hal ini diharuskan untuk mengidentifikasi semua entitas eksternal yang akan berinteraksi dengan sistem, dan mendefinisikan interaksi pada level tertentu. Ini juga termasuk mengidentifikasi semua *use cases* dan menjelaskan beberapa yang signifikan. *Business case* termasuk kriteria keberhasilan, perkiraan resiko, dan mengestimasikan sumber daya yang dibutuhkan.

1. *Elaboration*

Tujuan dari tahap *elaboration* adalah menganalisis domain masalah, membuat sebuah dasar arsitektur, membangun rencana proyek, dan mengeliminasi resiko terbesar dari proyek. Untuk menjalankan objek-objek tersebut diperlukan melihat lebih luas dan lebih dalam terhadap sistem. Pada tahap ini merupakan tahap paling sulit karena pada tahap ini memastikan bahwa arsitektur, kebutuhan, dan perencanaan cukup stabil sehingga waktu dan biaya tidak berubah.

1. *Construction*

Dalam tahap ini semua komponen dan fitur aplikasi yang dibuat dan di integrasikan kedalam *software*. Dalam tahapan ini juga dituntut untuk mengoptimalkan sumber daya, biaya, jadwal dan kualitas. Pada tahapan ini meliputi bagaimana suatu aplikasi biasa diimplementasikan dan diuji coba.

* 1. Implementasi

Penjelasan mengenai perangkat keras dan perangkat lunak apa saja yang dibutuhkan untuk mengimplementasikan sistem.

* 1. *Coding*

Proses pengkodean dilakukan dengan menggunakan bahasa pemrograman. Pengkodean sendiri berisi tahapan-tahapan perhitungan metode.

* 1. *Testing*

Pada tahap ini dilakukan pengujian terhadap aplikasi yang telah dibangun untuk mengetahui tingkat akurasi dan kualitas dari aplikasi tersebut, apakah sudah sesuai dengan yang diharapkan atau tidak. *Testing* dilakukan dengan menguji semua tombol-tombol yang terdapat pada aplikasi apakah sudah berjalan sesuai dengan fungsinya atau tidak.

1. *Transition*

Pada tahap ini dilakukan *testing* akhir pada sistem yang telah jadi, kemudian dilakukan sosialisasi penggunaan perangkat lunak yang telah dibangun ke *administrator*.

### 2.8 *Flowchart*

*Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program (Pahlevy. 2010). *Flowchart* menolong analis dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* terbagi atas lima jenis, yaitu:

1. *Flowchart* Sistem (*System Flowchart*)
2. *Flowchart Paperwork* / *Flowchart* Dokumen (*Document Flowchart*)
3. *Flowchart* Skematik (*Schematic Flowchart*)
4. *Flowchart* Program (*Program Flowchart*)
5. *Flowchart* Proses (*Process Flowchart*)

**Tabel. 2.18 Simbol-Simbol *Flowchart***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO.** | **GAMBAR** | |  | **NAMA** | **KETERANGAN** |
| 1. |  |  |  | Proses | Mempresentasikan operasi. |
|  |
| 2. |  | |  | *Input / Output* | Mempresentasikan *Input* data atau *Output* data yang diproses atau informasi. |
| 3. |  | |  | Keputusan | Keputusan dalam program. |
| 4. |  | |  | Dokumen | Dokument I / O dalam format cetak. |
| 5. |  | |  | *Terminal points* | Awal / akhir *flowchart*. |

**Tabel. 2.18 Simbol-Simbol *Flowchart* (Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 6. |  | *Preparation* | Pemberian harga awal. |
| 7. |  | Manual *input* | *Input* yang dimasukkan secara manual dari keyboard. |
| 8. |  | Penghubung | Keluar atau masuk dari bagian lain *flowchart* khususnya |
| 9. |  | Penghubung | Keluar atau masuknya dari bagian lain *flowchart* khususnya halaman lain. |
| 10. |  | *Display* | *Output* yang ditampilkan pada terminal. |
| 11. |  | Anak panah | Mempresentasikan alur kerja. |

### 2.9 *Unified Modeling Language* (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artefact* dapat berupa model, deskripsi atau perangkat lunak dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan (Rosa dan Shalahuddin, 2013).

#### 2.9.1 Use Case Diagram

*Diagram Use case* adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem secara fungsional yang terlihat *user.* Sebuah *use case* mempresentasikan sebuah interaksi antara aktor dengan sistem.

**Tabel 2.19 Simbol *Use Case Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case.* |
| 2. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*dependent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri  (*independent*). |
| 3. |  | *Generalization* | Hubungan dimana objek anak  (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*). |
| 4. |  | *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit.* |
| 5. |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7. |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8. |  | *Use case* | Deskripsi dari uraian aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor.* |
| 9. |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya  (sinergi). |
| 10. |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi. |

#### 2.9.2 Activity Diagram

*Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* *diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

*Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). *Activity diagram* ini mirip dengan *flowchart diagram*.

**Tabel 2.20 Simbol *Activity Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Activity* | Memperlihatkan bagaimana masingmasing kelas antarmuka saling berinteraksi satu sama lain. |
| 2. |  | *Action* | *State* dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3. |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| 4. |  | *Activity Final Node* | Bagaimana objek dibentuk dan diakhiri. |

#### 2.9.3 Sequence Diagram

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu.

**Tabel 2.21 Simbol S*equence Diagram***

## NO GAMBAR NAMA KETERANGAN

1. *LifeLine* Objek *entity*, antarmuka yang saling berinteraksi.
2. *Message* Komunikasi antar objek yang memuat

informasi-informasi tentang aktifitas yang terjadi.

1. *Message* Spesifikasi dari komunikasi antar

objek yang memuat informasi tentang

aktifitas yang terjadi.

1. *Message* Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*



### 2.9.4 *Class Diagram*

*Class* adalah kumpulan objek-objek dengan dan yang mempunyai struktur umum, *behavior* umum, relasi umum, dan semantic/kata yang umum. *Class-class* ditentukan/ditemukan dengan cara memeriksa objek-objek dalam *sequence diagram* dan *collaboration diagram*. Sebuah *class* digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan.

**Tabel 2.22 Simbol *Class Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |
| 2. |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek |
| 3. |  | *Class* | Himpunan dari objek-objek yang  berbagi atribut serta operasi yang sama |
| 4. |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor* |
| 5. |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek |
| 6. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri |

### 2.10 *XAMPP*

*XAMPP* adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, *MySQL* *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *XAMPP* 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 digunakan yang dapat melayani tampilan halaman *web* yang dinamis (Bunafit Nugroho, 2008).

### 2.11 *Database MySQL*

#### 2.11.1 Database

Basis data atau *Database* merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya yang diaorganisasikan sesuai struktur tertentu dan disimpan dengan baik. Untuk mendapatkan informasi yang berguna dari kumpulan data maka diperlukan suatu perangkat lunak (*software*) untuk memanipulasi data sehingga mendapatkan informasi yang berguna. *Database* terbentuk dari beberapa komponen, yaitu (Bunafit Nugroho, 2008):

1. *Table*

*Table* atau Tabel adalah sekumpulan data dengan struktur yang sedemikian rupa, terbentuk dari *record* dan *field.* Istilah Tabel disini berbeda dengan istilah Tabel pada HTML, walaupun secara visual hampir sama.

1. *Record*

*Record* adalah sekumpulan *field* yang membentuk suatu objek tertentu.

1. *Field*

*Field* adalah atribut dari objek yang memiliki tipe data tertentu.

#### 2.11.2 MySQL

Menurut Abdul Kadir (2008), *MySQL* (dibaca: mi-se-kyu-el) merupakan software yang tergolong sebagai *DBMS* (*Database Management System*) yang bersifat *open source. Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (code yang dipakai untuk membuat *MySQL*). Selain tentu saja bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi dan bisa diperoleh secara gratis dengan mengunduh di *internet*.

*SQL* *(Structured Query Language)* merupakan sebuah bahasa *relational* yang berisi pernyataan yang digunakan untuk memasukkan, mengubah, menghapus, memilih dan melindungi data. *SQL* bukan *database* aplikasi, tetapi lebih berarti dengan suatu bahasa yang digunakan untuk mengajukan pertanyaan ke dalam *database* berupa pengguna *SQL*. *Database* sistem yang memiliki konsep sama dengan *SQL*, adalah *Postgres* dan *MySQL*, dimana *database* tersebut bisa didapatkan gratis atau dengan harga yang murah. *MySQL* adalah *server multithreaded*, sehingga sangat memungkinkan untuk menghandle permintaan layanan secara stimultan. Model *koneksi* dengan *protocol* *TCP-IP* membuat akses ke *server* *database* lebih cepat jika dibandingkan dengan menggunakan *mapping drive*.

### 2.12 *PHP* (*Hypertext Preprocessing*)

Jika diartikan *PHP* memiliki beberapa pandangan dalam mengartikannya, akan tetapi kurang lebih *PHP* dapat kita ambil arti sebagai PHP: *HypertextPreprocesor.* Ini merupakan bahasa yang hanya dapat berjalan pada *server* dan hasilnya dapat di tampilkan pada *client*. *PHP* adalah produk *open source* yang dapat digunakan secara gratis tanpa harus membayar untuk menggunakannya. *Interpreter PHP* dalam mengeksekusi kode *PHP* pada sisi *server* (*server side*), sedangkan tanpa adanya *Interpreter PHP*, maka semua skrip dan aplikasi *PHP* yang dibuat tidak dapat dijalankan (Bunafit Nugroho, 2008).

### 2.13 *HTML* (*Hyper Text Markup Language*)

*HTML* yang merupakan singkatan dari *Hyper Text Markup Language* adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman *web*. Didalamnya berisi kumpulan informasi yang disimpan dalam *tag-tag* tertentu, dimana *tag*-*tag* tersebut digunakan untuk melakukan format terhadap informasi yang dimaksud.

Berbagai pengembangan telah dilakukan terhadap kode *HTML* dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman *web*. Kendati demikian, sampai sekarang *HTML* tetap berdiri kokoh sebagai dasar dari bahasa *web* seperti *PHP*, *ASP*, *JSP* dan lainnya. Bahkan secara umum, mayoritas situs *web* yang ada di *internet* pun masih tetap menggunakan *HTML* sebagai teknologi utama (Constantianus & Suteja, 2010).

### 2.14 *Application Programming Interface* (API)

Secara umum API merupakan ekspresi terfokus keseluruhan fungsional dalam suatu modul software yang dapat diakses oleh orang yang membutuhkan dengan cara yang telah ditentukan layanan. Representasi terfokus dari fungsi yang dideklarasikan dalam API dimaksudkan untuk menyediakan rangkaian layanan yang spesifik untuk target tertentu. Jika dalam satu modul memiliki API ganda, hal ini sudah menjadi hal yang umum karena setiap API dimaksudkan untuk penggunaan yang spesifik dari modul terkait (Rama & Avinash, 2015).

# BAB III METODOLOGI PENELITIAN

## 3.1 Tempat dan Waktu Penelitian

### 3.1.1 Tempat Penelitian

Penelitian tugas akhir dengan judul Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra dilakukan pada *website pengaduan-propam-poldasultra.com.*

### 3.1.2 Waktu Penelitian

Waktu yang diperlukan dalam penelitian ini dapat dilihat pada Tabel 3.1: **Tabel 3.1 Waktu Kegiatan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Kegiatan |  | | | | Waktu (2018-2019) | | | | | | | |  |  | | |
| Desember | | | | Januari | | | | Februari | | | |  | Maret | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | *Inception* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | *Elaboration* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | *Construction* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | *Transition* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 3.2 Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini adalah dengan melakukan studi kepustakaan. Melalui studi pustaka penulis menghimpun data dari jurnal, Perpustakaan Prodi Teknik Informatika Universitas Halu Oleo, *Website* *pengaduan-propam-poldasultra.com* dan *interview* dosen dan pihak Propam Polda Sultra yang berkaitan dengan pembangunan sistem Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra. Baik itu untuk perancangan basis data antarmuka, operasi standar dan *interview* pihak yang terkait dalam pembuatan sistem.

37

## 3.3 Metode Pengembangan Sistem

Metode yang digunakan untuk membangun sistem Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra adalah *Rational Unified Model* (RUP). Adapun tahap–tahap dalam RUP adalah sebagai berikut :

### 3.3.1 Permulaan *(Inception)*

Pada tahap ini dilakukan penentuan ruang lingkup dan kebutuhan sistem. Ruang lingkup dan kebutuhan disesuaikan dengan hasil yang diperoleh saat pengumpulan data. Adapun hasil yang diperoleh berkaitan dengan sistem ini adalah sistem mengolah data yang berkaitan dengan sistem Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra.

### 3.3.2 Perluasan/ Perencanaan (*Elaboration)*

Perencanaan dari Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra dilakukan di tahap ini. Meliputi rancangan antarmuka, rancangan basis data dan seleksi operasi apa saja yang dapat diimplementasikan sesuai dengan ruang lingkup dan kebutuhan yang telah didefinisikan pada tahap permulaan (*inception*).

### 3.3.3 Konstruksi (*Construction)*

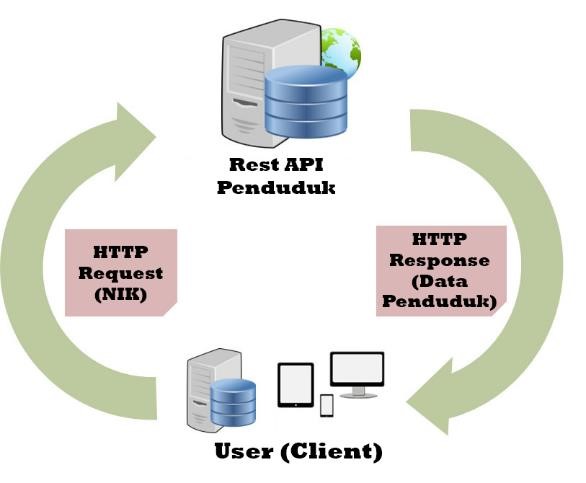
Pada tahap ini dilakukan pembangunan Sistem Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra melalui proses penulisan kode program (*coding*), pembuatan basis data, dan penginputan data. Pembangunan sistem dilakukan berdasarkan rancangan yang telah direncanakan pada tahap perluasan/ perencanaan (*elaboration*).

### 3.3.4 *Transisi (Transition)*

Pada tahap ini dilakukan proses pengujian terhadap Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra*.*

## 3.4 Analisis Sistem yang Direncanakan

Sistem Implementasi metode *Rocchio* untuk Klasifikasi laporan menggunakan merupakan sebuah sistem yang dapat membantu mengklasifikasikan laporan yang masuk di Propam Polda. Sistem ini dapat melakukan proses klasifikasi kategori laporan berdasarkan isi laporan , di mana *user* hanya perlu mengisi laporan/pengaduan ke sistem. Kemudian sistem akan melakukan klasifikasi kategori laporan secara otomatis, selanjutnya data tersebut disimpan di dalam *database*.



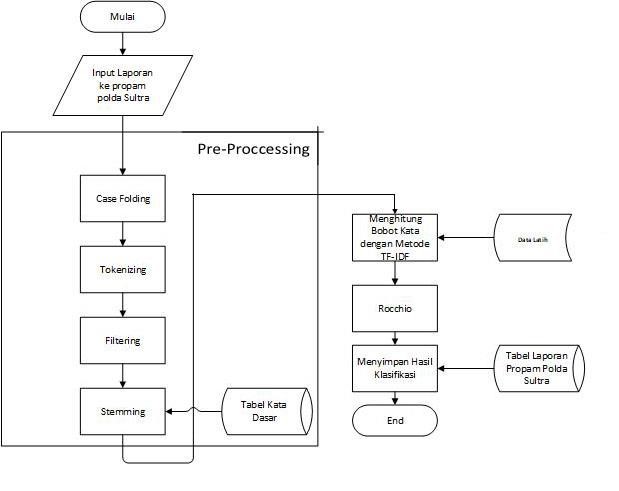
**Gambar 4.1 Arsitektur Sistem *API***

Gambar 4.1 mengilustrasikan proses kerja sistem. Proses kerja sistem ini *user* akan mengirimkan NIK ke *API* penduduk untuk mendapatkan data-data identitas dari *user*, kemudian *API* tersebut akan mengirimkan data identitas penduduk sesuai dengan NIK yang dikirimkan oleh *user*.Selanjutnya hasil *request* dari *user* akan ditampilkan pada *form* pengisian laporan pengaduan.

### 3.4.1 *Flowchart*

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* Sistem Implementasi metode *Rocchio* untuk Klasifikasi laporan yang ditunjukkan oleh Gambar 4.2. Adapun alur kerja *flowchart diagram* sistem adalah sebagai berikut:

1. *User* menginput laporan.
2. Isi laporan melalui proses *preprocessing*.
3. Laporan melalui proses *case folding*
4. Laporan melalui proses *tokenizing*
5. Laporan melalui proses *filtering*.
6. Laporan melalui proses *stemming,* dengan mencocokan kata pada laporan dengan kata dalam tabel kata dasar di *database* sistem.
7. Menghitung bobot kata dengan metode *TF-IDF* dan membandingkan dengan data latih.
8. Menghitung kecocokan laporan yang masuk menggunakan metode *Rocchio*.
9. Menentukan kategori laporan berdasarkan nilai tertinggi dari hasil perhitungan metode *Rocchio*.
10. Simpan laporan yang telah ditentukan kategorinya pada tabel laporan di *database*.

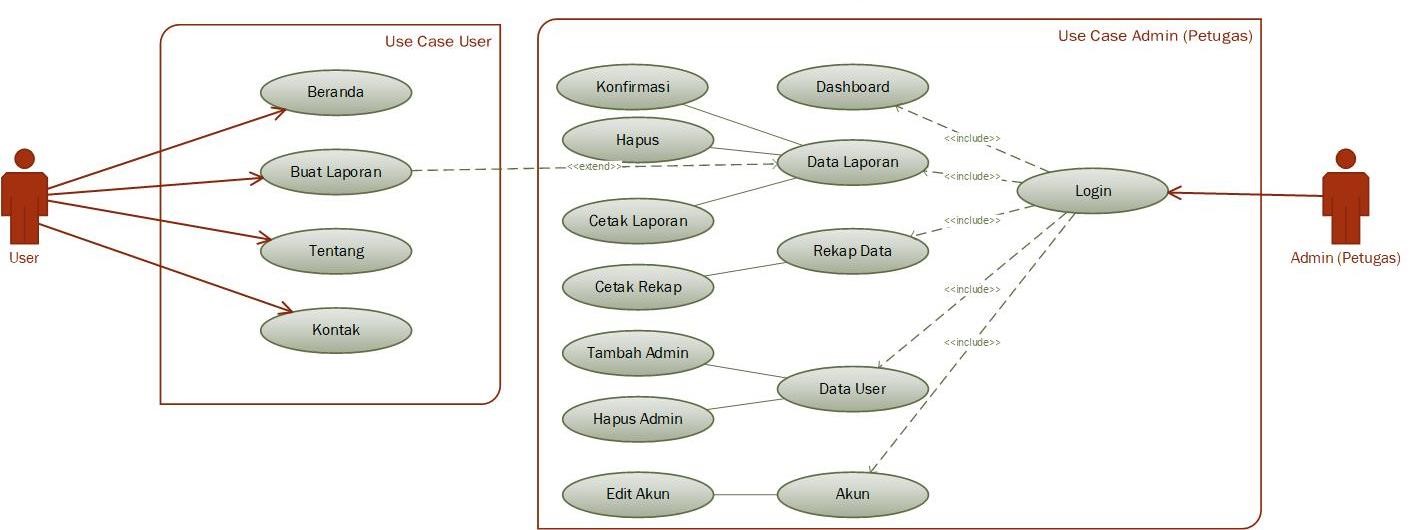


## Gambar 4.2 *Flowchart Diagram* Sistem

### 3.4.2 *Use Case* Diagram

Sistem ini mempunyai dua tingkatan pengguna, yaitu *administrator* dan *user*. Hak akses menu masing-masing pengguna pun berbeda sesuai dari tingkatan pengguna. Adapun deskripsi dari hak akses menu masing-masing adalah sebagai berikut :

1. *User* dapat mengakses menu buat laporan. Menu ini ditampilkan pada halaman *web* yang berbeda dari halaman *web* dari *administrator*.
2. *Administrator* dapat mengakses *login*, di mana fungsinya sebagai syarat untuk mengakses halaman *administrator*, yang meliputi beranda, data laporan, rekap data, pengaturan akun.



**Gambar 4.3 *Use Case* Diagram Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra.**

**3.4.3 Activity Diagram**

### 1. Buat Laporan(*User*)

Setelah menganalisis sistem, maka didapatkan rancangan diagram *activity* beranda (*user*) yang ditunjukkan oleh Gambar 4.4. Adapun alur kerja menu beranda (*user*) adalah sebagai berikut:

1. *User* mengakses sistem.
2. *User* memilih menu buat laporan sistem mengirim *form* laporan dan menampilkan *form* laporan.
3. *User* mengisi *form* laporan.
4. Laporan kemudian disimpan di *database*.

User

Sistem

Database

Memilih Buat Laporan

Membuka Website

Menampilkan

Form Laporan

Mengisi Form Laporan

Kirim Data

Simpan

Proses Klasifikasi

**Gambar 4.4 *Activity Diagram* Beranda (*User*)**

### 2. *Login (Administrator)*

Setelah menganalisis sistem, maka didapatkan rancangan diagram *activity* *login (administrator)* yang ditunjukkan oleh Gambar 4.5. Adapun alur kerja *login* (*admin*) adalah sebagai berikut :

1. *Admin* memasukkan *username* dan *password*.
2. Data akan divalidasi oleh sistem dengan mengecek kecocokan data yang ada pada *database*. Jika data *valid*, maka sistem akan menampilkan halaman beranda. Namun jika data tidak *valid*, maka sistem akan menampilkan pesan *error*, yang meminta *admin* memasukkan ulang *username* dan *password*.

Admin

Sistem

*Database*



Input username dan

p

assword



Validasi



Cek username dan

password



Pesan

Error



Masuk beranda

admin

Benar

Salah

## Gambar 4.5 *Activity Diagram Login (Administrator)*

### 3. Beranda (*Administrator*)

Setelah menganalisis sistem, maka didapatkan rancangan diagram *activity* beranda (*administrator*) yang ditunjukkan oleh Gambar 4.5. Jika *user id* dan *password* yang dimasukkan dalam *form login* benar, maka sistem akan mengalihkan halaman ke beranda khusus *admin*.

Admin

Sistem

Database



Username dan

p

assword

benar



Mengalihkan

halaman

ke beranda admin



Pilih data yang

dibutuhkan



Menampilkan

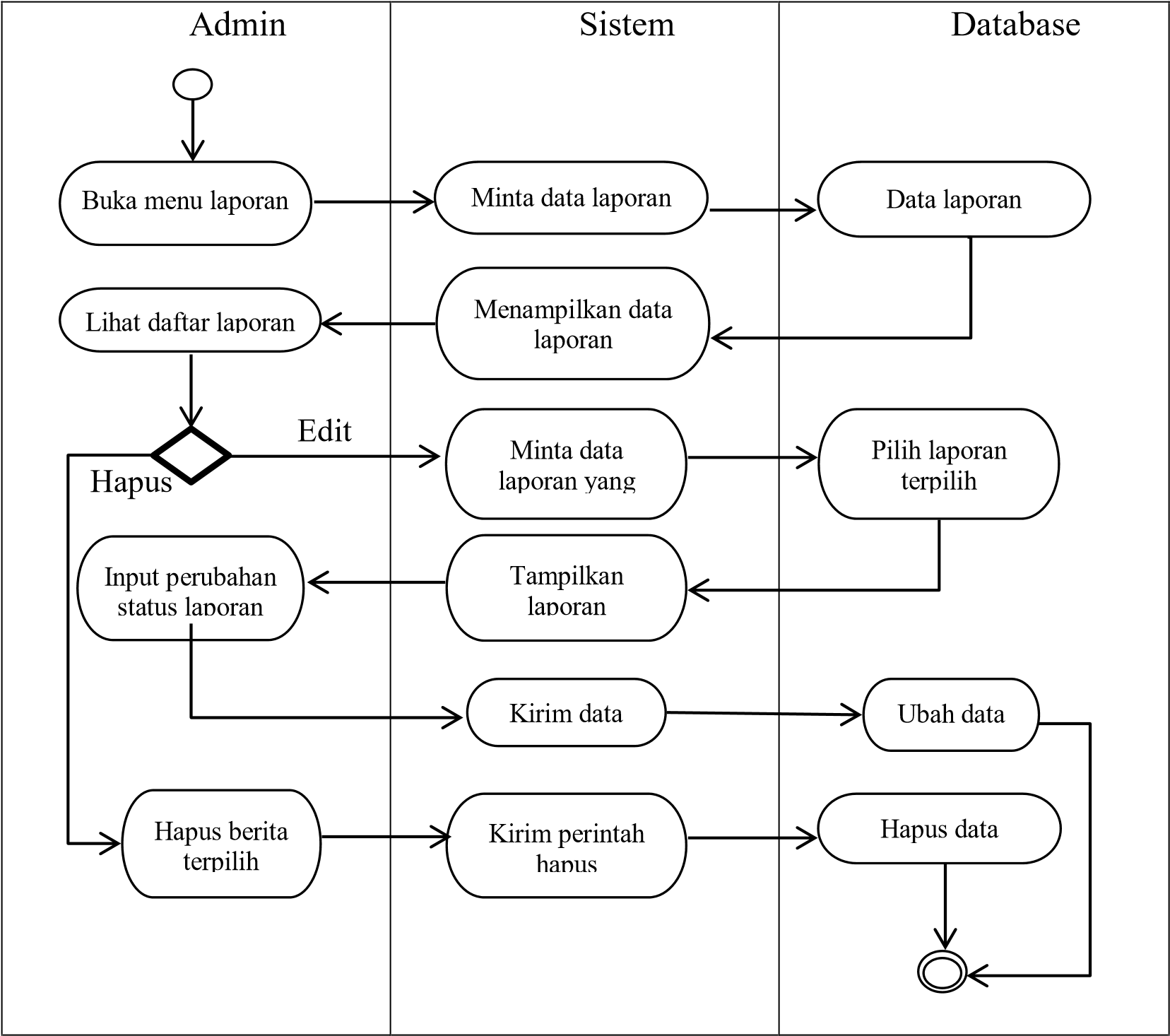
beranda admin



Meminta data yang

dibutuhkan

## Gambar 4.6 *Activity Diagram* Beranda (*Administrator*) 4. Laporan (*Administrator*)



**Gambar 4.7 *Activity Diagram* Laporan *(Administrator)***

### 5. Data *Admin* (*Administrator*)

Berdasarkan analisis sistem, maka didapatkan rancangan diagram *activity* data *admin* (*administrator*) yang ditunjukkan oleh Gambar 4.8. Adapun alur kerja *form* data *admin* adalah sebagai berikut:

1. *Admin* membuka menu data *admin*, yang secara bersamaan sistem akan mengirimkan perintah ke *database* untuk memilih tabel *admin*.
2. Daftar *admin* selanjutnya akan ditampilkan berupa tabel.
3. Jika *admin* memilih tomboltambah *admin*, maka akan diarahkan ke halaman *form* *input* *admin*, sehingga *admin* dapat memasukkan *admin* yang baru.
4. Jika *admin* memilih aksi hapus, maka data yang terpilih akan segera dihapus.

Admin

Sistem

Database



Simpan data



Validasi data



Hapus data admin

terpilih



Kirim perintah

hapus



Hapus data

admin terpilih



Buka menu data

admin



Input data

admin

Hapus

Tambah

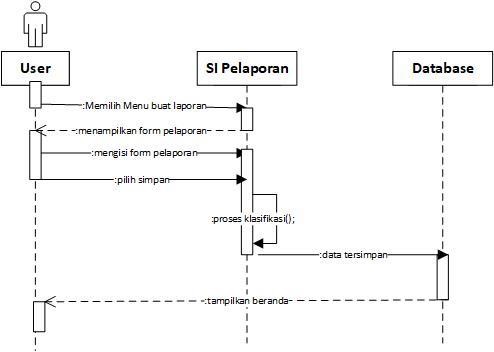
## Gambar 4.8 *Activity Diagram* Data *Admin* (*Administrator*)

**3.4.4 *Sequence Diagram***

### 1. Buat Laporan (*User)*

Setelah menganalisis sistem, maka didapatkan rancangan diagram *sequance* buat laporan yang ditunjukkan oleh Gambar 4.9. Adapun alur kerja menu buat laporan adalah sebagai berikut:

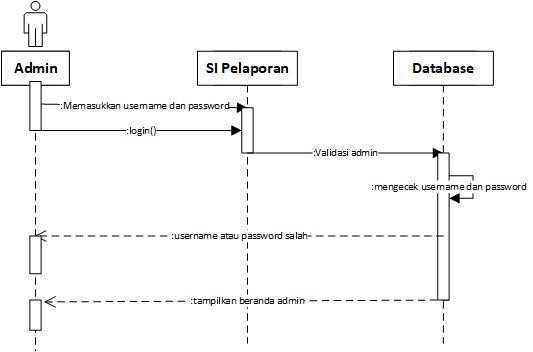
1. *User* memilih menu buat laporan.
2. Sistem menampilkan form pelaporan.
3. *User* mengisi *form* pelaporan yang dimana disini data pelapor dan terlapor harus diisi lengkap*.*
4. Lalu *user* memilih simpan.
5. Setelah itu laporan akan diklasifikasikan oleh sistem secara otomatis dan disimpan ke *database*.



## Gambar 4.9 *Sequence Diagram* Buat Laporan (*User*)

### 2. *Login (Administrator)*

Setelah menganalisis sistem, maka didapatkan rancangan diagram *Sequence* *login* (*administrato****r***) yang ditunjukkan oleh Gambar 4.10.



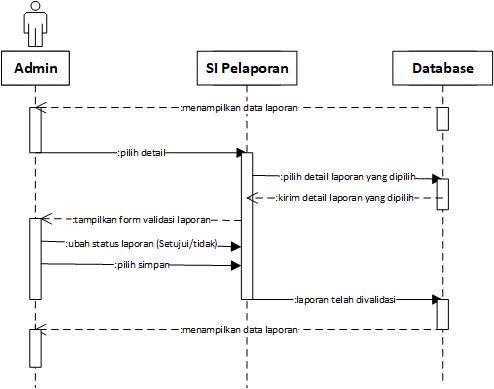
### Gambar 4.10 *Sequence Diagram Login (Administrator)*

Adapun alur kerja *login* (*administrato****r***) adalah sebagai berikut :

1. *Admin* memasukkan *username* dan *password*.
2. Data akan divalidasi oleh sistem dengan mengecek kecocokan data yang ada pada *database*. Jika data *valid*, maka sistem akan menampilkan halaman beranda. Namun jika data tidak *valid*, maka sistem akan menampilkan pesan *error*, yang meminta *admin* memasukkan ulang *username* dan *password*.

#### 3. Validasi Laporan (Administrator)

Setelah menganalisis sistem, maka didapatkan rancangan diagram *sequance* validasi laporan yang ditunjukkan oleh Gambar 4.11. Adapun alur kerja *form* validasi laporan adalah sebagai berikut :

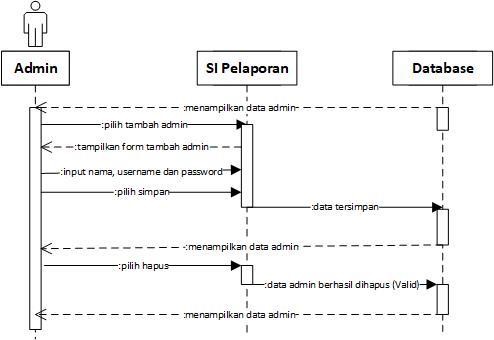


**Gambar 4.11 *Sequence Diagram* Validasi Laporan (*Administrator*)**

#### 4. Data Admin (Administrator)

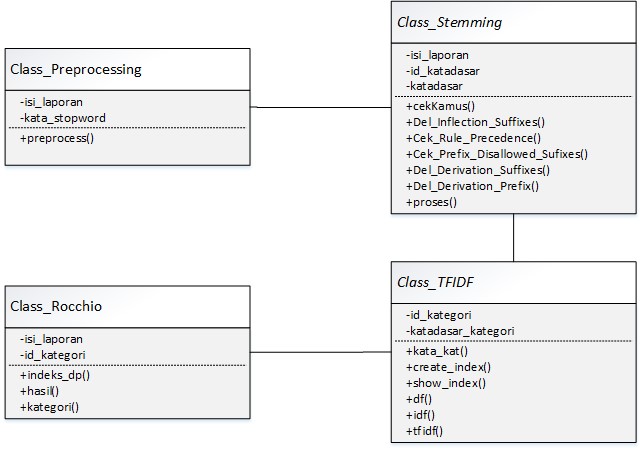
Setelah menganalisis sistem, maka didapatkan rancangan diagram *sequance* data *admin* yang ditunjukkan oleh Gambar 4.12. Adapun alur kerja menu data *admin* adalah sebagai berikut :

1. *Admin* membuka menu data *admin* , selanjutnya akan ditampilkan daftar *admin* pada halaman menu data *admin* berupa tabel.
2. Jika *admin* memilih tomboltambah *admin*, maka akan diarahkan ke halaman *form* *input* *admin*, sehingga memungkinkan *admin* untuk memasukkan *admin* yang baru.
3. Jika *admin* memilih aksi hapus, maka data yang terpilih untuk aksi tersebut akan segera dihapus.



### Gambar 4.12 *Sequence Diagram* Diagram Data *admin (Administrator)*

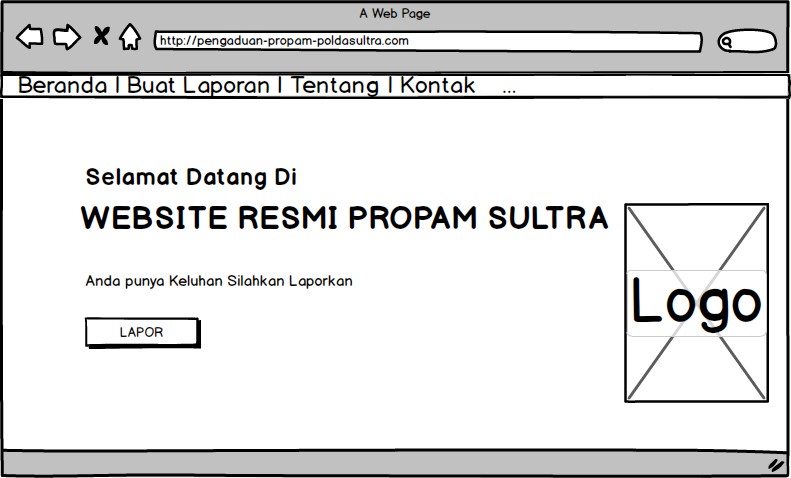
#### 3.4.5 Class Diagram



**Gambar 4.13 *Class Diagram* Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra**

#### 3.5 Perancangan Antarmuka (Interface)

##### 3.5.1 Beranda (User)



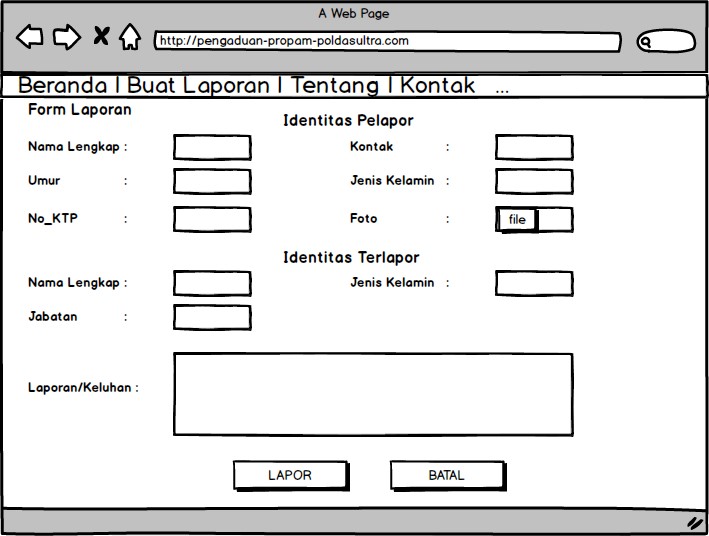
**Gambar 4.14 Halaman Beranda (*User*)**

Gambar 4.14 merupakan antarmuka halamanberanda *(user)* pada laman *pengaduan-propam-poldasultra.com*. Halamantersebut berisi:

1. Pada halaman beranda *(user)* terdapat empat menu utama, yaitu beranda, buat laporan, tentang kami dan kontak.
2. Beranda (*user)* berisi pemberitahuan selamat datang dan logo propam.

##### 3.5.2 Buat Laporan (User)

Gambar 4.15 merupakan rancangan antarmuka halaman formbuat laporan yang terdiri dari *input text*, *textarea* dan *input type file*.

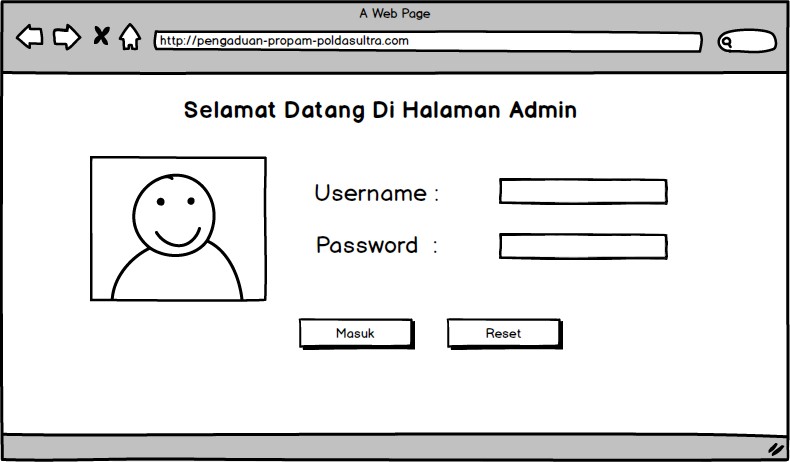


## Gambar 4.15 Halaman Buat Laporan (User)

### 3.5.3 *Login (Administrator)*

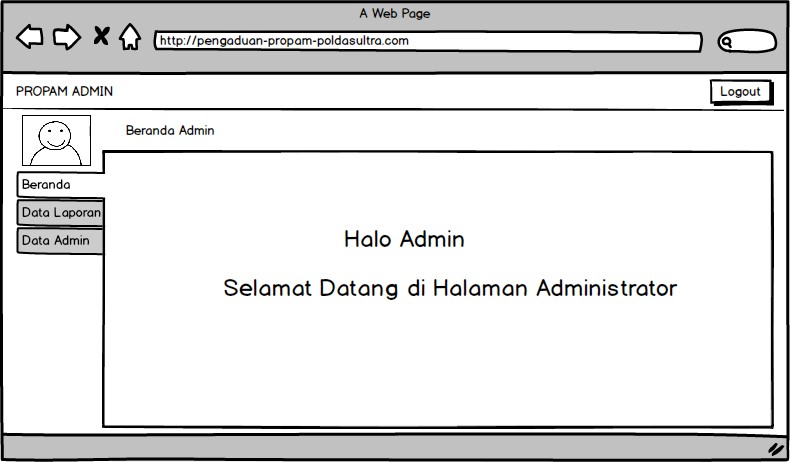
Gambar 4.16 merupakan rancangan antarmuka halaman *login* *(administrator)* Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra. Halamantersebut berisi:

1. *Textfield* *username*. Berfungsi sebagai tempat mengisikan *username* pengguna.
2. *Textfield password*. Berfungsi sebagai tempat mengisikan *password* pengguna.
3. *Button* masuk. Berfungsi sebagai pemberi perintah kepada sistem untuk mencocokkan data *username* dan *password* dengan data pengguna yang telah terdaftar di *database*.
4. *Button* masuk. Berfungsi sebagai pemberi perintah kepada sistem untuk mencocokkan data *username* dan *password* dengan data pengguna yang telah terdaftar di *database*.



## Gambar 4.16 Halaman *Login* (*Administrator*)

### 3.5.4Beranda *(Administrator)*



## Gambar 4.17 Halaman Beranda (*Administrator*)

Pada gambar 4.17 merupakan rancangan antarmuka halamanberanda *(administrator)* Implementasi Metode *Rocchio* untuk Klasifikasi Jenis Pelaporan di Laman Propam Polda Sultra.

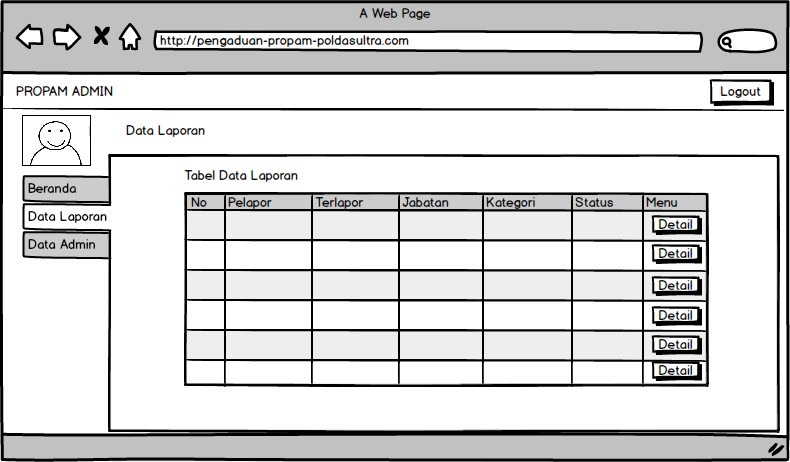
Pada halaman beranda (*administrator)*, ditampilkan tulisan selamat datang. Selain itu, pada halaman *administrator* terdapat beberapa menu yaitu, beranda, data laporan, akun, dan *logout*.

### 3.5.5Data Laporan (*Administrator*)

Gambar 4.18 merupakan rancangan antarmuka halamandata laporan

*(administrator)*. Halaman tersebut berisi :

1. Tabel yang berisi daftar laporan yang datanya telah tersimpan pada *database*.
2. *Button* detail, berfungsi sebagai pemberi perintah ke sistem agar menampilkan halaman detail laporan.



## Gambar 4.18 Halaman Data Laporan (*Administrator*)

### 3.5.6Detail Laporan (*Administrator*)

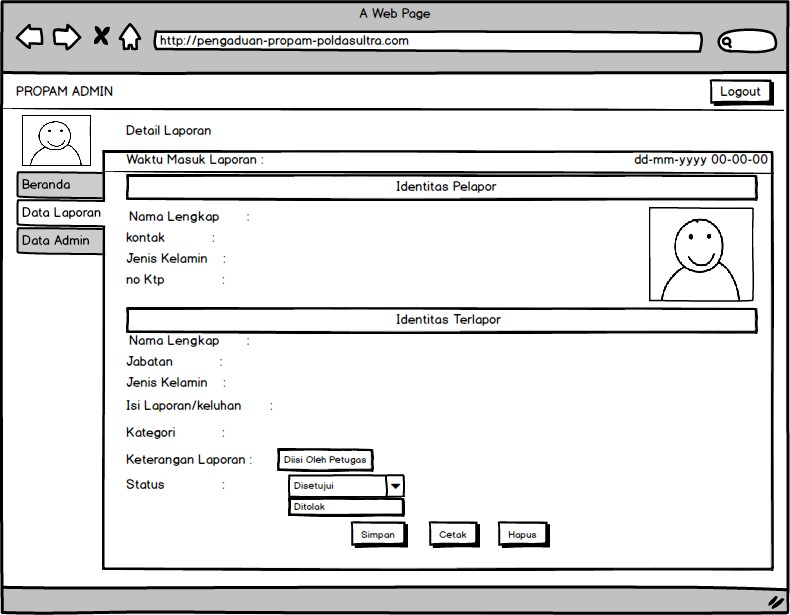
Gambar 4.19 merupakan rancangan antarmuka halamandata laporan

*(administrator)*. Halaman tersebut berisi :

1. Pada halaman ini berisi data-data laporan secara detail mengenai laporan.
2. Terdapat *input text* yang digunakan untuk admin untuk mengisi keterangan laporan.
3. Terdapat *combo box* yang digunakan untuk mengubah status laporan

(Disetujui/Tidak Disetujui).

1. *Button* simpan, berfungsi sebagai pemberi perintah ke sistem agar menyimpan data yang telah dimasukkan.
2. *Button* cetak, berfungsi sebagai pemberi perintah ke sistem agar mencetak data laporan.
3. *Button* hapus, berfungsi sebagai pemberi perintah ke sistem agar menghapus data laporan.



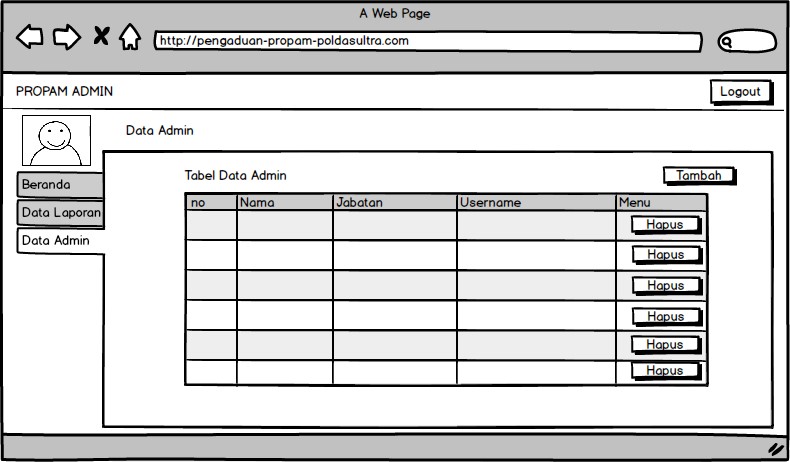
## Gambar 4.19 Halaman Detail Laporan (*Administrator*)

### 3.5.7Data *Admin* (*Administrator*)

Gambar 4.20 merupakan rancangan antarmuka halamandata *admin*

*(administrator)*. Halaman tersebut berisi:

1. Tabel yang berisi data akun (*admin*) yang datanya telah tersimpan pada *database*.
2. *Button* tambah. Berfungsi sebagai pemberi perintah ke sistem agar menampilkan halaman input *admin* baru
3. *Button* ubah. Berfungsi sebagai pemberi perintah ke sistem agar menampilkan halaman ubah data *admin*.
4. *Button* hapus. Berfungsi sebagai pemberi perintah ke sistem agar menghapus data *admin* yang dipilih.



**Gambar 4.20 Data *Admin* (*Administrator*)**

# DAFTAR PUSTAKA

Adhitia, 2009, *Ragam dan Analisis Kalimat Bahasa Indonesia*, Edisi 1, Muhammadiyah University Press, Surakarta.

Adriani, M., dkk, *Stemming Indonesian: A confix-stripping approach*, *ACM Transactions on Asian Language Information Processing (TALIP)*, Vol. 6, Issue. 4, Hal. 1-33.

Afriza, A., dan Adisantoso, J., *Metode Klasifikasi Rocchio untuk Analisis Hoax*, *Jurnal Ilmu Komputer dan Agri-Informatika,* No. 1,Vol. 5, Hal. 1–10, ISSN: 2089-6026.

A.S. Rosa dan Shalahuddin, M., 2013, *Visual Modelling Menggunakan UML dan Rational Rose*, Informatika, Bandung.

Belsky, G., 2012. Why Text Mining May Be The Next Big Thing, TIME Magazine, *http://business.time.com/2012/03/20/why-text-mining-may-be-the-next-bigthing/*, 20 Maret 2012, diakses 12 Desember 2018.

Berry, Michael W., dan Jacob Kogan, 2010, *Text mining: applications dan theory*. John Wiley & Sons, USA, ISBN: 978-0-470-74982-1.

Bhatnagar,S. dan Deane, A., 2004, *Building blocks of e-government: lessons from developing countries*, *Prem Notes Public Sector*, No 23, The World Bank.

Chakrabarti, Biswajit, Peter MA Calverley, dan Peter DO Davies, 2007, Tuberculosis dan its incidence, special nature, dan relationship with chronic obstructive pulmonary disease, *International journal of chronic obstructive pulmonary disease* , No. 3, Vol. 2, Hal: 285-295.

Brady, Malcolm, dan John Loonam, 2010, Exploring the use of entity-relationship diagramming as a technique to support grounded theory inquiry, *Qualitative Research in Organizations dan Management: An International Journal 5.3*, Hal: 224-237

Constantianus, Frederick, dan Bernard Renaldy Suteja, 2015, 9. Analisa dan Desain Sistem Bimbingan Tugas Akhir Berbasis Web dengan Studi Kasus Fakultas Teknologi Informasi." *Jurnal Informatika*, No. 2, Vol 1, Hal: 93-105.

Feldman, R., dan Sanger, J., 2007, *The text mining handbook: advanced approaches in analyzing unstructured data*, Cambridge University Press, Cambridge.

Han, J., Jian Pei, dan Kamber, M., *Data mining: concepts dan techniques*. Elsevier.

ISBN: 9780123814807

Hasugian, M., 2017, Klasifikasi Jurnal Berbahasa Inggris Berdasarkan Abstrak Dengan Algoritma Rocchio*, Skripsi,*  Fakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Sumatera Utara, Medan.

J. Hwan, C. Hoon, and S. Hyun, 2010, *Applying text and data mining techniques to forecasting the trend of petitions filed to e-People*, Elsevier, Hal. 7255–7268.

Joachims, T., 1997, A Probabilistic Analysis Of The Rocchio Algorithm With Tfidf For Text Categorization. *Proceeding Icml Proceedings Of The Fourteenth International Conference On Machine Learning*, Hal. 143-151.

Kadir, A., Dasar pemrograman web dinamis menggunakan PHP. Dani, 2003.

Khan, A., Baharuddin, B., & Lee, H., 2010. A Review Of Machine Learning Algorithms For Text-Documents Classification, *Journal Of Advances In Information Technology*, Hal. 4-20.

Lumbanraja, F. R., 2013, *Sistem Pencarian Data Teks dengan Menggunakan Metode Klasifikasi Rocchio(Studi Kasus:Dokumen Teks Skripsi), Skripsi,* MIPA, Universitas Lampung, Bandar Lampung.

Manning, C., Raghavan P., Schűtze H., 2008. *Introduction to Information Retrieval*, Cambridge University Press, Cambridge.

Nugroho. E., Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp, *Skripsi,* MIPA, Universitas Brawijaya, Malang.

Nugroho, B., *Aplikasi pemrograman web dinamis dengan PHP dan MySQL*, Gava Media, Yogyakarta.

Nazief, B., dan Adriani, M., *Confix-stripping: Approach to Stemming Algorithm for Bahasa Indonesia. Internal publication*, Computer Science, Universitas Indonesia, Depok, Jakarta.

Pahlevy, R., 2010, Pengertian Flowchart dan definisi data, *http://www.ldanasanteori.com /2015/10/ flowchart-dan definisi-data.html*, diakses tanggal 15 November 2017.

Raharjo, B., 2001, Membangun e-Government, *Seminar Nasional Jaringan Komputer II*, Makassar.

Rama, Girish Maskeri dan Avinash Kak, 2013, *Software – Practice and Experience*, New Jersey: Wiley Online Library.

Robertson, S., 2004, Understanding inverse document frequency: on theoretical arguments for IDF, Journal of documentation, Vol. 60, Issue: 5, Hal. 503-520.

Selvi, S. P., dkk., 2016, Text Categorization using Rocchio Algorithm and Random Forest Algorithm, *IEEE Eighth International Conference on Advanced Computing (ICoAC)*, Hal. 7-12.

Uden, Mark van, 2011, Rocchio: Relevance Feedback in Learning Classification Algorithms. *Thesis*, Department of Computing Science, University of Nijmegen. Nijmegen.

Weya Pantouw,J.C., 2017, Perbandingan Klasifikasi Rocchio Dan Multinomial Naïve Bayes Pada Analisis Sentimen Data Twitter Bahasa Indonesia*, Skripsi,* MIPA, Institut Pertanian Bogor, Bogor.

Widjojo, E.A., Rachmat, C.A., dan Santosa, R.G., 2014, *Implementasi Rocchio’s*

*Classification dalam Mengkategorikan Renungan Harian Kristen.*

Ultimatics*,* No. 1, Vol. 6, Hal. 1-7, ISSN 2085-4552.

Z. Xiang, Z. Schwartz, J. H. Gerdes, and M. Uysal, 2015, *What can big data and text analytics tell us about hotel guest experience and satisfaction?,* *Int. J.*

*Hosp. Manag*., Vol. 44, Hal. 120–130.