

**PENERAPAN *MAXIMUM* TF-IDF NORMALIZATION
TERHADAP METODE KNN UNTUK KLASIFIKASI *DATASET*
MULTICLASS PANICHELLA PADA *REVIEW* APLIKASI *MOBILE***

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Billy Kaleb Hananto
NIM: 145150200111012



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENERAPAN *MAXIMUM TF-IDF NORMALIZATION* TERHADAP METODE KNN
UNTUK KLASIFIKASI DATASET *MULTICLASS PANICHELLA* PADA *REVIEW APLIKASI
MOBILE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Billy Kaleb Hananto
NIM: 145150200111012

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II



Aryo Pinandito, S.T, M.MT
NIP: 19830519 201404 1 001

Agi Putra Kharisma, S.T, M.T
NIK: 20130486 0430 1 001



Mengetahui
Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001



PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018



Billy Kaleb Hananto

NIM: 145150200111012



KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Allah YME, karena hanya dengan rahmat dan Karunia-Nya, penulis dapat menyelesaikan skripsi dengan judul “PENERAPAN *MAXIMUM TF-IDF NORMALIZATION* TERHADAP METODE KNN UNTUK KLASIFIKASI DATASET *MULTICLASS PANICHELLA* PADA *REVIEW* APLIKASI *MOBILE*”. Tugas akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Penulis juga menyampaikan rasa terima kasih kepada pihak – pihak yang telah membantu penulis diantaranya:

1. Bapak Aryo Pinandito, S.T, M.MT, selaku dosen pembimbing I yang telah memberikan masukan, ilmu serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
2. Bapak Agi Putra Kharisma, S.T, M.T, selaku dosen pembimbing II yang telah memberikan masukan, ilmu serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
3. Orang tua yang selalu menyemangati, memberikan doa dan memberi jalan sehingga terselesaikannya skripsi ini.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Tri Astoto Kurniawan S.T, M.T, Ph.D, selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Teman-teman yang telah membantu memberi masukan atas skripsi ini.
7. Seluruh pihak yang tidak dapat penulis sebutkan satu persatu yang terlibat secara langsung maupun tidak langsung dalam proses pengerjaan skripsi ini.

Malang, 2 Agustus 2018

Penulis

billykaleb@yahoo.com

ABSTRAK

Billy Kaleb Hananto, 2018. Penerapan *Maximum* TF-IDF Normalization terhadap metode KNN untuk Klasifikasi *Dataset Multiclass* Panichella pada *Review Aplikasi Mobile*. Skripsi Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Aryo Pinandito, S.T, M.MT dan Agi Putra Kharisma, S.T, M.T.

Review aplikasi mobile merupakan sumber wacana dari pengembang aplikasi, dimana dari *review* ini bisa didapat pengetahuan berupa kearah mana aplikasi harus dibawa dan diperbaiki. *Review* ini sendiri memiliki banyak jenis, baik yang positif ataupun negatif, baik yang memberikan info, atau hanya sekedar menanyakan permasalahan. Diperlukan waktu, usaha dan ketelitian yang cukup tinggi apabila ingin mengklasifikasikan *review* ini secara manual. Maka dari itu, diperlukan sebuah metode yang dapat mengklasifikasikan secara otomatis *review* ini, terutama kedalam kategori kategori unik yang diharapkan dapat membantu pengembang dalam mengarahkan aplikasinya, lebih dari sekedar *review* positif atau negatif, salah satunya adalah 4 kategori *review* yang dikembangkan oleh Panichella. Selain berbagai macam kategori tersebut, penelitian ini membahas salah satu metode klasifikasi yang cukup populer yaitu *Term Frequency – Inverse Document Frequency* dan *K-Nearest Neighbor*. Kombinasi kedua metode ini sering digunakan untuk mengklasifikasikan teks dalam berbagai bidang. Penelitian ini mencoba menerapkan metode bernama *Maximum Term Frequency – Inverse Document Frequency*, metode normalisasi yang bekerja dengan cara membagi nilai frekuensi dengan jumlah kata terbanyak yang muncul, dengan harapan bahwa metode *K-Nearest Neighbor* dapat memberikan hasil akurasi yang lebih tinggi dengan diubahnya metode *Term Frequency – Inverse Document Frequency* ini.

Kata kunci: Klasifikasi, KNN, TF-IDF, Panichella.

ABSTRACT

Billy Kaleb Hananto, 2018. Appliacion of Maximum TF-IDF Nomalization into KNN method to Classify Panichella's Multiclass Dataset in case of Mobile Application's Review. Thesis Departement of Informatics, Faculty of Computer Science, Universitas Brawijaya, Malang. Supervisor: Aryo Pinandito, S.T, M.MT and Agi Putra Kharisma, S.T, M.T.

Mobile Application's Review is a source for application developer to develop their apps into the way user's want to. There's multiple type of Mobile Application's Review itself, from positive review to negative review, from giving information to just stating problems. Time and effort is needed to classify these review manually. Hence, there's a need for some method to classify these review automatically, especially to unique categories that hopefully can help developer to improve their applications, more than just negative and positive review, one of them being Panichella's Categories which categorize review into 4 categories. Beside those categories, this research will talk about one classification method that really popular, which is Term Frequency – Inverse Document Frequency and K-Nearest Neighbor. Combination of these two method are usually used to categorize texts. This research will try to apply a method called Maximum Term Frequency – Inverse Document Frequency, normalization technique that works by dividing a Term's Frequency by it's Document's highest number of Term, which hopefully will help increase K-Nearest Neighbor's accuracy.

Keywords: Classification, KNN, TF-IDF, Panichella.

DAFTAR ISI

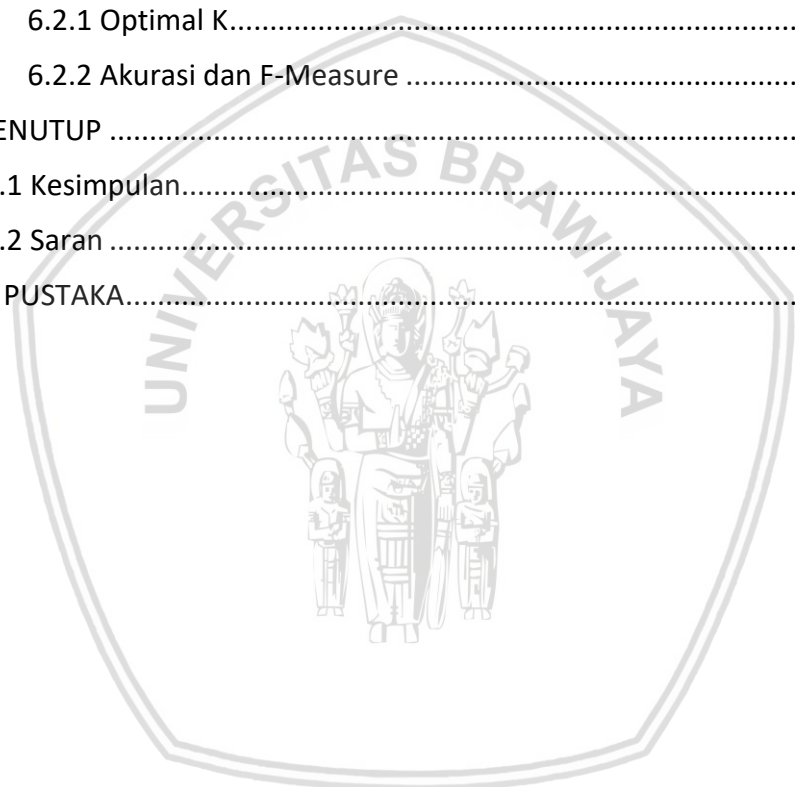
PERNYATAAN ORISINALITAS	ii
KATA PENGANTAR.....	iii
ABSTRAK.....	v
ABSTRACT	vi
DAFTAR ISI.....	vii
DAFTAR Gambar.....	x
DAFTAR TABEL.....	xi
BAB 1 PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah.....	3
1.3 Tujuan	3
1.4 Batasan masalah	3
1.5 Sistematika pembahasan.....	4
BAB 2 TINJAUAN PUSTAKA.....	5
2.1 <i>Review</i>	5
2.2 Preprocessing.....	5
2.2.1 Filtering Stopwords	5
2.2.2 <i>Stemming</i>	5
2.2.3 <i>Term Weighting</i>	6
2.3 Pengklasifikasian Teks.....	8
2.3.1 K-Nearest Neighbor.....	8
2.3.2 KNN + TF-IDF	10
2.3.3 F-Measure	10
2.3.4 Akurasi.....	11
BAB 3 METODOLOGI PENELITIAN	12
3.1 Studi Pustaka.....	13
3.2 Analisis dan Perancangan	13
3.3 Pembuatan Program TF-IDF dan KNN	13
3.4 Pemrosesan Dataset	14
3.4.1 <i>Review Extracting</i>	14



3.4.2 Filtering <i>Stopword</i>	15
3.4.3 <i>Stemming</i>	15
3.5 Pengujian dan Analisis	15
3.5.1 Perubahan Dataset menjadi nilai melalui TF-IDF.....	15
3.5.2 Pengujian menggunakan dataset testing.....	16
3.6 Kesimpulan.....	16
BAB 4 PERANCANGAN.....	17
4.1 Persiapan Dataset	17
4.1.1 Penjelasan dan Contoh	17
4.1.2 Pola Kategori Panichella.....	19
4.2 Daftar Kebutuhan	20
4.2.1 Kebutuhan Fungsional.....	20
4.2.2 Use Case Diagram	22
4.2.3 Use Case Scenario	22
4.3 Perancangan TF-IDF.....	26
4.3.1 <i>Raw TF</i>	26
4.3.2 <i>Max TF</i>	27
4.3.3 IDF	29
4.3.4 TF-IDF	30
4.4 Perancangan KNN	30
4.4.1 <i>Euclidean Distance</i>	30
4.4.2 <i>Cosine Similarity</i>	32
4.5 Perancangan Klasifikasi Teks	33
4.6 Perhitungan Manual	34
BAB 5 IMPLEMENTASI	42
5.1 Lingkungan Implementasi.....	42
5.1.1 Lingkungan Implementasi Perangkat Keras.....	42
5.1.2 Lingkungan Implementasi Perangkat Lunak	42
5.2 Penerapan TF-IDF.....	42
5.2.1 <i>Raw Term Frequency</i>	43
5.2.2 <i>Maximum Term Frequency</i>	43
5.2.3 <i>Inverse Document Frequency</i>	44



5.3 Penerapan KNN.....	45
5.3.1 <i>Euclidean Distance</i> KNN	45
5.3.2 <i>Cosine Similarity</i> KNN	48
BAB 6 Pengujian dan Analisis	52
6.1 Dataset Positif Negatif	52
6.1.1 Optimal K.....	52
6.1.2 Akurasi and F-Measure	55
6.2 Dataset Panichella	60
6.2.1 Optimal K.....	60
6.2.2 Akurasi dan F-Measure	64
BAB 7 PENUTUP	76
7.1 Kesimpulan.....	76
7.2 Saran	77
DAFTAR PUSTAKA.....	78



DAFTAR GAMBAR

Gambar 2.1 StopWords Reuters-RCV1	5
Gambar 2.2 Contoh KNN dengan nilai K yang berbeda beda	9
Gambar 4.1 Flowchart <i>Raw TF</i>	26
Gambar 4.2 Flowchart <i>Max TF</i>	28
Gambar 4.3 Flowchart <i>IDF</i>	29
Gambar 4.4 Flowchart <i>Euclidean Distance</i>	31
Gambar 4.5 Flowchart <i>Cosine Similarity</i>	33
Gambar 6.1 Tabel perbandingan Akurasi <i>Raw TF-IDF</i> dan <i>Max TF-IDF</i> terhadap KNN <i>Euclidean Distance</i>	56
Gambar 6.2 Tabel perbandingan F-Measure <i>Raw TF-IDF</i> dan <i>Max TF-IDF</i> terhadap KNN <i>Euclidean Distance</i>	57
Gambar 6.3 Tabel perbandingan Akurasi <i>Raw TF-IDF</i> dan <i>Max TF-IDF</i> terhadap KNN <i>Cosine Similarity</i> Dataset Positif Negatif	58
Gambar 6.4 Tabel perbandingan F-Measure <i>Raw TF-IDF</i> dan <i>Max TF-IDF</i> terhadap KNN <i>Cosine Similarity</i> Dataset Positif Negatif	59
Gambar 6.5 Tabel Akurasi KNN <i>Euclidean Distance</i> Dataset Panichella.....	67
Gambar 6.6 Tabel F-Measure KNN <i>Euclidean Distance</i> Dataset Panichella	67
Gambar 6.7 Tabel Akurasi KNN <i>Cosine Similarity</i> Dataset Panichella	70
Gambar 6.8 Tabel F-Measure KNN <i>Cosine Similarity</i> Dataset Panichella	70
Gambar 6.9 Akurasi per kelas <i>Raw TF-IDF</i> + KNN <i>Euclidean Distance</i>	71
Gambar 6.10 Akurasi per kelas <i>Max TF-IDF</i> + KNN <i>Euclidean Distance</i>	71
Gambar 6.11 F-Measure per kelas <i>Raw TF-IDF</i> + KNN <i>Euclidean Distance</i>	72
Gambar 6.12 F-Measure per kelas <i>Max TF-IDF</i> + KNN <i>Euclidean Distance</i>	72
Gambar 6.13 Akurasi per kelas <i>Raw TF-IDF</i> + KNN <i>Cosine Similarity</i>	73
Gambar 6.14 Akurasi per kelas <i>Max TF-IDF</i> + KNN <i>Cosine Similarity</i>	73
Gambar 6.15 F-Measure per kelas <i>Raw TF-IDF</i> + KNN <i>Cosine Similarity</i>	74
Gambar 6.16 F-Measure per kelas <i>Max TF-IDF</i> + KNN <i>Cosine Similarity</i>	74

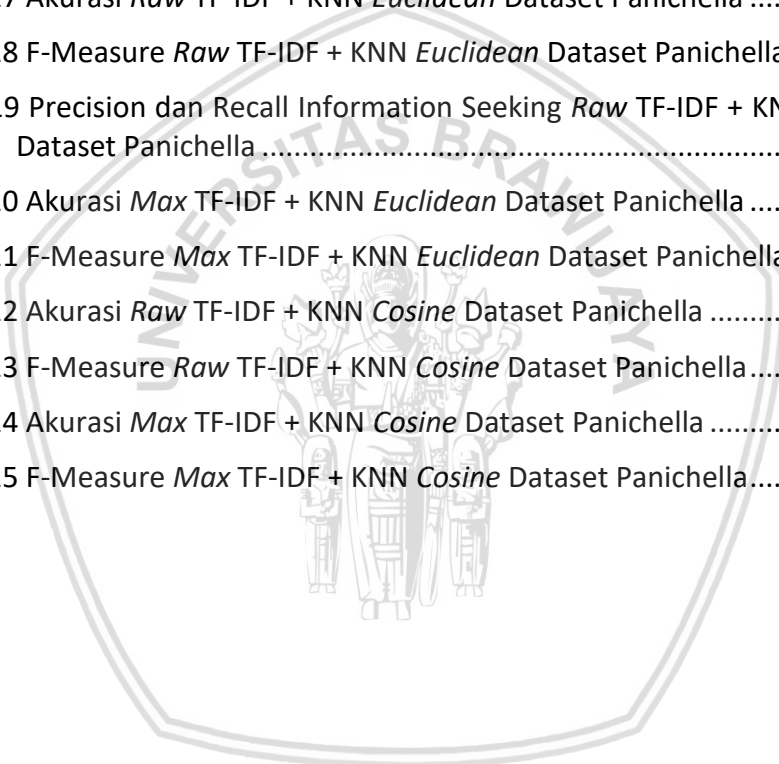


DAFTAR TABEL

Tabel 4.1 Deskripsi Kategori <i>Review</i> Dataset Positif Negatif	17
Tabel 4.2 Deskripsi Kategori <i>Review</i> Multiclass Dataset Panichella	18
Tabel 4.3 Deskripsi Kategori <i>Review</i> Multiclass Dataset Panichella	19
Tabel 4.4 Data Latih Perhitungan Manual	34
Tabel 4.5 Data Uji Perhitungan Manual	34
Tabel 4.6 Data Latih Perhitungan Manual setelah <i>preprocessing</i>	35
Tabel 4.7 Data Uji Perhitungan Manual setelah <i>preprocessing</i>	35
Tabel 4.8 Frekuensi dan nilai IDF setiap istilah dari data latih.....	35
Tabel 4.9 Nilai <i>Max</i> TF-IDF tiap istilah dalam tiap kelas pada data latih	37
Tabel 4.10 Nilai <i>Max</i> TF-IDF tiap istilah dalam tiap kelas pada data uji	38
Tabel 4.11 Hasil Pengujian Data Uji Pertama.....	39
Tabel 4.12 Hasil Pengujian Data Uji Kedua	40
Tabel 4.13 Hasil Pengujian Data Uji Ketiga	40
Tabel 4.14 Hasil Pengujian Data Uji Keempat.....	41
Tabel 5.1 <i>Raw Term Frequency</i>	43
Tabel 5.2 <i>Maximum Term Frequency</i>	43
Tabel 5.3 <i>Inverse Document Frequency</i>	45
Tabel 5.4 <i>Euclidean Distance</i>	45
Tabel 5.5 <i>Cosine Similarity</i>	48
Tabel 6.1 Pengujian Nilai K Dataset Positif Negatif <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> 52	
Tabel 6.2 Pengujian Nilai K Dataset Positif Negatif <i>Max</i> TF-IDF + KNN <i>Euclidean</i> 53	
Tabel 6.3 Pengujian Nilai K Dataset Positif Negatif <i>Raw</i> TF-IDF + KNN <i>Cosine</i> 54	
Tabel 6.4 Pengujian Nilai K Dataset Postif Negatif <i>Max</i> TF-IDF + KNN <i>Cosine</i> 54	
Tabel 6.5 Akurasi dan F-Measure <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Positif Negatif	55
Tabel 6.6 Akurasi dan F-Measure <i>Max</i> TF-IDF + KNN <i>Euclidean</i> Dataset Positif Negatif	55
Tabel 6.7 Akurasi dan F-Measure <i>Raw</i> TF-IDF + KNN <i>Cosine</i> Dataset Positif Negatif	57
Tabel 6.8 Akurasi dan F-Measure <i>Max</i> TF-IDF + KNN <i>Cosine</i> Dataset Positif Negatif	58



Tabel 6.9 Akurasi <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella	60
Tabel 6.10 F-Measure <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella.....	61
Tabel 6.11 Akurasi <i>Max</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella	61
Tabel 6.12 F-Measure <i>Max</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella.....	62
Tabel 6.13 Akurasi <i>Raw</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella	62
Tabel 6.14 F-Measure <i>Raw</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella.....	63
Tabel 6.15 Akurasi <i>Max</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella	63
Tabel 6.16 F-Measure <i>Max</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella.....	63
Tabel 6.17 Akurasi <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella	64
Tabel 6.18 F-Measure <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella.....	65
Tabel 6.19 Precision dan Recall Information Seeking <i>Raw</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella	65
Tabel 6.20 Akurasi <i>Max</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella	66
Tabel 6.21 F-Measure <i>Max</i> TF-IDF + KNN <i>Euclidean</i> Dataset Panichella.....	66
Tabel 6.22 Akurasi <i>Raw</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella	68
Tabel 6.23 F-Measure <i>Raw</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella.....	68
Tabel 6.24 Akurasi <i>Max</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella	69
Tabel 6.25 F-Measure <i>Max</i> TF-IDF + KNN <i>Cosine</i> Dataset Panichella.....	69



BAB 1 PENDAHULUAN

1.1 Latar belakang

Playstore adalah sarana penyebaran aplikasi secara digital. Pengguna dapat mencari, memilih dan mengunduh aplikasi yang tersedia pada *Playstore*, dari sekian banyak aplikasi yang tersedia. Pengguna juga dapat meninggalkan pendapat mereka mengenai suatu aplikasi, berupa *review*, yang didalamnya terkandung banyak hal. *Review* ini sendiri berfungsi sebagai penghubung antara pengguna dan pengembang, dimana pengguna dapat memberikan info info penting mengenai kondisi aplikasi yang dikembangkan, sehingga pengembang dapat mempertimbangkan masukan pengguna yang tertera dalam *review*. Pengguna dapat memberikan petunjuk kepada pengembang, permasalahan yang mungkin tidak tampak dari sudut pandang seorang pengembang, dan hanya nampak dari sudut pandang seorang pengguna. Selain perbaikan, *review* juga dapat berisi saran, dimana saran dapat membantu pengembang dalam mengarahkan aplikasi dan menentukan fitur apa yang selanjutnya akan ditambahkan kedalam aplikasi dimana pengguna memberikan masukannya.

Namun, *review* sendiri tidak semua berisi masukan, atau bahkan info yang dapat di ekstrak. Ada *review* yang hanya berisi ungkapan ketertarikan atau hujatan, yang walaupun membantu dalam penilaian keberhasilan aplikasi, tidak begitu membantu pihak pengembang dalam mengembangkan aplikasi, apabila dibandingkan dengan *review* yang memberikan info. Terdapat banyak jenis klasifikasi *review* aplikasi. Salah satu klasifikasi yang paling sederhana adalah positif dan negatif. Klasifikasi sederhana ini biasanya sulit apabila ingin digunakan oleh *developer*, karena *review* positif dan *review* negatif sendiri dapat berisi data yang berguna untuk *developer*, maupun hanya sekedar pengguna yang meluapkan pendapat dan emosinya, yang relatif kurang berguna dalam perkembangan aplikasi (Panichella et al, 2015). Untuk pengkategorian kompleks, terdapat penelitian yang membagi *review* menjadi 6 jenis, yaitu *Feature Request*, *Opinion Asking*, *Problem Discovery*, *Solution Proposal*, *Information Seeking* dan *Information Giving* (Guzzi et al, 2012). Kategori ini dibagi lagi menjadi 4 kategori yang lebih konkrit melalui proses perbandingan dengan data lain milik Pagano et al (2012). 4 kategori baru yang dikemukakan oleh Pagano ini meliputi *Information Giving*, *Information Seeking*, *Feature Request* dan *Problem Discovery* (Panichella et al, 2015).

Walaupun pengembang bisa mendapatkan banyak keuntungan dari *review-review* yang sudah disediakan oleh pengguna, pengembang masih menghadapi berbagai macam permasalahan. Salah satu permasalahan yang sering muncul adalah jumlah *review* yang sangatlah banyak. Setiap harinya, jumlah *review* rata rata yang diterima sebuah aplikasi adalah sejumlah 23 *review*. Untuk aplikasi aplikasi yang populer seperti Facebook, jumlah *review* yang diterima setiap harinya meningkat hingga berjumlah rata rata 4.275 *review* (Pagano et al, 2012). Jumlah ini bukanlah jumlah yang kecil, dimana pengembang akan mengalami

kesulitan apabila harus membaca dan mengerti maksud dari setiap *review*. Selain itu, terdapat juga permasalahan kualitas *review* seperti yang sudah dibahas pada paragraf 1 dan 2, yang terkadang membuat inti dari *review* tidak dapat langsung tersampaikan pada pengembang dalam satu kali pembacaan. Hal ini memunculkan perlunya sebuah sistem yang dapat mengklasifikasikan *review* yang didapat secara otomatis.

Review ini sendiri tentunya sering dipenuhi dengan kumpulan kata yang mengekspresikan sebuah perasaan atau pendapat. Untuk dapat diproses oleh komputer (dalam proses klasifikasi), diperlukan proses lain untuk merubah kumpulan kata ini menjadi angka, dan kemudian memperhitungkan jarak perbedaan akan hasil angka angka ini. Salah satu metode yang dapat melakukan proses perubahan kata kata menjadi angka adalah proses *Term Frequency – Inverse Document Frequency* (TF-IDF), sedangkan pengklasifikasian dari jarak yang didapat adalah *K-Nearest Neighbor* (KNN). Pada dasarnya, proses TF-IDF ini akan menentukan frekuensi relatif sebuah kata tertentu pada satu dokumen, dan membandingkan angka yang didapat dengan proposi kata tersebut pada sebuah korpus (Ramos, 2003). Namun *review* sendiri tidak memiliki jumlah kata yang tetap untuk setiap dokumen, dan proses TF-IDF memiliki masalah dalam menilai secara adil sebuah korpus yang memiliki dokumen dengan jumlah kata yang berbeda, dengan beberapa kata yang diulang. Untuk menanggulangi permasalahan ini, metode TF-IDF dimodifikasi dengan *Maximum TF Normalization* menjadi *Maximum TF-IDF* (*Max TF-IDF*), sehingga metode dapat mengurangi permasalahan dimana nilai *Term Frequency* semakin tinggi untuk dokumen yang semakin panjang (Manning et al, 2008).

Sedangkan untuk proses KNN, akan dicari jumlah tetangga terbanyak dimana sebuah data yang tidak memiliki kelas berada, kemudian data tersebut akan dimasukkan kedalam kategori dimana tetangga terbanyaknya berada (Bruno et al, 2014). Untuk mencari tetangga terdekat ini, tentunya diperlukan cara untuk menghitung jarak antara nilai data yang dicari dan nilai data data lainnya. Beberapa cara penghitungan yang biasa dipakai untuk metode KNN ini adalah *Euclidean Distance* dan *Cosine Similarity*, dimana *Euclidean Distance* akan mencari jarak secara langsung dari dua titik, dan *Cosine Similarity* akan mencari nilai *cosine* dari sudut kedua vektor ini.

Dalam penelitian ini, akan diterapkan kedua sistem kategorisasi, yaitu pengkategorian positif negatif, dan pengkategorian 4 kategori yang diteliti oleh Panichella *et al* (2012) kedalam aplikasi Android, serta menyelesaikan permasalahan nilai TF-IDF yang tidak stabil pada dokumen yang lebih panjang, dimana aplikasi diharapkan dapat mengklasifikasikan sebuah *review* yang diberikan kepadanya dengan akurasi yang cukup tinggi. Diharapkan pada akhirnya, dapat diketahui apakah *Maximum TF-IDF* dapat meningkatkan hasil performa klasifikasi KNN terhadap kedua dataset.

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, rumusan masalah yang didapatkan adalah:

1. Seberapa besarkah perbandingan akurasi dari penerapan *Maximum TF-IDF* kedalam KNN *Euclidean Distance* dan KNN *Cosine Similarity* dalam mengklasifikasikan dataset positif negatif, apabila dibandingkan dengan *Raw TF-IDF*?
2. Seberapa besarkah perbandingan akurasi dari penerapan *Maximum TF-IDF* kedalam KNN *Euclidean Distance* dan KNN *Cosine Similarity* dalam mengklasifikasikan dataset Panichella, apabila dibandingkan dengan *Raw TF-IDF*?
3. Bagaimanakah hasil perbandingan performa *Maximum TF-IDF* kedalam KNN *Euclidean Distance* dan KNN *Cosine Similarity* terhadap dataset Panichella, apabila dibandingkan dengan dataset positif negatif?

1.3 Tujuan

Tujuan dari penelitian ini adalah perancangan dan penerapan metode pengklasifikasian teks untuk membantu pengembang aplikasi untuk memproses *review-review* yang sudah ada dengan nilai akurasi yang lebih tinggi. Selain itu, penelitian ini ingin mencari tahu, dapatkah metode metode yang disebutkan pada latar belakang dapat mengkategorikan *review* kedalam dataset yang dikemukakan oleh Panichella, sehingga tujuan secara umum dari penelitian ini adalah:

1. Mencari tahu apakah metode *Maximum Term Frequency-Inverse Document Frequency* dapat meningkatkan akurasi dari metode KNN, apabila digunakan untuk mengkategorikan dataset yang hanya memiliki 2 kelas, yaitu positif dan negatif.
2. Mencari tahu apakah metode *Maximum Term Frequency-Inverse Document Frequency* dapat meningkatkan akurasi dari metode KNN, apabila digunakan untuk mengkategorikan dataset yang hanya memiliki lebih dari 2 kelas, yaitu dataset yang dikemukakan oleh Panichella et al.
3. Mencari tahu apakah penggunaan *Maximum TF-IDF* dalam KNN *Euclidean Distance* dan KNN *Cosine Similarity* dapat meningkatkan performa klasifikasi teks terhadap kedua dataset.

1.4 Batasan masalah

Batasan masalah penelitian ini adalah:

1. Permasalahan yang diangkat hanya akan membahas klasifikasi per kata. Konteks dari kata dan makna tidak akan secara spesifik diperhatikan dalam proses klasifikasi.

2. KNN dan TF-IDF akan digunakan sebagai metode penyelesaian masalah. Metode lain tidak akan secara langsung diujikan, namun metode yang sama akan diujikan beberapa kali menggunakan berbagai macam nilai dan variabel.
3. *Review* yang akan digunakan adalah *review* berbahasa Inggris, yang didapat dari playstore, diambil secara manual.
4. Jumlah data yang akan digunakan merupakan 100 data untuk masing masing kelas sebagai data latih, dan 30 data untuk masing masing kelas sebagai data uji.
5. Pengujian hasil akan membandingkan nilai akurasi dan F-Measure dari setiap metode, apabila digunakan untuk setiap jenis dataset.
6. Variasi nilai K yang digunakan berjumlah 10 nilai K.

1.5 Sistematika pembahasan

1. Pendahuluan

Berisi penjelasan awal mengenai permasalahan dan metode yang dibawa untuk memecahkan masalah.

2. Tinjauan Pustaka

Berisi deskripsi dari permasalahan dan literatur pendukung. dari metode dan teori yang ingin digunakan dalam memecahkan masalah

3. Metodologi Penelitian

Berisi tahapan tahapan dalam pemecahan masalah, serta penjelasan lebih jelas mengenai metode dan teori yang ingin digunakan untuk menyelesaikan masalah.

4. Perancangan

Berisi pernyataan masalah yang lebih elaboratif daripada yang terdapat pada pendahuluan, serta perancangan dan kebutuhan yang diperlukan dalam implementasi TF-IDF dan KNN.

5. Implementasi

Berisi uraian langkah langkah, tabel tabel dan penjelasan proses penerapan perancangan kedalam sistem.

6. Pengujian dan Analisis

Berisi hasil pengujian yang sudah dilakukan, serta analisis yang menyertai hasil pengujian untuk menyelesaikan permasalahan yang diajukan di awal penelitian.

7. Penutup

Berisi kesimpulan dan harapan kedepan mengenai sistem yang sudah dibuat.

BAB 2 TINJAUAN PUSTAKA

2.1 Review

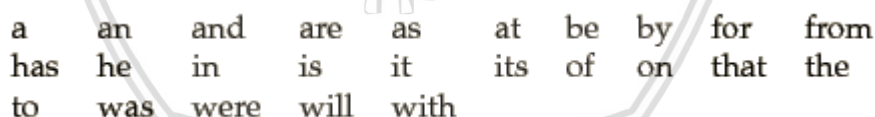
Review merupakan sarana komunikasi kepada pengembang oleh pengguna dimana pengguna dapat menyediakan informasi yang relevan untuk membantu perkembangan aplikasi dimana *review* itu diberikan (Panichella, 2012). *Review* dapat membantu aplikasi dalam proses pembenahan dan evolusinya, dimana *review* ini dapat memberikan masukan berupa fitur baru, pembetulan bug ataupun pengembangan fitur yang sudah tersedia di dalam aplikasi sedari awal.

2.2 Preprocessing

Preprocessing merupakan proses awal dalam proses klasifikasi, dimana *preprocessing* ini bertujuan untuk mengidentifikasi fitur dari teks yang paling penting dan membedakannya dari teks lain (Gaigole et al, 2013). Proses ini terbagi menjadi beberapa tahap yaitu *Filtering Stopwords*, *Stemming* dan *Term Weighting*.

2.2.1 Filtering Stopwords

Stopword merupakan daftar istilah yang sangat umum digunakan, sehingga istilah istilah ini hampir tidak mempunyai nilai dalam pengklasifikasian (Rajaraman, 2011). *Stopword* akan dibuat menjadi sebuah list dimana setiap istilah yang terdapat dalam *Stopword* akan secara otomatis dikecualikan dari proses, sehingga selain mempercepat proses KNN, diharapkan akurasi dari proses KNN juga meningkat. Salah satu contoh *Stopword* yang dapat menjadi referensi penelitian adalah *Stopword Reuters-RCV1* pada Gambar 2.1.



a an and are as at be by for from
has he in is it its of on that the
to was were will with

Gambar 2.1 StopWords Reuters-RCV1

2.2.2 Stemming

Stemming merupakan metode untuk mencari inti / kata dasar dari sebuah kata. *Stemming* merubah sebuah kata ke akarnya, yang secara langsung mempengaruhi hasil akhir dari proses klasifikasi menjadi lebih baik (Gaigole et al, 2013). Salah satu proses *Stemming* yang sering digunakan dalam bahasa inggris adalah *Porter*, dimana *Stemming* ini cukup efektif dalam mengembalikan kata berbahasa inggris ke dalam kata dasarnya. *Stemming Porter* ini sendiri bekerja dengan merubah sekumpulan hurub konsonan dan vokal menjadi satu simbol, kemudian memperhatikan pola yang dibuat. Dari pola ini, dapat ditentukan kata akhir yang harus diubah / ditiadakan.

2.2.3 Term Weighting

Term Weighting merupakan proses memberikan nilai terhadap sebuah istilah dalam dokumen. Proses pemberian nilai ini dapat dipengaruhi oleh jumlah dokumen, panjang dokumen, seringnya sebuah istilah muncul dan berbagai faktor lainnya (Gaigole et al, 2013). Dalam penelitian ini, proses *Term Weighting* merupakan salah satu proses yang diujikan, dikarenakan penelitian akan menerapkan modifikasi dari proses *Term Weighting* ini, yaitu proses *Term Frequency-Inverse Document Frequency* (TF-IDF).

Metode *Term Frequency – Inverse Document Frequency* (TF-IDF) ini merupakan sebuah metode untuk menentukan seberapa pentingnya sebuah kata terhadap sebuah dokumen, dan terhadap sebuah dokumen korpus. Metode ini bekerja dengan cara menghitung jumlah kata unik dalam satu dokumen, dan dibandingkan dengan jumlah total kata pada dokumen dimana metode ini diterapkan (Ramos et al, 2003). Proses TF-IDF, sesuai namanya, dibagi menjadi dua bagian yaitu proses *Term Frequency* (TF) dan proses *Inverse Document Frequency* (IDF).

2.2.3.1 Term Frequency

Proses TF merupakan proses pertama dalam metode TF-IDF. Proses ini dapat berdiri sendiri, dimana apabila digunakan sendiri, proses ini akan menghasilkan nilai berupa nilai kepentingan sebuah kata unik dalam suatu kalimat. Nilai yang dihasilkan sendiri dapat langsung digunakan, namun nilai relevansinya sangatlah rendah apabila digunakan dengan set data yang memiliki banyak dokumen yang berbeda.

TF memiliki banyak penyelesaian, dengan penyelesaian yang paling sederhana yaitu menggunakan *Raw Term Frequency* (*Raw TF*). *Raw TF* ini merupakan penyelesaian yang paling dasar, menghitung seberapa bobot sebuah kata dalam satu dokumen.

Salah satu permasalahan yang dihadapi oleh *Raw TF* ini adalah nilai TF yang dimiliki suatu dokumen menjadi lebih tinggi, hanya karena dokumen memiliki jumlah kata yang lebih banyak dan sering mengulang ulang sebuah kata tertentu. Hal ini dapat diselesaikan dengan menggunakan versi lain dari TF, yaitu *Maximum Term Frequency* (*Max TF*) (Manning et al, 2008). Kedua versi dari TF ini akan diujikan dalam penelitian ini.

Persamaan untuk *Raw TF* atau TF yang paling dasar adalah menggunakan Persamaan 2.1, dimana α merupakan banyaknya kata unik yang muncul di dokumen β .

$$tf(\alpha, \beta) = f_{\alpha, \beta} \quad (2.1)$$

Raw TF merupakan salah penyelesaian yang digunakan pada penelitian ini. Banyaknya dokumen yang akan dipakai, dimana setiap dokumen memiliki ukuran yang berbeda, membuat metode penyelesaian *Term Frequency* ini menjadi solusi yang cocok digunakan dalam kasus penelitian ini (Na, Kang dan Lee, 2015). *Raw*

TF menggunakan modifikasi dari Persamaan 2.1, yaitu Persamaan 2.2 untuk menyesuaikan rumus dengan panjang dokumen, dimana l adalah jumlah kata pada dokumen dimana $f_{\alpha,\beta}$ berasal.

$$tf(\alpha, \beta) = \frac{f_{\alpha,\beta}}{l} \quad (2.2)$$

Proses ini secara garis besar dikerjakan dengan melakukan perhitungan terhadap jumlah kata unik dalam satu dokumen, kemudian nilainya dibagi dengan jumlah kata pada dokumen yang sudah diproses, sehingga didapatkan nilai relevansi sebuah kata terhadap dokumen dimana metode ini diterapkan.

2.2.3.2 Inverse Document Frequency

Inverse Document Frequency (IDF) merupakan proses untuk memberikan relevansi pada nilai yang didapat dari proses TF terhadap dokumen korpus (Ramos et al, 2003). Proses ini dilakukan dengan mencari nilai relevansi dari sebuah kata unik terhadap jumlah total kata pada semua dokumen dalam sebuah dokumen korpus, lalu menggunakan nilainya untuk membagi nilai TF. Nilai TF-IDF yang tinggi dapat dilakukan dengan mencari kata yang banyak muncul di dalam suatu dokumen namun sedikit pada total semua dokumen. Proses ini bukanlah proses yang dapat berdiri sendiri, melainkan proses yang dapat membantu menambah akurasi dari proses lain. Penyelesaian IDF menggunakan Persamaan 2.3, dimana N merupakan jumlah total dokumen yang berada dalam dokumen korpus dan $|\{d \in D : t \in d\}|$ merupakan perhitungan jumlah dokumen dimana kata yang dicari muncul.

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad (2.3)$$

2.2.3.3 Maximum Term Frequency

Maximum Term Frequency (Max TF) adalah salah satu jenis dari TF yang diubah untuk menghindari terjadinya anomali dimana nilai F yang didapat oleh sebuah dokumen lebih tinggi daripada seharusnya, dikarenakan jumlah kata pada dokumen tersebut melebihi jumlah kata pada dokumen lain (Manning, Raghavan dan Schutze, 2008).

Persamaan yang digunakan dalam Max TF ini merupakan modifikasi dari Persamaan 2.1, dimana Persamaan 2.1 merupakan persamaan dasar untuk menyelesaikan *Term Frequency*. Setelah nilai TF ditemukan, nilainya akan dibagi oleh jumlah terbesar dari sebuah *Term* yang terdapat pada dokumen tersebut, dan kalikan dengan nilai *smoothing* dimana nilai *smoothing* ini digunakan untuk mengurangi kontribusi dari kata kedua. Persamaan yang dipakai adalah Persamaan 2.4

$$ntf_{t,d} = a + (1 - a) \frac{tf_{t,f}}{tf_{Max(d)'}} \quad (2.4)$$

Dimana a merupakan nilai *smoothing* yang memiliki nilai antara 0 hingga 1, dengan nilai yang biasa digunakan adalah 0.4 (Manning et al, 2008). *Smoothing* ini digunakan untuk mencegah terjadinya perubahan yang ekstrim pada nilai ntf apabila terjadi sedikit perubahan pada tf .

2.3 Pengklasifikasian Teks

Pengklasifikasian teks merupakan sebuah teknik untuk mengenali konteks sebuah teks dalam rangka memberikan label yang sudah ditentukan sebelumnya kepada tiap kumpulan teks (M. Krendzelak, 2015). Pengklasifikasian teks ini memiliki banyak fungsi dan banyak metode. Beberapa metode yang sering digunakan adalah Naive Bayes, K-means, Decision Tree Making dan K Nearest Neighbor. Masing masing metode memiliki kelebihan dan kelemahannya sendiri, baik dari sisi keakuratan, waktu proses, dimensi yang ideal untuk diproses dan juga beberapa faktor lainnya. Beberapa metode juga tidak dapat secara langsung memproses teks. Setiap variabel dari dokumen (baik tiap kata, maupun tiap huruf sesuai kebutuhan) harus diberi sebuah nilai agar metode dapat dilakukan. Dalam penelitian ini, metode yang digunakan adalah K-Nearest Neighbor. Alasan penggunaan K-Nearest Neighbor adalah akurasi, dimana KNN memiliki nilai akurasi yang cukup tinggi, namun dengan waktu pemrosesan yang relatif lebih panjang.

2.3.1 K-Nearest Neighbor

K-Nearest Neighbor (KNN) adalah sebuah metode klasifikasi terhadap objek dengan cara membandingkan dengan posisi objek yang diletakan di sebuah dimensi melalui perhitungan nilainya dengan objek objek yang sebelumnya sudah dikenalkan ke perangkat. Setelah objek diletakan, dicari objek lain dengan jarak paling dekat dengan objek ini dengan jumlah yang sebelumnya sudah ditentukan, dan diklasifikasikan.

KNN merupakan salah satu *supervised learning* yang cukup populer. *Supervised learning* merupakan metode pembelajaran mesin, dimana data yang akan digunakan untuk pelatihan, harus sebelumnya dipisah pisahkan menurut kelasnya masing masing (Rai, 2011). Skenario yang terjadi pada *supervised learning* adalah pengelompokan dataset pengujian kedalam kelas-kelas yang sudah dibagikan pada dataset pelatihan. KNN sendiri memiliki nilai akurasi yang cukup tinggi dalam kategori *Supervised Learning* ini.

KNN bekerja dengan cara menyimpan sekumpulan dataset pelatihan (Quiros, et al, 2017). Dataset yang akan dilatih harus sudah memiliki nilai, sehingga apabila dataset yang diberikan berupa sekumpulan huruf atau data, dataset yang sudah bernilai ini sebelumnya harus diubah menjadi sekumpulan nilai dengan parameter tertentu, sebelum disimpan kedalam sistem. Nilai dari setiap data inilah yang nantinya akan diperhitungkan dalam pengklasifikasian teks.

Setelah data disimpan, dataset pengujian akan diujikan kedalam sistem. Dataset pengujian ini juga harus diberi nilai untuk setiap atributnya agar dapat diproses oleh metode ini. Ada beberapa metode yang dapat diterapkan dalam

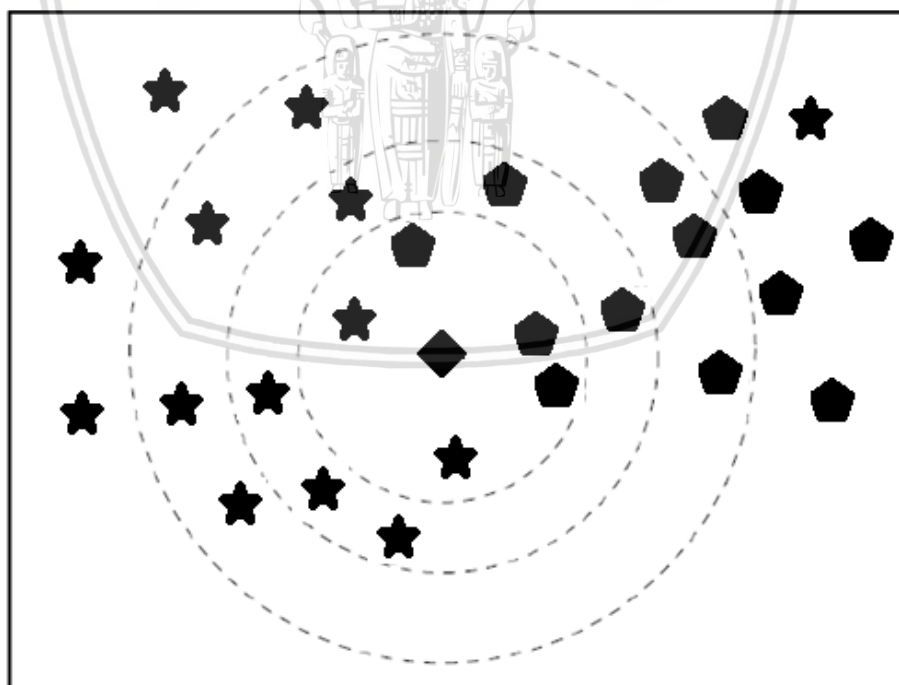
mengukur kedekatan KNN, beberapa diantaranya adalah *Cosine Similarity* dan *Euclidean Distance* (Qamar et al, 2008). Metode *Euclidean Distance* menggunakan rumus yang tertera pada Persamaan 2.5 (Mulak, 2013), dimana i bernilai 1 dan n merupakan jumlah *term* terbanyak diantara data latih ataupun data uji. x dan u merepresentasikan nilai dari *term* yang diperhitungkan.

$$d_{euclidean}(x, u) = \sqrt{\sum_{i=1}^n (x^{(i)} - u^{(i)})^2} \quad (2.5)$$

Sedangkan *Cosine Similarity* menggunakan rumus yang tertera pada Persamaan 2.6 (Lahitani et al, 2016), dengan i bernilai 1 dan n merupakan jumlah *term* terbanyak pada masing masing data.

$$\text{cosine similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (2.6)$$

Kedua metode akan diujikan menggunakan *Maximum Term Frequency – Inverse Document Frequency (Max TF-IDF)* untuk dijadikan perbandingan. Setelah metode-metode pada Persamaan 2.5 dan Persamaan 2.6 dilakukan, data yang sedang diuji akan diukur kedekatannya dengan semua data dalam dataset pelatihan dengan sejumlah data lain yang paling dekat, yang jumlah datanya sudah ditentukan di awal. Jumlah data yang akan dijadikan acuan ini biasa disebut sebagai variabel K , dimana banyak sedikitnya variabel ini akan berpengaruh terhadap keakuratan metode KNN. Beberapa contoh nilai K dapat dilihat pada Gambar 2.2.



Gambar 2.2 Contoh KNN dengan nilai K yang berbeda beda

Setiap simbol merepresentasikan salah satu jumlah K yang diujikan dalam sistem. Selain keakuratan, nilai K juga mempengaruhi kecepatan sistem dalam memproses data.

2.3.2 KNN + TF-IDF

Proses implementasi TF-IDF mungkin berbeda beda untuk setiap metode, namun secara garis besar, TF-IDF bekerja menggunakan Persamaan 2.7.

$$w_d = f_{w,d} * \log(|D|/f_{w,D}) \quad (2.7)$$

dimana $f_{w,d}$ adalah jumlah seberapa banyak variabel w muncul dalam d , $|D|$ merupakan ukuran dari korpus, dan $f_{w,D}$ merupakan jumlah dimana w muncul dalam D (Juan Ramos, 2003).

Nilai TF-IDF akan disimpan, kemudian digunakan sebagai referensi pada saat pengecekan KNN. Jumlah K akan diambil senilai 10 angka ganjil disekitar nilai umum yang biasa digunakan sebagai K , yang tertera pada Persamaan 2.8 (Hassanat et al, 2014).

$$K = n^{1/2} \quad (2.8)$$

Dimana n merupakan jumlah instansi data yang digunakan.

2.3.3 F-Measure

F-Measure merupakan nilai yang merepresentasikan hasil paling optimal dari kombinasi Precision dan Recall, dimana semakin tinggi nilai F-Measure ini, semakin optimal pula hasil Precision dan Recall yang didapat.

Precision merupakan salah satu nilai yang merepresentasikan hasil dari pembagian nilai *True Positive* terhadap hasil penjumlahan *True Positive* dan *False Positive*, seperti pada Persamaan 2.9.

$$precision = \frac{Tp}{Tp+Fp} \quad (2.9)$$

Precision ini mengekspresikan nilai kebenaran relevansi hasil pengujian suatu sistem dibandingkan dengan semua nilai yang dinilai benar oleh sistem (Powers, 2011).

Selain Precision, penilaian lain dari merupakan Recall. Recall menilai perbandingan antara data yang sesuai dengan kelas yang sedang diujikan, dibandingkan dengan semua data yang seharusnya bernilai benar (Powers, 2011). Recall dapat dirumuskan Pada persamaan 2.10.

$$recall = \frac{Tp}{Tp+Fn} \quad (2.10)$$

Hasil dari Precision dan Recall memiliki nilai antara 0 – 1. Nilai 1 pada masing masing metode ini sendiri pada dasarnya mudah didapat. Apabila sistem menilai hanya satu data sebagai benar, dan menilai semua sisa data sebagai salah, maka akan didapatkan nilai Precision yang sempurna (1). Nilai Precision yang tinggi ini tidak merepresentasikan model klasifikasi yang sempurna, karena nilai Recall akan menurun, disebabkan banyaknya *False Negative* (Koehrsen, 2018). Sedangkan untuk Recall, apabila semua data dimasukkan ke dalam penilaian

benar, maka akan didapat nilai Recall yang sempurna (1), dengan mengorbankan Precision (Koehrsen, 2018).

Untuk beberapa situasi, sistem mungkin diharapkan untuk memiliki nilai Recall yang lebih tinggi daripada Precision, atau sebaliknya, mengorbankan Recall untuk mencapai Precision yang lebih tinggi. Namun untuk klasifikasi dimana tidak ada penilaian khusus yang dicari, maka hasil paling optimal dari kedua penilaian ini didapat dengan menghitung nilai F-Measure (Koehrsen, 2018).

Rumus untuk permasalahan F-Measure dapat dilihat pada Persamaan 2.11

$$F1 = 2 * \frac{precision*recall}{precision+recall} \quad (2.11)$$

F-Measure sendiri bukanlah penilaian akurasi yang sempurna, terdapat permasalahan permasalahan (Powers, 2015) yaitu :

- F-Measure hanya berfokus pada 1 kelas
- F-Measure memiliki bias terhadap kelas dengan jumlah data terbanyak
- F-Measure sama sekali tidak memperhitungkan nilai *True Negative*, sehingga nilai *True Negative* dapat bertambah sangat banyak dan nilai F-Measure tidak berubah

Poin kedua tidak akan menjadi permasalahan dalam penelitian ini, dikarenakan jumlah dataset yang sudah terbagi secara merata untuk setiap kelas. Namun bagaimana dengan 2 poin lainnya?

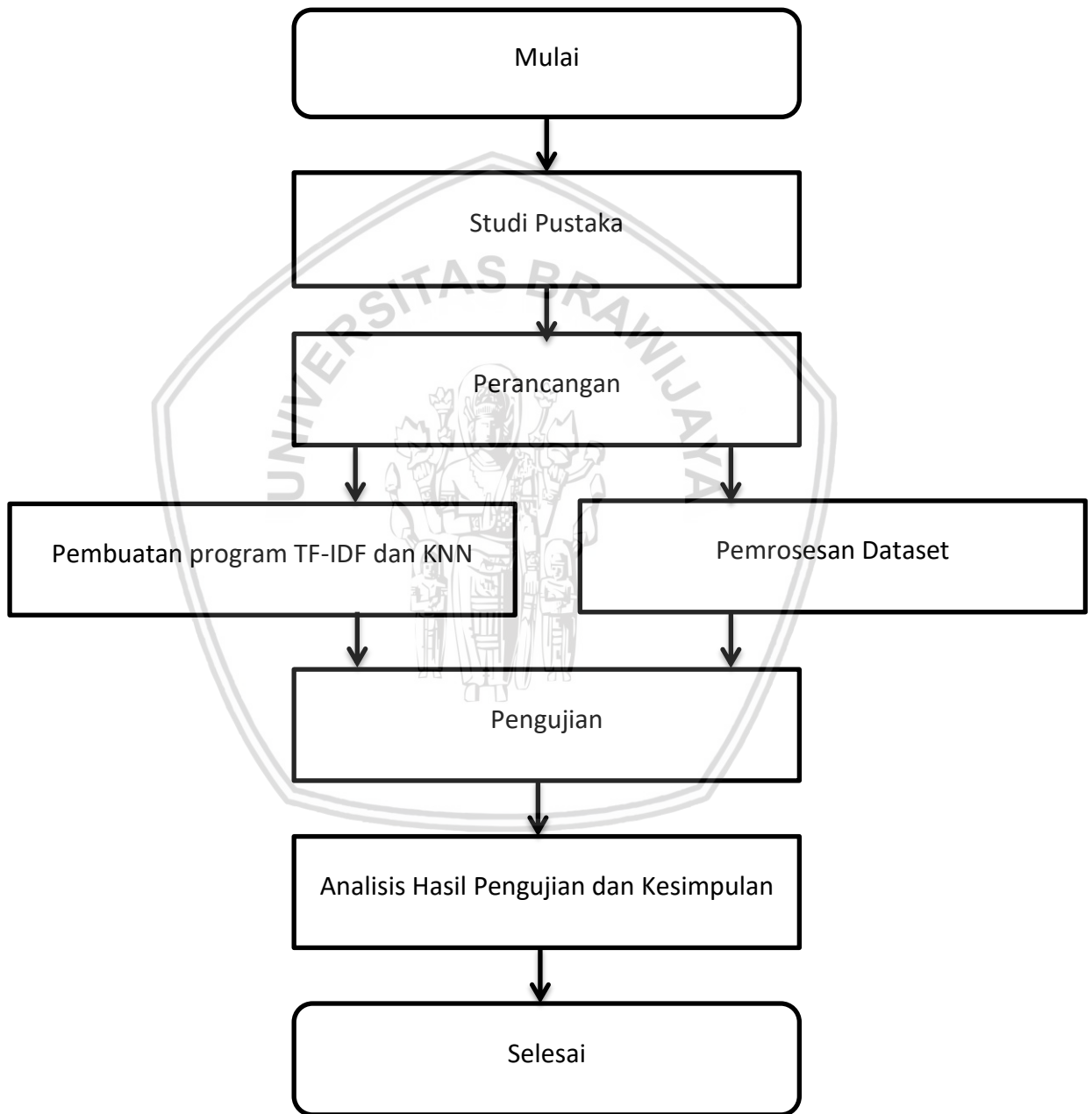
Permasalahan dalam poin pertama dan ketiga dapat diselesaikan dengan mencari F-Measure untuk setiap kelas (Powers, 2015). Dengan mencari F-Measure untuk setiap kelas, nilai *True Negative* akan berpengaruh terhadap rata rata F-Measure yang didapat. Pencarian rata rata menggunakan *Macro Average*, dengan alasan bahwa *Macro Average* memberikan nilai rata rata yang lebih baik apabila tidak terdapat permasalahan dengan perbedaan jumlah dataset pada tiap kelas (Ozgun et al, 2005)

2.3.4 Akurasi

Akurasi merupakan salah satu metode penilaian untuk menilai performa metode lain. Akurasi merupakan salah satu metode yang paling mudah dan sering digunakan, dimana perhitungannya merupakan hasil pembagian jumlah data yang benar dibagi dengan jumlah total data (Baratloo et al, 2015).

BAB 3 METODOLOGI PENELITIAN

Penelitian ini merupakan penelitian bertipe analitik. Diperlukan pengumpulan data dan implementasi pada aplikasi menggunakan metode KNN dan TF-IDF, dimana akhirnya akan dibandingkan menggunakan jumlah dataset yang berbeda. Metodologi penelitian ini dideskripsikan pada Gambar 3.1.



Gambar 3.1 Metodologi

3.1 Studi Pustaka

Proses Studi Pustaka diperlukan dalam penelitian ini, dikarenakan diperlukannya pengetahuan untuk memperkuat landasan teori, serta data data yang akan digunakan selama pemrosesan. Proses Studi Pustaka akan dilakukan melalui berbagai medium, seperti buku, jurnal, paper maupun internet. Ada beberapa topik yang perlu dipelajari pada proses ini, yaitu :

1. Klasifikasi Teks

Konsep dari Klasifikasi Teks ini sendiri akan dipelajari lebih lagi, baik mengenai jenis jenisnya, tipe tipenya, penerapan dan poin poin lain. Diharapkan dengan dipelajarinya konsep ini, tidak terjadi kesalahan fatal mengenai pengertian pengertian yang dipakai.

2. *K-Nearest Neighbor (KNN)*

KNN perlu dipelajari untuk memperdalam pengertian tentang metode klasifikasi ini, agar saat penerapannya tidak terjadi kesalahan.

3. *Term Frequency – Inverse Document Frequency (TF-IDF)*

TF-IDF perlu dipelajari lebih lanjut, agar pengaplikasiannya dan penghubungannya dengan KNN tidak terjadi kesalahan, baik data yang diterima maupun yang diberikan.

Dalam proses ini, diharapkan pengetahuan yang didapat cukup banyak sehingga proses proses selanjutnya dapat berjalan dengan lancar.

3.2 Analisis dan Perancangan

Analisis dan Perancangan merupakan proses yang dilakukan sebelum mengaplikasikan pengetahuan yang didapat sebelumnya kedalam kode. Proses ini meliputi proses pencarian kebutuhan kebutuhan yang diperlukan aplikasi, kemudian membuat diagram diagram yang akan menjadi dasar dari aplikasi yang akan dibuat.

Proses Analisis dan Perancangan dilakukan untuk mempersiapkan dasar dasar pembuatan aplikasi, agar aplikasi yang dibuat tidak keluar dari batasan rancangan yang sudah dibuat. Hasil akhir dari proses ini adalah berbagai macam diagram yang menjadi patokan dalam pembuatan aplikasi.

3.3 Pembuatan Program TF-IDF dan KNN

Menggunakan pengetahuan yang sudah didapat dari proses Studi Pustaka dan desain yang didapat dari proses Perancangan, program akan dibuat menggunakan Android Studio dengan bahasa pemrograman Java. Pengerjaan akan dilakukan sembari terus mempelajari cara kerja TF-IDF dan KNN, agar mendapat hasil yang optimal saat program selesai dibuat.

Proses ini juga mencakup pembuatan UI kasar. UI dibuat hanya semata untuk mengakses fitur fitur yang tersedia, dan keindahan UI bukan poin yang akan

sangat diperhatikan. Pembuatan UI akan sepenuhnya berfokus pada fungsionalitas.

Hasil yang diharapkan dari tahapan ini adalah aplikasi Android yang dapat memproses sebuah *review* menggunakan metode KNN dan TF-IDF, untuk mengkategorikannya kedalam kelas kelas yang sudah ditentukan sebelumnya.

3.4 Pemrosesan Dataset

Proses ini berjalan seiring dengan Proses 3.3. Pada proses ini, dataset berupa *review* akan diambil dari aplikasi populer yang tersedia secara global di PlayStore, yaitu Steam dan Facebook, kemudian sekumpulan dataset yang sudah diambil ini akan diproses melalui beberapa langkah *preprocessing* untuk membuat dataset yang tersedia menjadi lebih akurat dan mudah diproses oleh sistem.

Melalui proses ini, akan didapatkan dataset berupa sejumlah *review* yang siap untuk diproses aplikasi, sebagai bahan pengujian.

Proses yang akan dilakukan dalam tahapan ini meliputi:

3.4.1 Review Extracting

Proses pertama merupakan proses pengambilan *review*. Pengambilan *review* akan dilakukan secara manual, termasuk didalamnya proses pengkategorian. Dalam proses ini, *review* yang diambil juga akan dikategorikan kedalam 2 kategori umum, yang masing masing berisi 2 dan 4 kelas. Kategori pertama, dimana *review* dibagi menjadi 2, yaitu:

a. Positive

Review positif merupakan semua *review* yang memuji, menunjukkan kelebihan, serta hal hal yang mendukung dan menunjukkan support terhadap developer mengenai arah yang dituju aplikasi.

b. Negative

Review negatif merupakan *review* yang menjelek jelekkan dan atau tidak menunjukkan support terhadap arah yang dituju oleh perkembangan aplikasi.

Pada kategori *review* kedua, *review* dibagi menjadi 4 kelas menurut penelitian sebelumnya (Panichella et al, 2015), yaitu:

a. Information Giving

Review yang memberikan informasi atau perkembangan mengenai aplikasi atau pengguna.

b. Information Seeking

Review yang berhubungan tentang suatu usaha untuk mendapatkan informasi dari pengguna lain / pengembang.

c. Feature Request

Review yang memberikan saran atau permintaan berupa tambahan fitur atau pengembangan fitur yang sudah tersedia.

d. Problem Discovery

Review yang menjelaskan tentang permasalahan yang dihadapi oleh aplikasi, atau kesalahan yang tidak diharapkan.

3.4.2 Filtering *Stopword*

Proses kedua adalah proses penghapusan kata kata yang termasuk dalam daftar *Stopword*. Hal ini dilakukan untuk menghilangkan kata-kata yang ulang yang tidak bernilai dalam proses klasifikasi.

Stopword yang akan digunakan adalah *Stopword* bahasa inggris yang tersedia pada website NLTK, dimana website NLTK merupakan salah satu website yang dipercaya untuk melakukan penelitian mengenai *Natural Language Processing* (Bird, S dan Loper, E, 2002).

3.4.3 *Stemming*

Stemming merupakan proses perubahan kata-kata menjadi kata dasar. Proses ini dilakukan juga untuk meningkatnya akurasi, dikarenakan tanpa adanya proses ini, jumlah variabel yang terlibat akan meningkat dengan cukup signifikan, mengingat banyaknya jumlah kata apabila setiap jenis kata dijadikan variable masing masing.

Proses *Stemming* yang akan dilakukan adalah *Stemming* Porter (Porter, 1980), dikarenakan *Stemming* Porter merupakan salah satu proses *Stemming* yang paling sering digunakan dan memiliki akurasi yang cukup besar.

3.5 Pengujian dan Analisis

3.5.1 Perubahan Dataset menjadi nilai melalui TF-IDF

Proses ini merupakan langkah pertama setelah aplikasi selesai dibuat. Pada langkah ini, dataset pelatihan yang sudah diklasifikasikan akan dimasukan kedalam sistem TF-IDF yang sudah dibuat. Proses ini mungkin akan memakan waktu yang cukup lama, karena banyaknya jumlah data yang akan diproses. Hasil proses yang diharapkan adalah sebuah nilai dalam bentuk angka, dan / atau matrix yang berisi nilai nilai angka yang telah diproses.

Semua klasifikasi kelas akan dimasukan kedalam sistem TF-IDF ini untuk diubah menjadi nilai. Proses akan dilakukan secara serentak apabila perangkat memadai, diukur saat percobaan pertama. Apabila perangkat terlihat sangat terbebani oleh perubahan data 4 kelas sekaligus, perubahan data akan dilakukan satu persatu per kelas, hingga 4 kali perubahan.

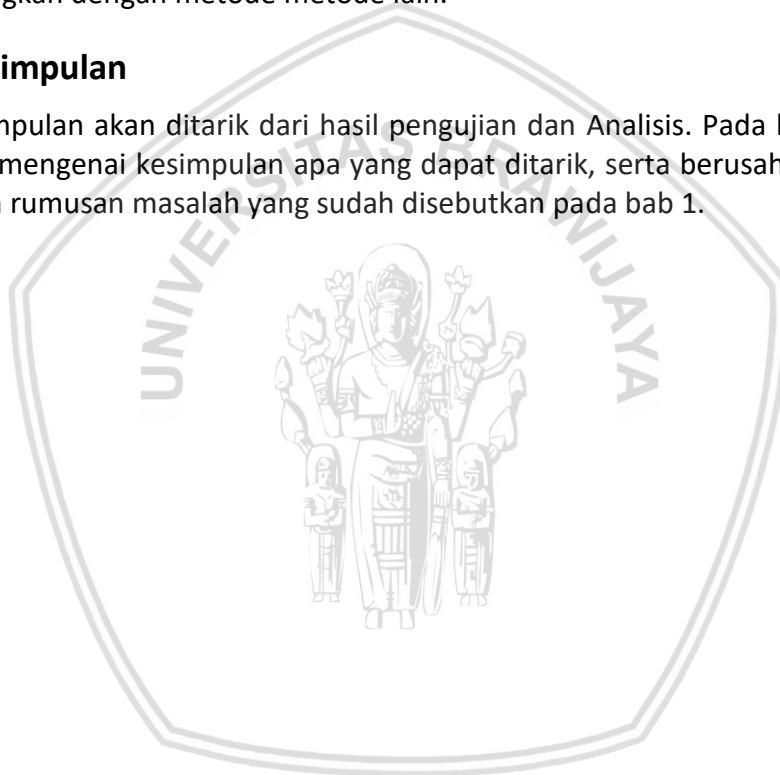
3.5.2 Pengujian menggunakan dataset testing

Pengujian dilakukan setelah semua dataset sudah dimasukkan kedalam KNN. Dataset yang tersedia untuk pengujian berbeda dengan dataset pelatihan, dimana kelas juga sudah ditentukan secara manual, namun tidak menjadi faktor penilaian saat pengujian.

Proses ini akan memakan waktu yang lama, dikarenakan perhitungan akan dilakukan terhadap setiap dokumen. Disini juga akan dilakukan pengujian terhadap performa aplikasi, dengan jumlah dataset pelatihan tertentu dan dibandingkan dengan durasi yang diperlukan aplikasi yang sama, dengan jumlah dataset yang berbeda. Penilaian lain, yaitu akurasi juga akan dilakukan setelah selesai pengujian, dimana akan dilihat jumlah kesalahan yang ada dan dibandingkan dengan metode metode lain.

3.6 Kesimpulan

Kesimpulan akan ditarik dari hasil pengujian dan Analisis. Pada bab ini, akan dibahas mengenai kesimpulan apa yang dapat ditarik, serta berusaha menjawab rumusan rumusan masalah yang sudah disebutkan pada bab 1.



BAB 4 PERANCANGAN

4.1 Persiapan Dataset

4.1.1 Penjelasan dan Contoh

Terdapat 2 dataset yang dipersiapkan dalam perancangan ini. Dataset pertama merupakan dataset positif negatif. Dataset diambil secara manual dari *Playstore*, aplikasi *Steam* dan *Facebook*. Dataset ini berjumlah 130 per kelasnya, sehingga total data yang dibutuhkan adalah 260 untuk 2 kelas, yaitu positif dan negatif. Setelah 130 dataset diambil per kelas, penguji melakukan penilaian secara manual untuk masing masing data. Tidak terjadi perubahan dari 260 total data yang diambil secara acak. Dataset positif negatif ini akan ditampilkan pada Tabel 4.1.

Tabel 4.1 Deskripsi Kategori Review Dataset Positif Negatif

Nama Kategori	Penjelasan
Positif	<p>Kategori <i>review</i> positif mencakup semua <i>review</i> yang menyatakan persetujuan dan dukungan terhadap konsep, ide dan perlakuan yang dilakukan oleh pengembang.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>Good one! This is one of the best apps out there, must have</i> • <i>This is an awesome social media!</i>
Negatif	<p>Kategori <i>review</i> negatif merupakan kategori <i>review</i> yang berlawanan dengan kategori <i>review</i> positif. Didalam kategori ini, <i>user</i> mengekspresikan pendapat atau ide yang berlawanan dengan kondisi aplikasi pada saat <i>review</i> dibuat.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>This one really bad. How does creating new account work? They shouldn't bother with this if they gonna be half-assed about that</i> • <i>Hate no password recovery for my country hate hate hate 2go now</i>

Dataset kedua merupakan dataset Panichella. Dataset diambil dari *Playstore* secara manual, dengan pengambilan data sebesar 520 dokumen yang dibagi menjadi 400 data latih dan 120 data pengujian. *Review* diurutkan berdasarkan waktu, agar data yang diambil saling relevan satu dengan yang lain. 400 data tersebut dikategorikan kedalam 4 kategori yang diajukan pada penelitian

sebelumnya mengenai teknik improvisasi aplikasi (Panichella et al, 2015). Keempat kategori tersebut akan dijelaskan pada Tabel 4.2.

Tabel 4.2 Deskripsi Kategori *Review* Multiclass Dataset Panichella

Nama Kategori	Penjelasan
<i>Information Giving</i>	<p>Kategori <i>Review</i> yang mencakup kalimat yang menginformasikan atau memberi masukan mengenai kondisi aplikasi dari pengalaman yang dialami user pada waktu tertentu.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>Have to say it's super useful!</i> • <i>Got helped with this app</i>
<i>Information Seeking</i>	<p>Kategori <i>Review</i> yang mencakup kalimat yang menunjukkan usaha dalam mencari jawaban atau informasi dari pengguna lain / pengembang.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>How does this work?</i> • <i>Can someone help me with picking a video? I know you can do it but i don't know how or where to do</i>
<i>Feature Request</i>	<p>Kategori <i>Review</i> berupa kalimat yang mengekspresikan ide, saran atau pendapat mengenai kebutuhan fitur yang ingin ditambahkan kedalam aplikasi oleh user, untuk mempermudah penggunaan atau mengembangkan fitur fitur yang kurang dikembangkan.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>U make the login wayyyy more difficult than it needs to be it would be easy if it was more like discord or google</i> • <i>Please add a feature so i can redeem steam codes?</i>
<i>Problem Discovery</i>	<p>Kategori <i>Review</i> yang menjabarkan permasalahan yang sedang dialami user, yang berhubungan tentang kurang / cacatnya aplikasi.</p> <p>Contoh:</p> <ul style="list-style-type: none"> • <i>After update i can't view my wishlist anymore it always keep loading and not showing anything just background and search filter)</i>

	<ul style="list-style-type: none"> • <i>Sometimes some messages in the chat does not display, and updating your privacy settings is not functional</i>
--	---

Pengkategorian dilakukan secara manual oleh penguji, dengan jumlah 130 *review* per kategori, dengan pembagian 100 dataset pelatihan dan 30 dataset pengujian per kategori. Pembagian *review* secara merata dilakukan untuk mencegah jumlah dokumen mempengaruhi proses pengkategorian.

Dataset pengujian diambil dari total dataset secara acak, sebelum proses pembagian kedalam tahap dilakukan sebanyak 30 per kategori, 120 total.

Kedua dataset kemudian akan melalui proses NLTK *Stopword* dan *Stemming* Porter (Porter, 1980). Proses ini ditujukan untuk menghilangkan residu dari huruf-huruf tambahan yang dituliskan oleh pengguna untuk kata yang sama, dengan cara yang berbeda. Setiap kata akan dikembalikan ke kata dasarnya, sehingga cara menulis pengguna tidak mengganggu proses kategorisasi. Setelah selesai, proses selanjutnya adalah memisahkan kata menggunakan spasi, dan memasukan dataset pelatihan kedalam sebuah List.

List yang dibuat memiliki 2 tingkatan, dimana tingkatan dalam mewakili tiap kata dan tingkatan mewakili jumlah dokumen. Tiap kategori akan dimasukan kedalam List yang berbeda, sehingga akan terbuat 4 dan 2 List yang mewakili tiap Kategori. Dataset pengujian tidak mengalami hal ini, dimana nanti tiap dokumen dari dataset pengujian akan dimasukan secara manual kedalam aplikasi untuk melihat hasil prosesnya dan hasil tersebut akan dicatat.

4.1.2 Pola Kategori Panichella

Dari beberapa contoh yang disediakan, dapat dikenali beberapa pola yang muncul pada pengkategorian Panichella. Pola ini dapat digunakan untuk mengenali sebuah *review* secara lebih mendalam, apabila terjadi kesulitan dalam mengkategorikan sebuah *review* secara manual. Beberapa pola yang didapat dari kategorisasi Panichella akan dijelaskan pada Tabel 4.3.

Tabel 4.3 Deskripsi Kategori *Review* Multiclass Dataset Panichella

Nama Kategori	Penjelasan
<i>Information Giving</i>	<i>Information Giving</i> memiliki pola dimana sebagian besar dari <i>review</i> yang masuk kedalam kategori ini memiliki pembawaan yang positif. Hal ini terjadi karena saat pengguna memberikan informasi yang negatif kepada sebuah aplikasi, sebagian besar akan diikuti oleh permasalahan yang muncul pada aplikasi tersebut, sehingga dapat dikategorikan kedalam kategori <i>Problem Discovery</i> . <i>Information Giving</i> juga memiliki jumlah kata yang relatif lebih sedikit apabila dibandingkan dengan kategori yang lain, dengan beberapa <i>review</i> hanya



	memiliki 3 - 4 kata.
<i>Information Seeking</i>	<i>Information Seeking</i> memiliki pola dimana sebagian besar <i>review</i> yang termasuk dalam kategori ini berupa pertanyaan. <i>Review</i> ini muncul ketika pengguna yang menanyakan tidak tahu mengenai apa yang harus dilakukan. Terkadang mirip dengan <i>Problem Discovery</i> , namun seringkali tidak menyebutkan permasalahan dimiliki oleh aplikasinya sendiri.
<i>Feature Request</i>	<i>Feature Request</i> memiliki pola dimana pengguna memiliki hal spesifik yang ingin ditambahkan kedalam aplikasi. Kategori <i>review</i> kadang tercampur dengan kategori lain seperti <i>Information Seeking</i> atau <i>Problem Discovery</i> , tapi lebih menekankan saran atau permintaan fitur yang diharapkan pengguna. <i>Review</i> yang termasuk dalam kategori ini juga sering menggunakan kata kata khusus seperti <i>add</i> dan <i>feature</i> .
<i>Problem Discovery</i>	<i>Problem Discovery</i> memiliki pola dimana pengguna menunjukkan kekurangan yang spesifik dari sebuah aplikasi. Berkebalikan dengan <i>Information Seeking</i> , kategori <i>review</i> ini biasanya dibuat oleh pengguna yang memiliki pengetahuan lebih mengenai aplikasi atau perangkat, sehingga <i>review</i> dapat mencakup permasalahan inti yang dimiliki oleh aplikasi, dan bukan hanya sekedar menanyakan solusi untuk menyelesaikan permasalahan tersebut.

4.2 Daftar Kebutuhan

4.2.1 Kebutuhan Fungsional

No	Kode Fungsi	Kebutuhan Fungsional	Spesifikasi Kebutuhan	Use Case
1	SRS_TF K_F_01	Pengguna harus mampu melakukan proses pengkategorian <i>review</i> menggunakan <i>Raw TF-IDF</i> dan <i>KNN Euclidean Distance</i>	<ol style="list-style-type: none"> 1. Sistem harus mampu merubah data yang disediakan menjadi nilai <i>Raw TF-IDF</i> yang diharapkan 2. Sistem harus mampu memanfaatkan nilai <i>Raw TF-IDF</i> yang sudah 	Mengklasifikasi <i>Raw TF-IDF Euclidean</i>

			<p>didapatkan dalam proses pencarian nilai <i>Euclidean Distance</i></p> <p>3. Sistem harus mampu menampilkan termasuk pada kategori manakah <i>review</i> termasuk</p>	
2	SRS_TF K_F_02	Pengguna harus mampu melakukan proses pengkategorian <i>review</i> menggunakan <i>Raw TF-IDF</i> dan KNN <i>Euclidean Distance</i>	<p>1. Sistem harus mampu merubah data yang disediakan menjadi nilai <i>Raw TF-IDF</i> yang diharapkan</p> <p>2. Sistem harus mampu memanfaatkan nilai <i>Raw TF-IDF</i> yang sudah didapatkan dalam proses pencarian nilai <i>Cosine Similarity</i></p> <p>3. Sistem harus mampu menampilkan termasuk pada kategori manakah <i>review</i> termasuk</p>	Mengklasifikasi <i>Raw TF-IDF Cosine</i>
3	SRS_TF K_F_03	Pengguna harus mampu melakukan proses pengkategorian <i>review</i> menggunakan <i>Max TF-IDF</i> dan KNN <i>Euclidean Distance</i>	<p>1. Sistem harus mampu merubah data yang disediakan menjadi nilai <i>Max TF-IDF</i> yang diharapkan</p> <p>2. Sistem harus mampu memanfaatkan nilai <i>Raw TF-IDF</i> yang sudah didapatkan dalam proses pencarian nilai <i>Euclidean Distance</i></p> <p>3. Sistem harus mampu menampilkan termasuk pada kategori manakah <i>review</i> termasuk</p>	Mengklasifikasi <i>Max TF-IDF Euclidean</i>
4	SRS_TF K_F_04	Pengguna harus mampu melakukan	1. Sistem harus mampu merubah data yang	Mengklasifikasi <i>Max TF-</i>



	<p>1. <i>Actor</i> memasukan teks review yang ingin dikategorikan</p> <p>2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian</p>	
		<p>3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai <i>Raw TF-IDF</i></p> <p>4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode <i>KNN Euclidean Distance</i></p> <p>5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk</p>
<i>Alternative Flow</i>	1. Jika field teks belum terisi dan tombol ditekan, akan muncul notifikasi meminta field untuk diisi terlebih dahulu	
<i>Post – Condition</i>	<i>Actor</i> mendapatkan kategori dimana <i>review</i> yang baru saja dimasukan termasuk	

4.2.3.2 Mengklasifikasikan Raw TF-IDF Cosine

<i>Flow of Events</i> untuk <i>Raw TF-IDF Cosine</i>		
<i>Objective</i>	Untuk melakukan proses pengkategorian menggunakan <i>Raw TF-IDF</i> dan <i>KNN Cosine Similarity</i>	
<i>Actor</i>	<i>User</i>	
<i>Pre – Condition</i>	<i>Actor</i> telah berada di halaman awal aplikasi	
<i>Main Flow</i>	<i>Actor</i>	<i>System</i>
	<p>1. <i>Actor</i> memasukan teks review yang ingin dikategorikan</p> <p>2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian</p>	
		3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai



		<p><i>Raw TF-IDF</i></p> <p>4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode <i>KNN Cosine Similarity</i></p> <p>5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk</p>
<i>Alternative Flow</i>	1. Jika field teks belum terisi dan tombol ditekan, akan muncul notifikasi meminta field untuk diisi terlebih dahulu	
<i>Post – Condition</i>	<i>Actor</i> mendapatkan kategori dimana <i>review</i> yang baru saja dimasukan termasuk	

4.2.3.3 Mengklasifikasikan Max TF-IDF Euclidean

<i>Flow of Events</i> untuk <i>Max TF-IDF Euclidean</i>		
<i>Objective</i>	Untuk melakukan proses pengkategorian menggunakan <i>Max TF-IDF</i> dan <i>KNN Euclidean Distance</i>	
<i>Actor</i>	<i>User</i>	
<i>Pre – Condition</i>	<i>Actor</i> telah berada di halaman awal aplikasi	
<i>Main Flow</i>	<i>Actor</i>	<i>System</i>
	<p>1. <i>Actor</i> memasukan teks <i>review</i> yang ingin dikategorikan</p> <p>2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian</p>	<p>3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai <i>Max TF-IDF</i></p> <p>4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode <i>KNN Euclidean Distance</i></p> <p>5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk</p>
<i>Alternative Flow</i>	1. Jika field teks belum terisi dan tombol ditekan, akan	



	muncul notifikasi meminta field untuk diisi terlebih dahulu
<i>Post – Condition</i>	<i>Actor</i> mendapatkan kategori dimana <i>review</i> yang baru saja dimasukan termasuk

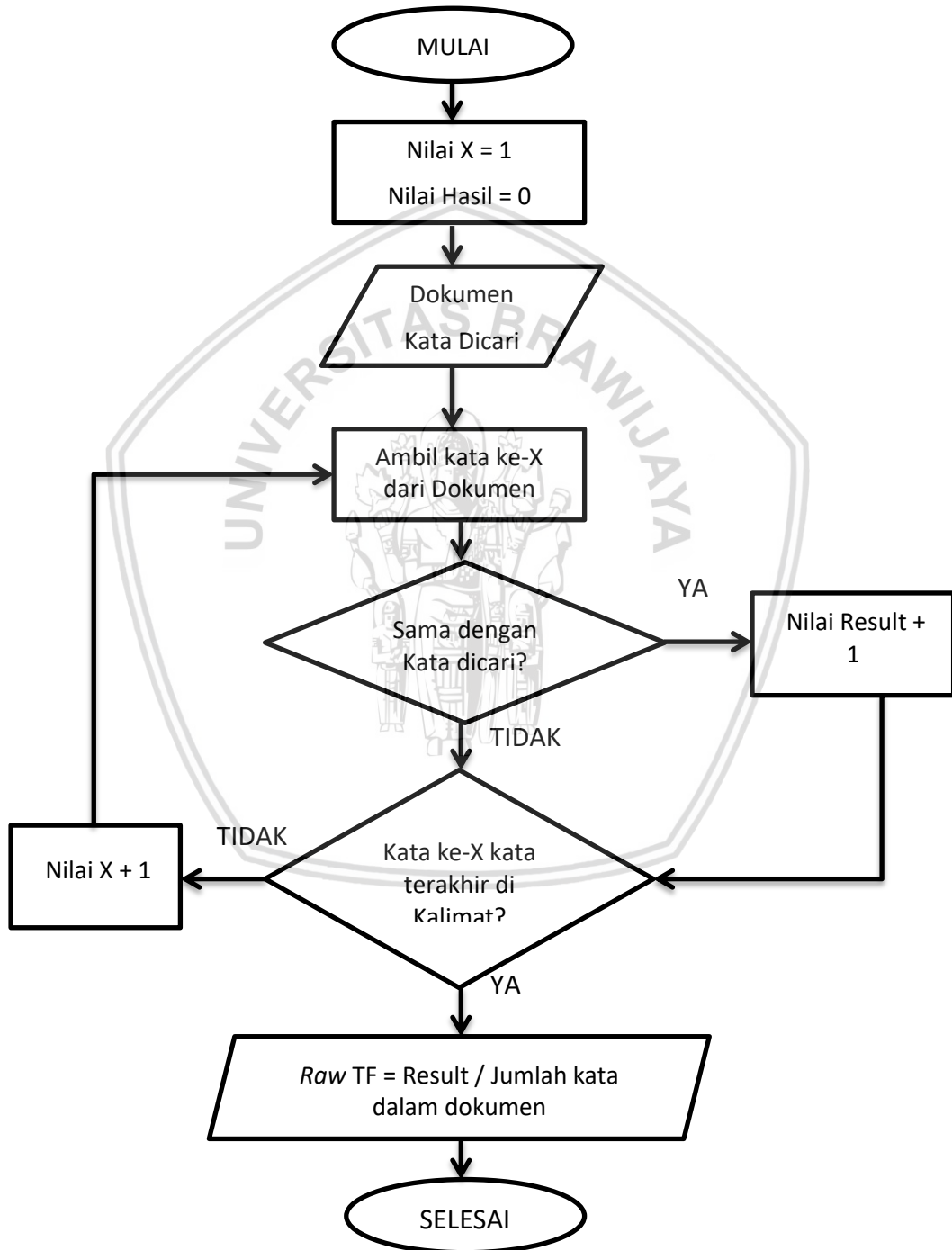
4.2.3.4 Mengklasifikasikan Max TF-IDF Cosine

<i>Flow of Events</i> untuk Max TF-IDF Cosine					
<i>Objective</i>	Untuk melakukan proses pengkategorian menggunakan Max TF-IDF dan KNN Cosine Similarity				
<i>Actor</i>	User				
<i>Pre – Condition</i>	<i>Actor</i> telah berada di halaman awal aplikasi				
<i>Main Flow</i>	<table border="1"> <thead> <tr> <th><i>Actor</i></th> <th><i>System</i></th> </tr> </thead> <tbody> <tr> <td> <ol style="list-style-type: none"> 1. <i>Actor</i> memasukan teks <i>review</i> yang ingin dikategorikan 2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian </td> <td> <ol style="list-style-type: none"> 3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai Max TF-IDF 4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode KNN Cosine Similarity 5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk </td> </tr> </tbody> </table>	<i>Actor</i>	<i>System</i>	<ol style="list-style-type: none"> 1. <i>Actor</i> memasukan teks <i>review</i> yang ingin dikategorikan 2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian 	<ol style="list-style-type: none"> 3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai Max TF-IDF 4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode KNN Cosine Similarity 5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk
	<i>Actor</i>	<i>System</i>			
<ol style="list-style-type: none"> 1. <i>Actor</i> memasukan teks <i>review</i> yang ingin dikategorikan 2. <i>Actor</i> menekan tombol yang tersedia untuk memulai pengkategorian 	<ol style="list-style-type: none"> 3. <i>System</i> memproses teks yang ingin dikategorikan menjadi nilai Max TF-IDF 4. <i>System</i> mengkategorikan <i>review</i> kedalam kategori yang tersedia menggunakan metode KNN Cosine Similarity 5. <i>System</i> menampilkan kategori dimana <i>review</i> tersebut termasuk 				
<i>Alternative Flow</i>	1. Jika field teks belum terisi dan tombol ditekan, akan muncul notifikasi meminta field untuk diisi terlebih dahulu				
<i>Post – Condition</i>	<i>Actor</i> mendapatkan kategori dimana <i>review</i> yang baru saja dimasukan termasuk				

4.3 Perancangan TF-IDF

Pada penelitian ini, dua jenis TF-IDF akan dipakai, yaitu *Raw* dan *Max* TF-IDF.

4.3.1 *Raw* TF



Gambar 4.1 Flowchart *Raw* TF

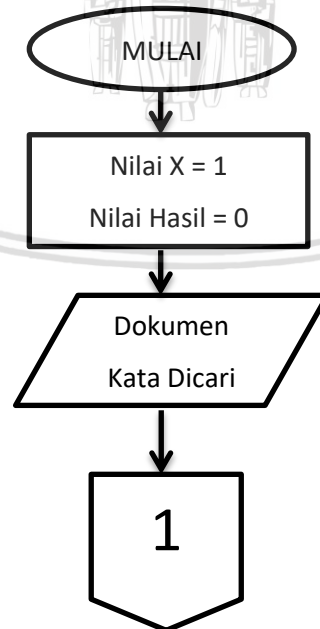
Pada *Raw TF* ini, proses yang dilakukan hanyalah menghitung jumlah kata unik pada suatu dokumen. Nilai *Raw TF* untuk sebuah istilah didalam sebuah dokumen adalah jumlah munculnya istilah yang dihitung dalam dokumen yang dihitung. Nilai ini diambil untuk tiap kata dari tiap dokumen yang telah melalui *Preprocessing*.

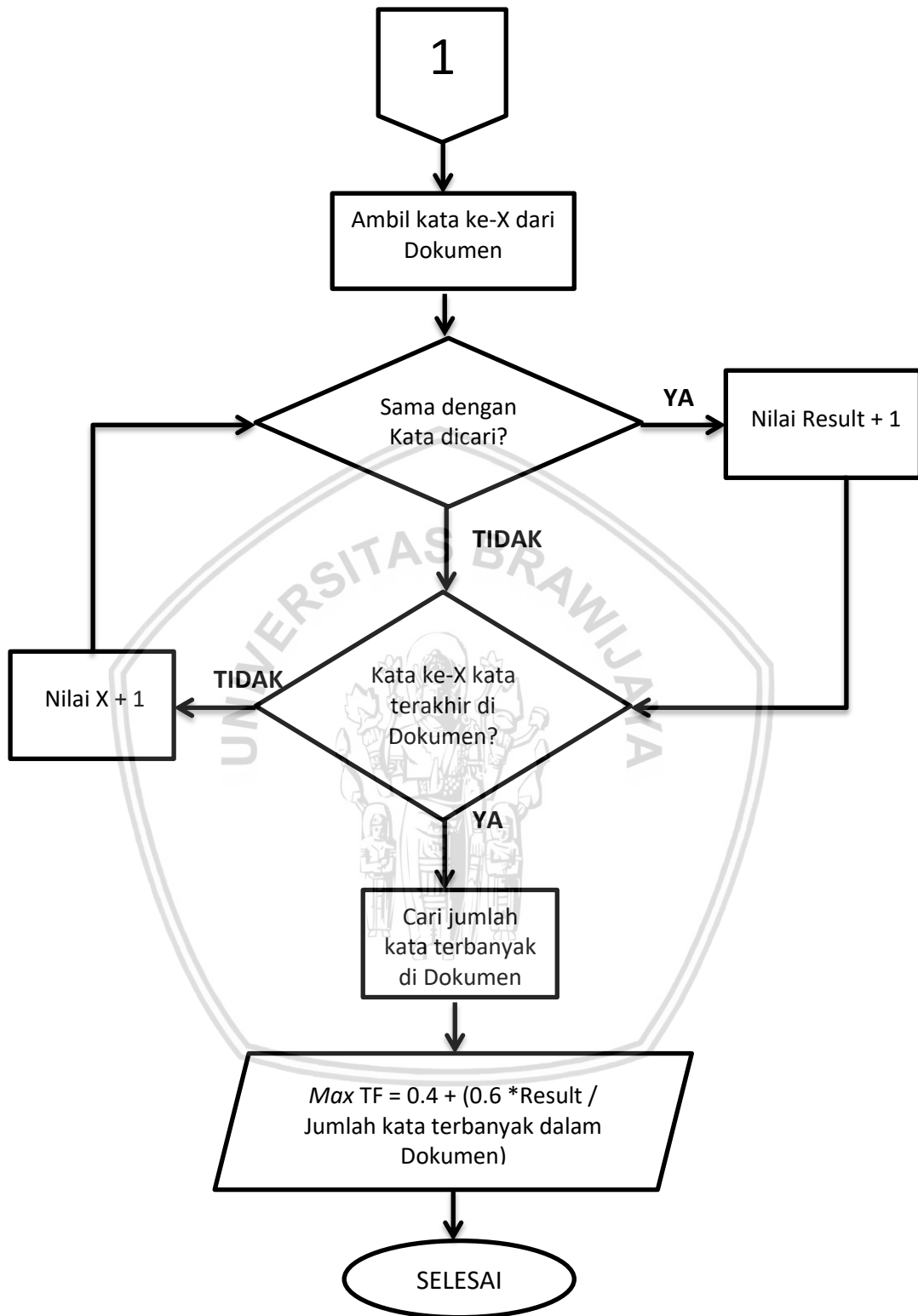
Proses *Raw TF* dilakukan untuk setiap kata. Proses dimulai dengan memasukan kata dan kalimat darimana kata tersebut berasal kedalam proses. Kemudian dari kalimat yang tersedia, akan dibandingkan. Apabila hasil perbandingan ditemukan sama, maka nilai result akan bertambah sebesar 1.

Proses ini diulang untuk setiap kata pada sebuah kalimat, hingga pada akhirnya ditemukan nilai *Raw TF* dari sebuah kata tertentu dengan membagi nilai result dengan jumlah kata pada kalimat yang digunakan sebagai perbandingan. Flowchart untuk *Raw TF* terdapat pada Gambar 4.1

4.3.2 *Max TF*

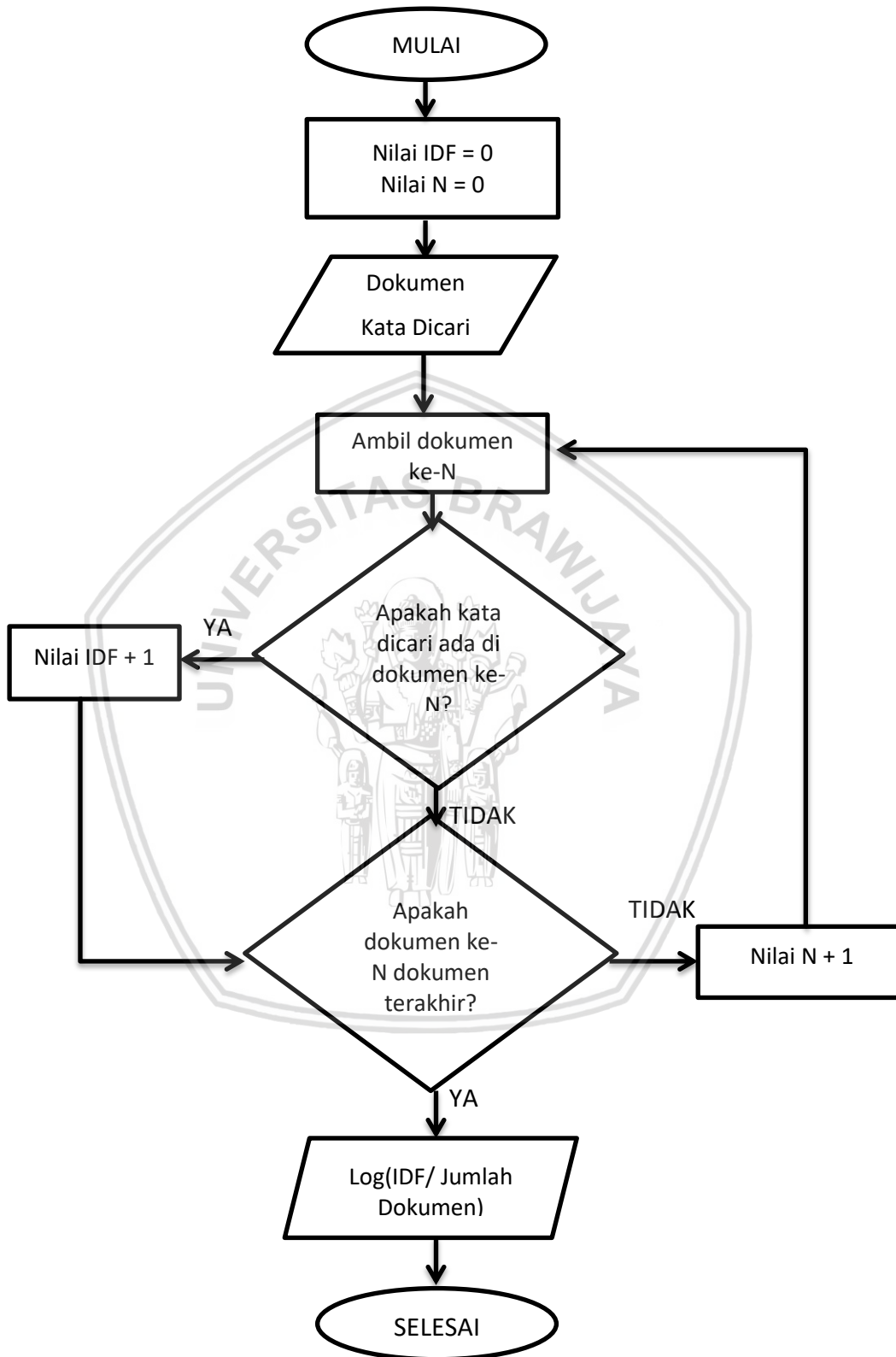
Max TF memodifikasi rumus dari *Raw TF*. Proses ini meliputi pencarian nilai *Raw TF*, sama seperti proses sebelumnya. Perbedaannya terdapat pada langkah selanjutnya, dimana akan ada proses pencarian nilai dari kata dengan jumlah muncul terbanyak pada tiap dokumen. Nilai ini akan membagi nilai *Raw TF* yang sudah didapatkan sebelumnya. Hasilnya kemudian akan dikalikan dengan nilai $1 - a$, sebelum ditambahkan pada a . Nilai a sendiri dapat berubah, namun nilai yang dipakai disini adalah 0.4, sesuai dengan penelitian sebelumnya (Manning, Raghavan dan Schutze, 2008). Flowchart untuk *Max TF* akan digambarkan pada Gambar 4.2





Gambar 4.2 Flowchart *Max TF*

4.3.3 IDF



Gambar 4.3 Flowchart IDF

Proses IDF ini menilai seberapa pentingnya sebuah kata terhadap semua dokumen yang tersedia. Proses IDF dimulai dengan mencari jumlah dokumen dimana sebuah istilah yang sedang dihitung muncul, kemudian membagi total dokumen yang dilatihkan dengan nilai tersebut. Proses diakhiri dengan melakukan logaritma terhadap nilai akhir yang didapat, sehingga akhirnya didapatkan nilai IDF untuk sebuah istilah. Flowchart untuk proses IDF akan digambarkan pada Gambar 4.3.

4.3.4 TF-IDF

Proses TF-IDF dilakukan dengan menjalankan salah satu proses TF yang sudah dijelaskan diatas, kemudian mengalikannya dengan nilai IDF untuk istilah tersebut. Proses ini dilakukan dengan mengambil kata pertama dari List yang sudah dibuat pada proses persiapan dataset, kemudian mencari nilai TF-IDFnya dan meletakkannya kembali kedalam List yang berbeda dari List diatas.

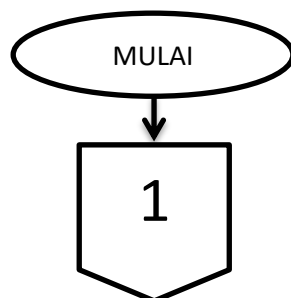
Berbeda dengan List awal, List baru yang dibuat ini perlu menyimpan info berupa kata yang diproses. Dikarenakan info mengenai kata yang diproses ini berbeda beda untuk tiap dokumen, dengan kata yang diproses pada posisi pertama pada List pertama berbeda apabila dibandingkan dengan kata yang diproses pada posisi pertama pada List kedua, info berupa kata ini tidak dapat dijadikan kolom dalam matriks imajinatif. Untuk kasus ini, akan dibuatkan sebuah obyek baru yang berisi nama, untuk menyimpan kata yang diproses, dan value, untuk menyimpan nilai TF-IDF yang dihasilkan.

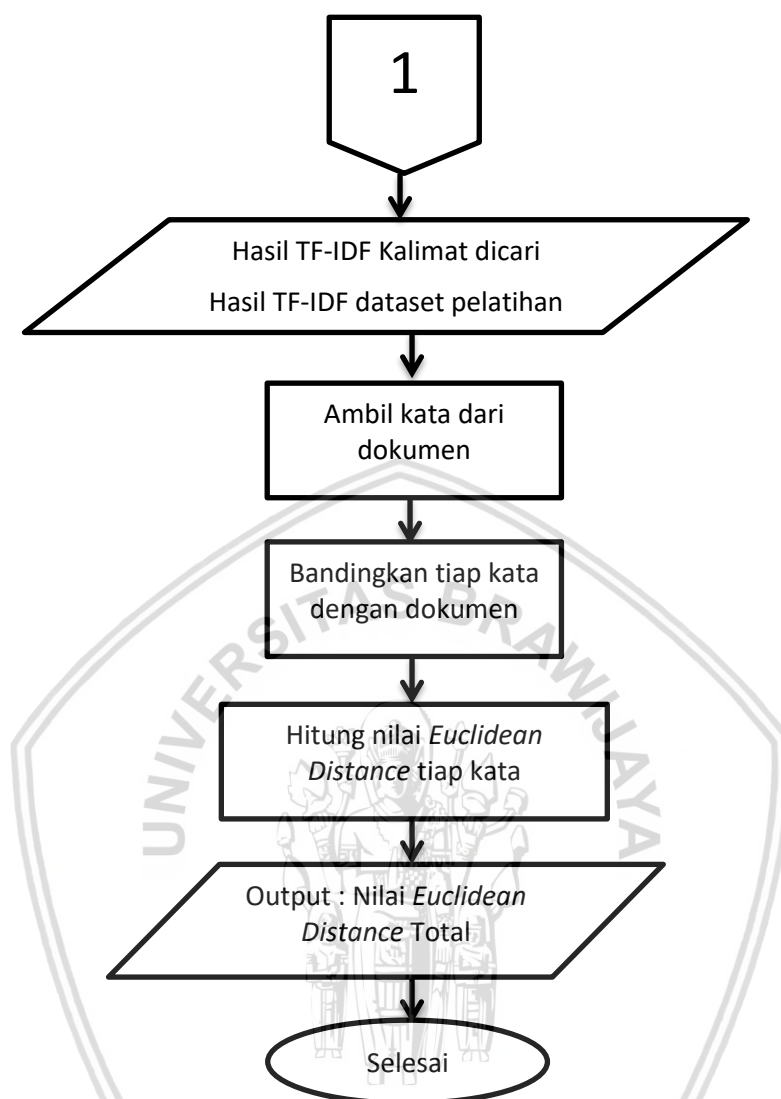
4.4 Perancangan KNN

KNN merupakan salah satu teknik klasifikasi. Cara KNN bekerja adalah dengan mencari sejumlah data yang sudah ditentukan sebelumnya, dengan menghitung kemiripan antara nilai data pengujian dengan semua nilai data pelatihan. Kemudian dari data tersebut, data pengujian akan dimasukan kategori yang memiliki anggota paling banyak didalamnya.

Cara menghitung kemiripannya berbeda beda, namun yang bisa digunakan adalah *Euclidean Distance* dan *Cosine Similarity*. Data dapat diproses oleh salah satu metode tersebut apabila data tersebut sudah dijadikan nilai, dimana pada proses inilah TF-IDF dilakukan. Proses TF-IDF juga dilakukan pada dataset pengujian, dan datanya juga dimasukan kedalam sebuah List yang berbeda dari List data pelatihan.

4.4.1 *Euclidean Distance*



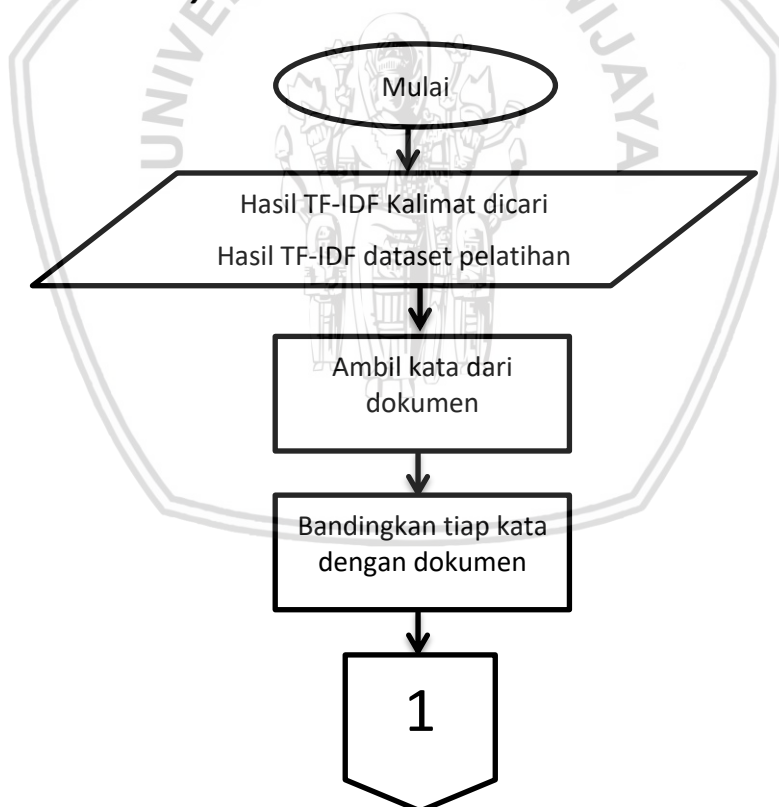


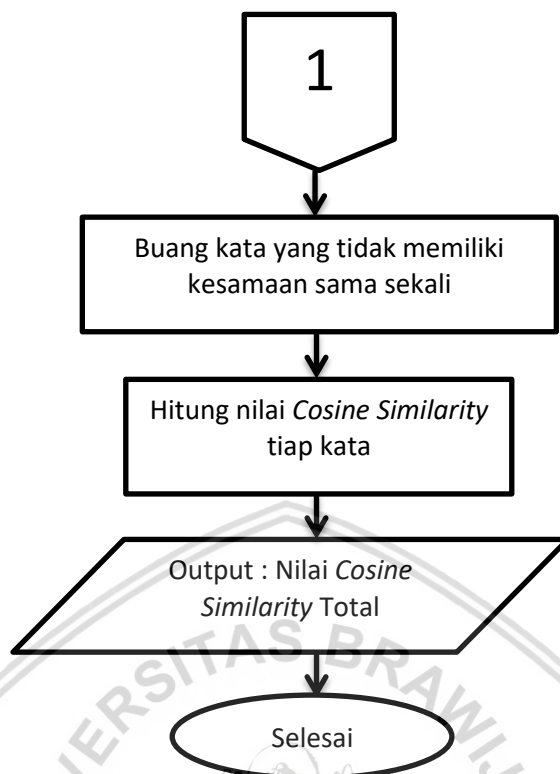
Gambar 4.4 Flowchart *Euclidean Distance*

Pada penghitungan *Euclidean Distance*, tahap pertama adalah mencari nilai pengurangan dari dataset pelatihan terhadap dataset pengujian, untuk kata yang sama. Dibuat sebuah *loop* yang awalnya mengambil kata pertama dari data pengujian, kemudian membandingkan katanya dengan tiap data nama pada List TF-IDF, untuk setiap dokumen. Apabila kata berhasil ditemukan, nilai kata yang baru saja ditemukan tersebut akan dikurangi dengan nilai TF-IDF dari kata pengujian yang sesuai, kemudian nilai tersebut akan dikalikan dengan dirinya sendiri, dan disimpan dalam sebuah variabel sementara. Apabila sebuah kata yang berada di data pengujian tidak ditemukan pasangannya dalam data pelatihan untuk suatu dokumen, maka nilai kata tersebut untuk dokumennya adalah 0, sehingga nilai TF-IDF dari data pengujian akan mewakili hasilnya, dan akan melalui proses pengalihan dengan diri sendiri dan penambahan kedalam variabel sementara juga.

Setelah proses ini dilakukan terhadap semua kata dari dataset pengujian, nilai dari variabel sementara yang sudah didapatkan akan mengalami proses pengakaran, dan hasilnya dimasukkan kedalam sebuah List baru berisi daftar *Euclidean Distance* dari sebuah dataset pengujian terhadap semua dataset pelatihan. Data yang dimasukkan dalam list ini berupa nilai *Euclidean Distance*, nomor dokumen dan kategori dari dokumen tersebut. Proses *Euclidean Distance* ini akan terus diulang untuk semua dokumen dalam semua kategori, hingga didapatkan List berisi semua data tentang nilai *Euclidean Distance* dataset pengujian terhadap dataset pelatihan. Nilai dari semua kategori akan digabungkan kedalam sebuah List, karena pada akhir proses, akan terjadi pengurutan List dari kecil terbesar terhadap nilai *Euclidean Distance* dari masing masing data, dengan nilai terkecil menandakan bahwa dataset pengujian memiliki kedekatan paling dekat terhadap dataset pelatihan yang dibandingkan. List akan dipotong sejumlah nilai K yang sudah ditentukan di awal, dan dari list baru akan dilihat kategori mana yang mendominasi. Dataset pengujian yang sedang diproses akan dimasukkan kedalam kategori yang mendominasi tersebut. Flowchart untuk proses *Euclidean Distance* ditampilkan pada Gambar 4.4.

4.4.2 Cosine Similarity





Gambar 4.5 Flowchart *Cosine Similarity*

Pada Gambar 4.5, ditampilkan *flowchart* dari *Cosine Similarity*. *Cosine Similarity* sendiri memiliki proses yang mirip dengan *Euclidean Distance*. Dibuat loop untuk mengambil data kata dari kalimat yang diuji, kemudian dibandingkan dengan tiap kata dari data pelatihan. Ketika ditemukan data yang sama, nilai TF-IDF dari kedua data akan dikalikan, kemudian dimasukkan kedalam sebuah value. Apabila hingga akhir kalimat tidak ditemukan kata yang sama, maka nilai akan langsung dibuang. Pada Persamaan 2.6, dapat dilihat bahwa data sebenarnya tidak dibuang, namun apabila tidak ditemukan pembandingnya, maka data dianggap dianggap bernilai 0, sehingga salah satu data akan dikalikan dengan nilai 0 yang menyebabkan datanya tidak digunakan.

4.5 Perancangan Klasifikasi Teks

Setelah ditemukan hasil yang menunjukkan kategori dari dataset pengujian, hasil akan dicatat, dan kemudian dibuatkan tabel yang memuat semua data kategori yang terlibat. Akan dibuatkan tabel yang berisi semua data, kebenaran dari proses pengkategorian dan juga presetase kebenaran dari tiap pengujian. Proses akan dilakukan 2 kali, proses pertama menggunakan *Raw TF* dan proses kedua menggunakan *Max TF*, untuk seluruh data yang diujikan. Di akhir pengujian, nilai akurasi akan dibandingkan untuk menentukan teknik TF manakah yang lebih cocok apabila digabungkan dengan *Euclidean Distance* dan *Cosine Similarity* untuk mencari KNN-nya.

4.6 Perhitungan Manual

Pada sub bab ini digambarkan proses perhitungan manual yang akan dilakukan oleh sistem yang dibangun. Untuk contoh, diambil 8 data dari dataset Panichella, dengan 2 data merepresentasikan 1 kelas. Kemudian akan dilakukan semua preprocessing terhadap semua data, dan akan diujikan menggunakan *Max TF-IDF* dan kedua proses KNN. Nilai akurasi akan dihitung sebagai hasil akhirnya. Daftar data yang digunakan akan ditampilkan pada Tabel 4.4 untuk data latih dan Tabel 4.5 untuk data uji

Tabel 4.4 Data Latih Perhitungan Manual

No	Data Latih	Kelas
1	i can't even get in because i don't already have an account	Problem Discovery
2	app logged out even though i had recently used it	Problem Discovery
3	why my wishlist show nothing	Information Seeking
4	why the wishlist won't load	Information Seeking
5	its a good app for viewing steam and your profile etc	Information Giving
6	i use it to see what games are on sale and my profile i don't have to get out my laptop for the same reason	Information Giving
7	this app lacks the ability to redeem steam keys please add it valve	Feature Request
8	add redeem key option too i will honestly be grateful	Feature Request

Tabel 4.5 Data Uji Perhitungan Manual

No	Data Uji	Kelas
1	i installed it and i could not even make an account	Problem Discovery
2	will this app notify me when my wishlists games are on sale	Information Seeking
3	awesome app ever getting news about games being released communicate with friends checking other steam profile and discuss about the issues or problem getting achievements in forum	Information Giving
4	please add the redeem option in the app please	Feature Request

Kedua data akan melalui proses *Tokenizing*, *Stopword* dan *Stemming*, dan hasil *preprocessing* akan ditampilkan pada Tabel 4.6 untuk data latih dan Tabel 4.7 untuk data uji.

Tabel 4.6 Data Latih Perhitungan Manual setelah *preprocessing*

No	Data Latih	Kelas
1	even, get, already, account	Problem Discovery
2	app, log, even, though, recent, us	Problem Discovery
3	wishlist, show, noth	Information Seeking
4	wishlist, load	Information Seeking
5	good, app, view, steam, profil, etc	Information Giving
6	us, see, game, sale, profil, get, laptop, reason	Information Giving
7	app, lack, abil, redeem, steam, kei, pleas, add, valv	Feature Request
8	add, redeem, kei, option, honestli, grate	Feature Request

Tabel 4.7 Data Uji Perhitungan Manual setelah *preprocessing*

No	Data Uji	Kelas
1	install, could, even, make, account	Problem Discovery
2	app, notifi, wishlist, game, sale	Information Seeking
3	awesom, app, ever, get, new, game, releas, commun, friend, check, steam, profil, discuss, issu, problem, get, achiev, forum	Information Giving
4	pleas, add, redeem, option, app, pleas	Feature Request

Setelah dipecah menjadi kata kata terpisah dan dikembalikan ke kata dasar, dilakukan proses IDF terhadap semua data. Tabel 4.8 akan menampilkan hasil pemisahan manual setiap istilah yang muncul dalam satu dokumen dari data latih, serta frekuensi kemunculan setiap istilah dalam setiap dokumen. Setelah itu, akan ditampilkan juga nilai IDF yang didapat dari setiap istilah, agar dapat digunakan untuk setiap dokumen kedepannya.

Tabel 4.8 Frekuensi dan nilai IDF setiap istilah dari data latih

No	Term	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8	IDF
1	even	1	1	0	0	0	0	0	0	1.386
2	get	1	0	0	0	0	1	0	0	1.386

No	Term	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8	IDF
3	alreadi	1	0	0	0	0	0	0	0	2.079
4	account	1	0	0	0	0	0	0	0	2.079
5	app	0	1	0	0	1	0	1	0	0.981
6	log	0	1	0	0	0	0	0	0	2.079
7	though	0	1	0	0	0	0	0	0	2.079
8	recent	0	1	0	0	0	0	0	0	2.079
9	us	0	1	0	0	0	0	0	0	2.079
10	wishlist	0	0	1	1	0	0	0	0	1.386
11	show	0	0	1	0	0	0	0	0	2.079
12	noth	0	0	1	0	0	0	0	0	2.079
13	load	0	0	0	1	0	0	0	0	2.079
14	good	0	0	0	0	1	0	0	0	2.079
15	view	0	0	0	0	1	0	0	0	2.079
16	steam	0	0	0	0	1	0	0	0	2.079
17	profil	0	0	0	0	1	1	0	0	1.386
18	etc	0	0	0	0	1	0	0	0	2.079
19	see	0	0	0	0	0	1	0	0	2.079
20	game	0	0	0	0	0	1	0	0	2.079
21	sale	0	0	0	0	0	1	0	0	2.079
22	laptop	0	0	0	0	0	1	0	0	2.079
23	reason	0	0	0	0	0	1	0	0	2.079
24	lack	0	0	0	0	0	0	1	0	2.079
25	abil	0	0	0	0	0	0	1	0	2.079
26	redeem	0	0	0	0	0	0	1	1	1.386
27	kei	0	0	0	0	0	0	1	1	1.386
28	pleas	0	0	0	0	0	0	1	0	2.079
29	add	0	0	0	0	0	0	1	1	1.386
30	valv	0	0	0	0	0	0	1	0	2.079
31	option	0	0	0	0	0	0	0	1	2.079
32	honestli	0	0	0	0	0	0	0	1	2.079

No	Term	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8	IDF
33	grate	0	0	0	0	0	0	0	1	2.079

Kolom dengan *Header* angka ID menunjukkan jumlah munculnya sebuah istilah yang tertera pada kolom *term* dalam dokumen dengan nomor ID tersebut.

Setelah didapat nilai IDF setiap istilah, akan dicari nilai *Max TF-IDF* untuk setiap istilah, pada setiap dokumen. Tabel 4.9 akan menampilkan hasil *Max TF-IDF* setiap istilah dalam setiap dokumen pada data latih, dan Tabel 4.10 akan menampilkan nilai *Max TF-IDF* dari setiap istilah dalam setiap dokumen pada data uji.

Tabel 4.9 Nilai *Max TF-IDF* tiap istilah dalam tiap kelas pada data latih

No	Term	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8
1	even	1.386	1.386	0	0	0	0	0	0
2	get	2.079	0	0	0	0	0	0	0
3	alreadi	2.079	0	0	0	0	0	0	0
4	account	2.079	0	0	0	0	0	0	0
5	app	0	0.981	0	0	0.981	0	0.981	0
6	log	0	2.079	0	0	0	0	0	0
7	though	0	2.079	0	0	0	0	0	0
8	recent	0	2.079	0	0	0	0	0	0
9	us	0	2.079	0	0	0	0	0	0
10	wishlist	0	0	1.386	1.386	0	0	0	0
11	show	0	0	2.079	0	0	0	0	0
12	noth	0	0	2.079	0	0	0	0	0
13	load	0	0	0	2.079	0	0	0	0
14	good	0	0	0	0	2.079	0	0	0
15	view	0	0	0	0	2.079	0	0	0
16	steam	0	0	0	0	2.079	0	0	0
17	profil	0	0	0	0	1.386	1.386	0	0
18	etc	0	0	0	0	2.079	0	0	0
19	see	0	0	0	0	0	2.079	0	0
20	game	0	0	0	0	0	2.079	0	0
21	sale	0	0	0	0	0	2.079	0	0

No	Term	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8
22	laptop	0	0	0	0	0	2.079	0	0
23	reason	0	0	0	0	0	2.079	0	0
24	lack	0	0	0	0	0	0	2.079	0
25	abil	0	0	0	0	0	0	2.079	0
26	redeem	0	0	0	0	0	0	1.386	1.386
27	kei	0	0	0	0	0	0	1.386	1.386
28	pleas	0	0	0	0	0	0	2.079	0
29	add	0	0	0	0	0	0	1.386	1.386
30	valv	0	0	0	0	0	0	2.079	0
31	option	0	0	0	0	0	0	0	2.079
32	honestli	0	0	0	0	0	0	0	2.079
33	grate	0	0	0	0	0	0	0	2.079

Tabel 4.10 Nilai *Max* TF-IDF tiap istilah dalam tiap kelas pada data uji

No	Term	ID1	ID2	ID3	ID4
1	install	0	0	0	0
2	could	0	0	0	0
3	even	1.386	0	0	0
4	make	0	0	0	0
5	account	2.079	0	0	0
6	app	0	0.981	0.981	0
7	notifi	0	0	0	0
8	wishlist	0	1.386	0	0
9	game	0	2.079	2.079	0
10	sale	0	0	2.079	0
11	awesom	0	0	0	0
12	ever	0	0	2.079	0
13	get	0	0	0.632	0
14	new	0	0	0	0
15	releas	0	0	0	0

No	Term	ID1	ID2	ID3	ID4
16	commun	0	0	0	0
17	friend	0	0	0	0
18	check	0	0	0	0
19	steam	0	0	2.079	0
20	profil	0	0	1.386	0
21	discuss	0	0	0	0
22	issu	0	0	0	0
23	problem	0	0	0	0
24	achiev	0	0	0	0
25	forum	0	0	0	0
26	pleas	0	0	0	0.632
27	add	0	0	0	0.421
28	redeem	0	0	0	1.386
29	option	0	0	0	1.386

Pada Tabel 4.10, beberapa nilai memiliki *Max TF-IDF* sebesar 0. Hal ini disebabkan karena istilah tersebut tidak muncul sama sekali pada data latih dalam kelas manapun, sehingga tidak memiliki nilai IDF.

Setelah didapat nilai *Max TF-IDF* untuk semua data, maka data akan dihitung menggunakan persamaan *Cosine Similarity* yang terdapat pada Persamaan 2.2 dan persamaan *Euclidean Distance* yang terdapat pada Persamaan 2.1. Hasil perhitungan setiap data uji akan ditampilkan pada Tabel 4.11, Tabel 4.12, Tabel 4.13 dan Tabel 4.14.

Tabel 4.11 Hasil Pengujian Data Uji Pertama

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID1	even get already account	0.707	6.160
ID2	app log even though recent us	0.182	6.082
ID3	wishlist show noth	0	7.270
ID4	wishlist load	0	7.270
ID5	good app view steam profil etc	0	7.270
ID6	us see game sale profil get laptop reason	0	7.270

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID7	app lack abil redeem steam kei pleas add valv	0	7.270
ID8	add redeem kei option honestli grate	0	7.270

Tabel 4.11 menampilkan hasil pengujian terhadap data uji pertama, dimana data uji pertama termasuk dalam kelas Problem Discovery. Hasil yang didapatkan menunjukkan bahwa nilai *Cosine Similarity* tertinggi berada pada data pertama, dan nilai *Euclidean Distance* terendah berada pada data kedua. Hal ini menunjukkan bahwa metode berhasil mengklasifikasikan data uji pertama ke kelasnya masing masing, walaupun dokumen yang ditunjuk oleh kedua proses KNN berbeda.

Tabel 4.12 Hasil Pengujian Data Uji Kedua

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID1	even get already account	0	8.903
ID2	app log even though recent us	0.067	8.024
ID3	wishlist show noth	0.174	8.546
ID4	wishlist load	0.226	8.546
ID5	good app view steam profil etc	0.067	8.601
ID6	us see game sale profil get laptop reason	0.487	7.873
ID7	app lack abil redeem steam kei pleas add valv	0.056	8.601
ID8	add redeem kei option honestli grate	0	8.904

Tabel 4.12 menampilkan hasil pengujian terhadap data uji kedua, dimana data uji kedua termasuk dalam kelas Information Seeking. Hasil yang didapatkan menunjukkan bahwa nilai *Cosine Similarity* tertinggi berada pada data keenam, dan nilai *Euclidean Distance* terendah berada pada data keenam. Kedua metode gagal mengklasifikasikan data uji kedua ke kelas yang seharusnya, dikarenakan data keenam termasuk dalam kelas Information Giving.

Tabel 4.13 Hasil Pengujian Data Uji Ketiga

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID1	even get already account	0.215	13.265
ID2	app log even though recent us	0.063	11.793
ID3	wishlist show noth	0	13.499

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID4	wishlist load	0	13.499
ID5	good app view steam profil etc	0.315	12.936
ID6	us see game sale profil get laptop reason	0.475	12.839
ID7	app lack abil redeem steam kei pleas add valv	0.156	13.131
ID8	add redeem kei option honestli grate	0	13.499

Tabel 4.13 menampilkan hasil pengujian terhadap data uji ketiga, dimana data uji pertama termasuk dalam kelas Information Giving. Hasil yang didapatkan menunjukkan bahwa nilai *Cosine Similarity* tertinggi berada pada data keenam, dan nilai *Euclidean Distance* terendah berada pada data kedua. Hal ini menunjukkan bahwa metode KNN *Cosine Similarity* berhasil mengklasifikasikan data uji pada kelas seharusnya, namun KNN *Euclidean Distance* gagal dan mengklasifikasikan data uji kedalam kelas Information Seeking

Tabel 4.14 Hasil Pengujian Data Uji Keempat

No	Data Train	Result <i>Cosine</i>	Result <i>Euclidean</i>
ID1	even get already account	0	8.100
ID2	app log even though recent us	0.054	7.162
ID3	wishlist show noth	0	8.100
ID4	wishlist load	0	8.100
ID5	good app view steam profil etc	0.054	7.804
ID6	us see game sale profil get laptop reason	0	8.100
ID7	app lack abil redeem steam kei pleas add valv	0.509	6.486
ID8	add redeem kei option honestli grate	0.445	7.026

Tabel 4.14 menampilkan hasil pengujian terhadap data uji keempat, dimana data uji pertama termasuk dalam kelas Feature Request. Hasil yang didapatkan menunjukkan bahwa nilai *Cosine Similarity* tertinggi berada pada data ketujuh, dan nilai *Euclidean Distance* terendah juga berada pada data ketujuh. Hal ini menunjukkan bahwa metode berhasil mengklasifikasikan data uji pertama ke kelasnya masing masing, dan juga dengan dokumen yang sama.

BAB 5 IMPLEMENTASI

Bab ini akan membahas tentang implementasi perangkat lunak dari hasil perancangan yang sudah dilakukan pada bab 4.

5.1 Lingkungan Implementasi

Dalam mengimplementasi hasil analisis kebutuhan, dibutuhkan beberapa aspek. Aspek aspek yang dibutuhkan adalah perangkat keras dan perangkat lunak, dengan perangkat keras sendiri menggunakan komputer untuk proses penerapan *coding* dan *mobile phone* untuk pengujian.

5.1.1 Lingkungan Implementasi Perangkat Keras

Dalam pengembangan dan pengujian, dibutuhkan beberapa komponen perangkat keras yaitu komputer dengan spesifikasi:

1. Processor Intel(R) Core(TM) i7-3517U CPU @ 1.90GHz 2.40 GHz
2. Memory 8 GB RAM
3. Hard Disk 500 GB
4. Mouse
5. Keyboard

Selain komputer, diperlukan juga perangkat *mobile phone* dengan spesifikasi:

1. Processor Octa Core (1.5 GHz, Quad Core, Cortex A53)
2. Memory 3 GB RAM

5.1.2 Lingkungan Implementasi Perangkat Lunak

Dalam pengembangan dan pengujian, dibutuhkan beberapa komponen perangkat lunak, yaitu:

1. Sistem operasi Windows 10
2. Bahasa pemrograman Java
3. Aplikasi pemrograman Android Studio
4. Microsoft Word, Microsoft Excel

5.2 Penerapan TF-IDF

Pada pengujian ini, terdapat 2 jenis *Term Frequency* dan 1 jenis *Inverse Document Frequency* yang diterapkan. Tiap fungsi akan dibuat metodenya sendiri, kemudian dipanggil ketika dibutuhkan, dengan mengkombinasikan salah satu jenis metode *Term Frequency* yang tersedia dengan *Inverse Document Frequency*nya.

5.2.1 Raw Term Frequency

Raw Term Frequency merupakan metode pertama dari 2 metode *Term Frequency* yang akan dibahas pada penelitian ini. Parameter yang diminta dari fungsi ini adalah List berisi String dan sebuah String. Fungsi ini akan memberikan nilai terhadap String yang didapat dari parameter, dengan membandingkannya terhadap sebuah list berisi daftar kata (String). Tabel 5.1 akan mendeskripsikan pseudocode dari *Raw Term Frequency* ini.

Tabel 5.1 Raw Term Frequency

1	<code>public double tf(List<String> doc, String Term) {</code>
2	<code> double result = 0;</code>
3	<code> for (String word : doc) {</code>
4	<code> if (Term.equalsIgnoreCase(word))</code>
5	<code> result++;</code>
6	<code> }</code>
7	<code> return result / doc.size();</code>
8	<code>}</code>

Metode ini dipanggil setiap dilakukannya proses TF-IDF. Metode ini memiliki parameter List<String> doc dan String Term, dimana doc berisi satu review yang akan dijadikan perbandingan, dan Term berisi kata yang akan diberi nilai. Pada for di baris ke 3, akan diambil 1 kata dari List<String> doc, kemudian dibandingkan dengan Term yang akan dicari nilainya. Apabila ditemukan persamaan, maka nilai result akan bertambah 1. Pada akhirnya, metode ini akan mengembalikan nilai berupa double yang didapat dari pembagian nilai result dengan jumlah kata yang terdapat dalam List<String> doc.

5.2.2 Maximum Term Frequency

Maximum Term Frequency merupakan metode kedua dari 2 metode *Term Frequency* yang akan dibahas pada penelitian ini. Parameter yang diminta dari fungsi ini adalah List berisi String dan sebuah String. Berbeda dengan *Raw Term Frequency*, *Maximum Term Frequency* memiliki perbedaan pada bagian perhitungan hasil akhir. Bertujuan untuk meredam pengaruh dari panjangnya sebuah kalimat terhadap nilai *Term Frequency*, *Maximum Term Frequency* akan menghitung Term mana yang memiliki jumlah terbanyak dalam suatu kalimat, dan memasukan nilai tersebut kedalam perhitungan. Tabel 5.2 akan mendeskripsikan pseudocode dari *Maximum Term Frequency* ini.

Tabel 5.2 Maximum Term Frequency

1	<code>public double augmentedtf(List<String> doc, String Term) {</code>
2	<code> double result = 0;</code>
3	<code> double maxResult = 0;</code>
4	<code> for (String word : doc) {</code>
5	<code> if (Term.equalsIgnoreCase(word))</code>
6	<code> result++;</code>
7	<code> }</code>
8	<code> for (String word : doc) {</code>
9	<code> int n = 0;</code>
10	<code> for (String wordCompare : doc)</code>

```

11         if (word.equalsIgnoreCase(wordCompare)) {
12             n++;
13         }
14         if (n > maxResult) {
15             maxResult = n;
16         }
17     }
18     return 0.4 + (0.6 * result / maxResult);
19 }

```

Method ini merupakan variasi dari metode pada Tabel 5.1, dimana metode ini adalah salah satu jenis dari *Term Frequency*. *Method* ini memiliki dua parameter, yaitu `List<String> doc` dan `String Term`. `List<String> doc` berisi satu *review* yang akan dijadikan perbandingan, dan *Term* merupakan kata yang akan dicari nilainya. Serupa dengan Tabel 5.1, `String Term` akan dibandingkan dengan semua kata yang terdapat dalam `List<String>` menggunakan `for` pada baris ke-4, kemudian apabila ditemukan persamaan, nilai `result` akan bertambah 1. Setelah `for` ini selesai dilakukan, akan terdapat perulangan `for` lain untuk mencari kata dengan jumlah terbanyak yang terdapat dalam `List<String> doc`.

`For` pada baris ke 8 akan mengambil 1 kata dari `List<String> doc`, dan menyimpannya kedalam variabel `String word`. `For` pada baris ke 10 memiliki fungsi yang sama, namun `for` ini menyimpannya dalam variabel lain, yaitu `wordCompare`. `For` pada baris ke-10 terdapat di dalam `for` pada baris ke-8, sehingga `for` ini akan terus diulang setiap 1 iterasi `for` sebelumnya. Tiap perulangan akan membandingkan kata yang terletak dalam `wordCompare` dengan `word`, dan apabila ditemukan persamaan, akan menambah nilai `n` dengan nilai 1. Pada akhir proses `for` baris ke-8, nilai `n` akan dibandingkan dengan nilai `maxResult` yang bernilai 0 pada awal pemanggilan *method*. Apabila nilai `n` melebihi nilai `maxResult`, maka nilai `maxResult` akan tergantikan oleh `n`. Hal ini memastikan bahwa nilai `maxResult` akan tergantikan setidaknya satu kali dalam *method* ini. Setelah `for` pada baris ke-8 selesai melakukan perulangannya, akan didapatkan nilai `maxResult` berupa nilai yang merepresentasikan jumlah kata terbanyak pada `List<String> doc`. Nilai ini akan mempengaruhi nilai `double` yang dikembalikan oleh *method*. Nilai yang dikembalikan merupakan kode yang terdapat pada baris ke-18. Angka 0.4 dan 0.6 merupakan angka yang sebelumnya sudah ditentukan, dan memiliki hasil yang paling stabil sebagai penghalus nilai (Manning, Raghavan dan Schutze, 2008).

5.2.3 Inverse Document Frequency

Inverse Document Frequency merupakan fungsi untuk mencari seberapa penting sebuah kata dibandingkan dengan sebuah dokumen. Kata ini akan dibandingkan dengan setiap baris dari semua dokumen. Apabila kata tersebut ditemukan pada sebuah baris, maka nilai dari *Inverse Document Frequency* akan bertambah, menyebabkan hasil TF-IDF yang semakin kecil. Tabel 5.3 akan mendeskripsikan pseudocode dari *Inverse Document Frequency* ini.

Tabel 5.3 *Inverse Document Frequency*

```

1 public double idf(List<List<String>> docs, String Term) {
2     double n = 0;
3     for (List<String> doc : docs) {
4         for (String word : doc) {
5             if (Term.equalsIgnoreCase(word)) {
6                 n++;
7                 break;
8             }
9         }
10    }
11    return Math.log(docs.size() / n);
12 }

```

Metode ini merupakan metode kedua yang dibutuhkan oleh TF-IDF. Metode *idf* ini memiliki 2 parameter, yaitu *List<List<String>> docs* dan *String Term*. Metode ini akan mencari jumlah *Term* yang terdapat dalam sebuah dokumen secara total. For pada baris ke-3 akan mengambil *List<String> doc* dari *List<List<String>> docs*, dimana *List* ini berisi setiap satu *review* yang terdapat dalam corpus *List<List<String>> docs*. Kemudian tiap *String Term* akan dibandingkan dengan tiap kata dari *List<String> doc* pada for di baris ke-4. Apabila ditemukan kata yang sama dengan *String Term*, nilai *n* akan bertambah. Setelah for pada baris ke-3 dan ke-4 selesai, metode akan mengembalikan nilai log dari ukuran corpus dibagi oleh nilai *n*, yang merupakan nilai IDF dari sebuah istilah dibandingkan dengan corpus dimana istilah tersebut berasal.

5.3 Penerapan KNN

KNN merupakan proses pengkategorian dengan mencari jumlah kata terdekat yang terlibat, dan membandigkannya classnya dengan class dari kata yang dicari. Kata kata yang ingin dibandingkan harus diberi nilai tersebut, dan proses pemberian nilainya adalah melalui proses TF-IDF diatas. Terdapat dua jenis KNN yang digunakan dalam penelitian ini, *Euclidean Distance* KNN dan *Cosine Similarity* KNN.

5.3.1 *Euclidean Distance* KNN

Euclidean Distance merupakan salah satu teknik KNN, yang menyatakan bahwa dalam mencari kedekatan antar dua titik dalam *Euclidean Space*, dapat ditarik sebuah garis untuk mengukur kedekatan antar kedua titik tersebut. Garis ini dapat merepresentasikan seberapa dekat kedua titik tersebut. Tabel 5.4 berisi kode dari *Euclidean Distance* ini

Tabel 5.4 *Euclidean Distance*

```

1 public String KNN(String testKNN, List<String> KNNArray,
2 List<TFIDFName> KNNtfidf, int k,
3 List<List<List<TFIDFName>>> parent) {
4     KNNEuclidean calculator = new KNNEuclidean();
5     List<TFIDFDistance> KNNDistance = new
6     ArrayList<TFIDFDistance>();

```

```

7      KNNDistance.clear();
8      List<List<TFIDFName>> test = new
9      ArrayList<List<TFIDFName>>();
10     test.clear();
11     String result;
12     String name = "";
13
14     double distance = 0;
15     Boolean checker = false;
16     int discovery = 0, seeking = 0, giving = 0, request =
17     0, category = 0;
18
19     // Counting Distance for each parent
20     for (int a = 0; a < parent.size(); a++) {
21         test = parent.get(a);
22         for (int i = 0; i < test.size(); i++) {
23             // Resetting Distance every new iteration
24             distance = 0;
25             // Checker to get out from next loop
26             checker = false;
27             for (int x = 0; x < KNNArray.size(); x++) {
28                 if (checker) {
29                     break;
30                 }
31                 String z = KNNtfidf.get(x).getName();
32                 for (int j = 0; j < test.get(i).size();
33 j++) {
34                     String comparison =
35 test.get(i).get(j).getName();
36                     // Looking for same text
37                     if (z.equals(comparison)) {
38                         distance +=
39 Math.pow(KNNtfidf.get(x).getValue() -
40 test.get(i).get(j).getValue(), 2);
41                         break;
42                     } else if (j == test.get(i).size() - 1)
43 {
44                         // Put 0 as value if can't found
45 the same one from text
46                         distance += Math.pow(0 -
47 test.get(i).get(j).getValue(), 2);
48                     }
49                 }
50             }
51             //Euclidean Ver
52             distance = Math.sqrt(distance);
53             name = "Parent " + String.valueOf(a + 1);
54             KNNDistance.add(new TFIDFDistance(name,
55 distance, a));
56         }
57     }
58     //Euclidean Ver
59     Collections.sort(KNNDistance, new KNNComparator());
60
61     //Euclidean Ver
62     List<TFIDFDistance> SelectedDistance = new
63 ArrayList<TFIDFDistance>(k);
64     SelectedDistance.clear();

```

```

66     for (int i = 0; i < k; i++) {
67         SelectedDistance.add(KNNDistance.get(i));
68         // Increase the value of category everytime each
69         new item
70         if (KNNDistance.get(i).getParent() == 0) {
71             discovery++;
72         } else if (KNNDistance.get(i).getParent() == 1) {
73             seeking++;
74         } else if (KNNDistance.get(i).getParent() == 2) {
75             giving++;
76         } else if (KNNDistance.get(i).getParent() == 3) {
77             request++;
78         }
79     }
80     // Decide which category has higher value
81     if (discovery > seeking && discovery > giving &&
82     discovery > request) {
83         result = "Problem Discovery";
84     } else if (seeking > giving && seeking > request) {
85         result = "Information Seeking";
86     } else if (giving > request) {
87         result = "Information Giving";
88     } else {
89         result = "Feature Request";
90     }
91     // Return Category
92     Log.d("list", "Discovery: " + discovery);
93     Log.d("list", "Seeking: " + seeking);
94     Log.d("list", "Giving: " + giving);
95     Log.d("list", "Request: " + request);
96     Log.d("list", "End Result: " + result);
97     return result;
98 }

```

Metode ini merupakan seluruh metode untuk memproses KNN melalui *Euclidean Distance*. Parameter yang dimiliki oleh metode ini adalah String *testKNN* yang berisi kalimat yang ingin diuji, *List<String> KNNArray* yang berisi hasil pemecahan kalimat yang ingin diuji menjadi sebuah *List*, *List<TFIDFName> KNNtfidf* yang berisi hasil TF-IDF terhadap kalimat yang akan diuji, *int k* yang berisi nilai K dari KNN dan *List<List<List<TFIDFName>>> parent* yang berisi seluruh corpus data training yang akan dijadikan perbandingan.

Proses pertama terjadi pada *for* di baris ke-17, dimana *for* ini akan diulangi untuk setiap data dalam *List parent*. *List parent* sendiri berisi *List* yang mencakup tiap kelas yang akan diuji, bervariasi antara 4 *List* atau 2 *List*, tergantung pada data training. *For* pada baris ke-19 akan mengulang untuk setiap *review* pada tiap kelas. *Distance* akan direset menjadi 0 untuk setiap iterasi *for* baris ke-19 pada baris ke-21. *For* pada baris ke-24 akan mengulang untuk setiap kata pada data yang akan diujikan. Setiap perulangan, nilai *String z* akan diupdate dengan kata yang sedang diuji. *String z* akan dibandingkan dengan tiap kata dalam tiap kelas pada *for* baris ke-29. *If* pada baris ke-32 akan mencari tahu apakah data dalam *String z* sama dengan tiap kata pada kalimat *review* yang sedang dibandingkan. Apabila ditemukan kata yang sama, maka nilai *distance*

akan ditambahkan dengan hasil kuadrat dari nilai TF-IDF kata yang diujikan dengan nilai TF-IDF kata pada data training, setelah itu proses akan kelar dari for baris ke-29 dan kembali ke for pada baris ke-24 untuk mencari kata lain pada data training. Namun apabila hingga akhir tidak ditemukan kata yang sesuai, akan tetap dilakukan penambahan nilai ke distance menggunakan kuadrat dari 0 dikurangi dengan nilai TF-IDF dari kata yang diuji, dan kemudian kembali ke for baris ke-24.

Setelah for baris ke-24 selesai dilakukan, sistem akan melanjutkan proses pada for ke-19, yaitu mengakar nilai distance, memberi nama untuk kelas yang diuji dan kemudian memasukan semua hasil tersebut kedalam List KNNDistance. Setelah selesai, sistem akan kembali ke for baris ke-17 untuk melakukan hal yang sama pada kelas kedua. Proses ini akan terus diulang hingga kelas terakhir. Setelah pengulangan selesai dilakukan, akan dilakukan pengurutan menggunakan metode KNNComparator pada baris ke-52. Pada for baris ke-58, akan dilakukan pengulangan sebanyak K yang sudah ditentukan sebelumnya. Di dalam for ini, akan dicari data yang paling dekat dengan data test sejumlah K, beserta dimana kelas asal dari masing masing data yang dekat. Setelah semua data diatur dan diketahui posisi serta kelas asalnya, hasil akhir akan di print ke layar sebagai hasil akhir.

5.3.2 Cosine Similarity KNN

Cosine Similarity merupakan metode kedua dalam mencari kesamaan antara dua data. *Cosine Similarity* mengukur kosine dari sudut antara dua vector. Jarak nilai yang bisa didapat dalam pengkalsifikasian teks adalah 1 – 1, dengan 0 merepresentasikan bahwa kedua vector memiliki orientasi yang sama dan 0 merepresentasikan bahwa kedua vector memiliki sudut sebesar 90 derajat, dan sangat berbeda antara satu dengan yang lain.

Tabel 5.5 *Cosine Similarity*

1	<code>public String KNN(String testKNN, List<String> KNNArray,</code>
2	<code>List<TFIDFName> KNNtfidf, int k,</code>
3	<code>List<List<List<TFIDFName>>> parent) {</code>
4	<code> KNNEuclidean calculator = new KNNEuclidean();</code>
5	<code> List<TFIDFDistance> KNNDistanceCosine = new</code>
6	<code>ArrayList<TFIDFDistance>();</code>
7	<code> KNNDistanceCosine.clear();</code>
8	<code> List<List<TFIDFName>> test = new</code>
9	<code>ArrayList<List<TFIDFName>>();</code>
10	<code> KNNDistanceCosine.clear();</code>
11	<code> String result;</code>
12	<code> String name = "";</code>
13	
14	<code> double distanceCosine = 0;</code>
15	<code> double dividerCosineTrain = 0;</code>
16	<code> double dividerCosineTest = 0;</code>
17	<code> Boolean checker = false;</code>
18	<code> int discovery = 0, seeking = 0, giving = 0, request =</code>
19	<code>0, category = 0;</code>
20	
21	<code> // Counting Distance for each parent</code>

```

22     for (int a = 0; a < parent.size(); a++) {
23         test = parent.get(a);
24         for (int i = 0; i < test.size(); i++) {
25             // Resetting Distance every new iteration
26             distanceCosine = 0;
27             dividerCosineTrain = 0;
28             dividerCosineTest = 0;
29             // Checker to get out from next loop
30             checker = false;
31             for (int x = 0; x < KNNArray.size(); x++) {
32                 if (checker) {
33                     break;
34                 }
35                 String z = KNNtfidf.get(x).getName();
36                 for (int j = 0; j < test.get(i).size();
37 j++) {
38                     String comparison =
39 test.get(i).get(j).getName();
40                     // Looking for same text
41                     if (z.equals(comparison)) {
42                         distanceCosine +=
43 KNNtfidf.get(x).getValue() * test.get(i).get(j).getValue();
44                         break;
45                     }
46                 }
47             }
48             //Cosine Ver
49             for (int j = 0; j < test.get(i).size(); j++) {
50                 dividerCosineTrain +=
51 Math.pow(test.get(i).get(j).getValue(), 2);
52             }
53             dividerCosineTrain =
54 Math.sqrt(dividerCosineTrain);
55             for (int x = 0; x < KNNArray.size(); x++) {
56                 dividerCosineTest +=
57 Math.pow(KNNtfidf.get(x).getValue(), 2);
58             }
59             dividerCosineTest =
60 Math.sqrt(dividerCosineTest);
61             distanceCosine = distanceCosine /
62 (dividerCosineTest * dividerCosineTrain);
63             name = "Parent " + String.valueOf(a + 1);
64             KNNDistanceCosine.add(new TFIDFDistance(name,
65 distanceCosine, a));
66         }
67     }
68     // Sort the List, the lower the value, higher it's
69 place supposed to be
70     //Cosine Ver
71     Collections.sort(KNNDistanceCosine, new
72 KNNComparatorCosine());
73
74     //Cosine Ver
75     List<TFIDFDistance> SelectedDistanceCosine = new
76 ArrayList<TFIDFDistance>(k);
77     SelectedDistanceCosine.clear();
78     // Delete OK
79     for (int i = 0; i < k; i++) {
80

```

```

81 SelectedDistanceCosine.add(KNNDistanceCosine.get(i));
82     // Increase the value of category everytime each
83     new item
84     if (KNNDistanceCosine.get(i).getValue() != 0) {
85         if (KNNDistanceCosine.get(i).getParent() == 0)
86     {
87         discovery++;
88     } else if (KNNDistanceCosine.get(i).getParent()
89 == 1) {
90         seeking++;
91     } else if (KNNDistanceCosine.get(i).getParent()
92 == 2) {
93         giving++;
94     } else if (KNNDistanceCosine.get(i).getParent()
95 == 3) {
96         request++;
97     }
98     }
99     }
100
101     // Decide which category has higher value
102     if (discovery > seeking && discovery > giving &&
103     discovery > request) {
104         result = "Problem Discovery";
105     } else if (seeking > giving && seeking > request) {
106         result = "Information Seeking";
107     } else if (giving > request) {
108         result = "Information Giving";
109     } else {
110         result = "Feature Request";
111     }
112     // Return Category
113     Log.d("list", "Discovery: " + discovery);
114     Log.d("list", "Seeking: " + seeking);
115     Log.d("list", "Giving: " + giving);
116     Log.d("list", "Request: " + request);
117     Log.d("list", "End Result: " + result);
118     return result;
119 }

```

Metode ini merupakan seluruh metode untuk memproses KNN melalui *Cosine Similarity*. Parameter yang dimiliki oleh metode ini adalah String testKNN yang berisi kalimat yang ingin diuji, List<String> KNNArray yang berisi hasil pemecahan kalimat yang ingin diuji menjadi sebuah List, List<TFIDFName> KNNtfidf yang berisi hasil TF-IDF terhadap kalimat yang akan diuji, int k yang berisi nilai K dari KNN dan List<List<List<TFIDFName>>> parent yang berisi seluruh corpus data training yang akan dijadikan perbandingan.

Proses dimulai pada for baris ke-20, dimana sama seperti pada table 5.4, akan memisakan tiap kelas. Tiap kelas akan dimasukan kedalam variabel test. For baris ke-22 akan memisahkan tiap *review*, dan untuk setiap *review*, kan terjadi prulangan for pada baris ke-29. For baris ke-29 akan mengambil setiap kata pada data test, dan membandingkannya dengan masing masing kata pada masing masing *review*. Apabila ditemukan kata yang sama, maka nilai distance akan

ditambahkan dengan nilai TF-IDF dari data test dikalikan dengan nilai TF-IDF dari data train. Apabila data tidak ditemukan, maka nilai akan dikalikan dengan 0 sehingga menghasilkan 0, dan data distance tidak bertambah, sehingga proses ini tidak perlu dituliskan. Proses akan kembali ke for baris ke-29 untuk melakukan proses yang sama pada kata selanjutnya pada data test.

Setelah semua kata pada data test selesai diproses, nilai TF-IDF pada tiap kata pada data test akan ditambahkan setelah dikuadrat. Hal serupa dilakukan pada *review* data test yang sedang diuji, seperti direpresentasikan pada for baris ke-45 dan baris ke-50. Hasil dari kedua for kemudian akan dikalikan, dan hasilnya digunakan untuk membagi nilai distance yang didapat dari for baris ke-22. Data ini kemudian akan disimpan dalam *KNNDistanceCosine* beserta di kelas mana data ini berasal. Semua proses akan diulangi untuk setiap data pada setiap kelas. Setelah pengulangan selesai dilakukan, akan dilakukan pengurutan menggunakan metode *KNNComparator* pada baris ke-65. Pada for baris ke-72, akan dilakukan pengulangan sebanyak K yang sudah ditentukan sebelumnya. Di dalam for ini, akan dicari data yang paling dekat dengan data test sejumlah K, beserta dimana kelas asal dari masing masing data yang dekat. Setelah semua data diatur dan diketahui posisi serta kelas asalnya, hasil akhir akan di print ke layar sebagai hasil akhir.



BAB 6 PENGUJIAN DAN ANALISIS

Bab pengujian dan analisis akan menampilkan semua pengujian yang dilakukan, hasil pengujian yang didapat serta analisis dari hasil hasil tersebut.

6.1 Dataset Positif Negatif

Pada proses ini, setiap pengujian akan dilakukan menggunakan dataset positif dan negatif. Pengujian diharapkan dapat memberikan hasil berupa akurasi dari setiap gabungan metode yang digunakan, sehingga diketahui pengaruh dari *Maximum* TF-IDF yang diterapkan ke masing masing metode, apabila dibandingkan dengan *Raw* TF-IDF. Pengujian ini juga diharapkan dapat menunjukan metode gabungan mana yang paling efektif diantara ke-4 metode yang diujikan, sehingga dapat menjadi acuan untuk pengujian menggunakan dataset Panichella.

6.1.1 Optimal K

Proses awal yang dilakukan adalah mencari nilai optimal K dari masing masing metode. Sistem akan mencari nilai K yang paling optimal dari masing masing gabungan *method*, untuk masing masing *dataset*. Nilai K sendiri berubah untuk setiap *dataset* yang digunakan, sehingga harus dicari ulang nilai K optimal untuk tiap gabungan *dataset* dan *method*.

Nilai K yang secara umum digunakan menggunakan rumus dasar yaitu akar daripada jumlah dataset *training* yang digunakan (Hassanat, 2014). Untuk dataset positif dan negatif, total dataset *training* yang digunakan sebesar 200 data, sehingga nilai umum yang didapat adalah 15. Nilai K yang akan diujikan menggunakan 10 nilai ganjil disekitar nilai umum ini, dengan nilai K yang diambil sebesar 11 – 27.

Pengujian pertama merupakan pengujian K Optimal menggunakan kedua *method* gabungan *Term Frequency - Inverse Document Frequency* (TF-IDF) dengan K-Nearest Neighbor (KNN). Hasil pengujian tertera pada tabel 6.1

Tabel 6.1 Pengujian Nilai K Dataset Positif Negatif Raw TF-IDF + KNN Euclidean

K	Accuracy (%)			Macro F-Measure
	Positive	Negative	Total	
11	80	60	70	0.70
13	77	60	68	0.68
15	80	60	70	0.70
17	90	40	65	0.63
19	87	33	60	0.57
21	87	33	60	0.57
23	87	33	60	0.57
25	83	40	62	0.60
27	87	43	65	0.63

K	Accuracy (%)			Macro F-Measure
	Positive	Negative		
29	83	47	65	0.64

Hasil pengujian pada Tabel 6.1 menunjukkan hasil pengujian dengan nilai K antara 11 hingga 29, dengan pengambilan nilai ganjil sebagai K. Pengujian nilai K dengan akurasi tertinggi didapat dengan nilai K sebesar 11 dan 15, dengan akurasi total masing masing sebesar 70%. Nilai K yang akan diambil adalah nilai yang paling mendekati aturan umum K dalam KNN, yaitu K = 15. Nilai K ini akan dibawa untuk menjadi perbandingan dengan *method method* lain, pada pengujian selanjutnya.

Pada proses *Raw TF-IDF* terhadap KNN *Raw Euclidean* ini, data yang memiliki akurasi tinggi juga memiliki perbandingan akurasi positif dan negatif yang relatif seimbang, sehingga nilai F-Measure tidak terlalu berbeda dengan akurasi. Perbedaan yang terlihat terdapat pada nilai K = 19, 21 dan 23, dimana perbedaan terdapat perbedaan akurasi sebesar lebih dari 50% antara kelas klasifikasi positif dan negatif. Hal ini mempengaruhi nilai F-Measure secara cukup signifikan.

Pengujian selanjutnya merupakan pengujian serupa dengan pengujian pertama. Pengujian optimal K akan dilakukan dengan nilai K yang sama dengan pengujian pada Tabel 6.1, menggunakan *method* gabungan *Maximum TF-IDF* dengan KNN *Euclidean Distance*. Hasil pengujian tertera pada Tabel 6.2

Tabel 6.2 Pengujian Nilai K Dataset Positif Negatif *Max TF-IDF* + KNN *Euclidean*

K	Accuracy (%)			Macro F-Measure
	Positive	Negative	Total	
11	53	83	68	0.68
13	73	90	82	0.82
15	67	77	72	0.72
17	60	60	60	0.60
19	80	50	65	0.64
21	80	40	60	0.58
23	80	50	65	0.64
25	70	53	62	0.61
27	73	70	72	0.72
29	67	73	70	0.70

Hasil pengujian pada Tabel 6.2 menunjukkan bahwa nilai K optimal yang didapat merupakan 13, dimana akurasi yang didapat dengan K sebesar 13 adalah 82% dan F-Measure sebesar 0.82.

Pengujian dilakukan lagi untuk mencari nilai K optimal dari *method Raw TF-IDF* dan KNN *Cosine Similarity*. Hasil pengujian tertera pada Tabel 6.3



Tabel 6.3 Pengujian Nilai K Dataset Positif Negatif *Raw TF-IDF + KNN Cosine*

K	Accuracy (%)			Macro
	Positive	Negative	Total	F-Measure
11	83	77	80	0.80
13	87	77	82	0.82
15	87	80	83	0.83
17	90	77	83	0.83
19	90	77	83	0.83
21	90	80	85	0.85
23	90	80	85	0.85
25	90	80	85	0.85
27	90	80	85	0.85
29	90	80	85	0.85

Hasil pengujian pada tabel 6.3 menunjukkan nilai akurasi dari pengujian dataset positif negatif terhadap *Raw TF-IDF KNN Cosine Similarity*. Didapatkan nilai akurasi dan F-Measure dengan rata rata lebih tinggi dari *KNN Euclidean Distance*. Hal ini menunjukkan bahwa *Cosine Similarity* lebih stabil dan akurat dalam pengujian menggunakan database positif negatif, karena *Cosine Similarity* tidak memperhatikan kata yang hanya muncul sekali. Nilai K terbaik merupakan K = 21, 23, 25, 27 dan 29. Pengambilan K mengikuti proses sebelumnya, yaitu K yang paling mendekati aturan umum. Dalam kasus ini, K = 21.

Pengujian diulangi lagi untuk pencarian K optimal dari *method* gabungan *Max TF-IDF* dan *KNN Cosine Similarity*. Hasil pengujian tertera pada Tabel 6.4

Tabel 6.4 Pengujian Nilai K Dataset Postif Negatif *Max TF-IDF + KNN Cosine*

K	Accuracy (%)			Macro
	Positive	Negative	Total	F-Measure
11	80	83	82	0.82
13	77	83	80	0.80
15	83	77	80	0.80
17	87	77	82	0.82
19	87	77	82	0.82
21	87	80	83	0.83
23	87	80	83	0.83
25	87	80	83	0.83
27	90	80	85	0.85
29	90	83	87	0.87

Pada tabel 6.4, terlihat bahwa nilai K paling optimal terdapat pada K = 29, sehingga nilai tersebut akan diambil sebagai nilai K mewakili *method* gabungan *Max TF-IDF KNN Cosine Similarity*, untuk pengujian *dataset positive negative*.

Dari keempat pengujian yang sudah dilakukan diatas, didapatkan nilai K yang mewakili masing masing metode pengujian, yaitu K = 11 untuk *Raw TF-IDF + KNN Euclidean Distance*, K = 15 untuk *Max TF-IDF + KNN Euclidean Distance*, K = 21 untuk *Raw TF-IDF + KNN Cosine Similarity* dan K = 29 untuk *Max TF-IDF + KNN Cosine Similarity*. Perbandingan akan tetap dilakukan untuk setiap 10 K yang sudah dihitung, namun nilai K optimal untuk masing masing metode akan menjadi acuan yang akan dibahas nantinya.

6.1.2 Akurasi and F-Measure

Pada analisis ini, akan dibandingkan nilai akurasi dan F-Measure dari setiap metode, K dan kelas untuk mencari tahu pengaruh penerapan *Max TF-IDF* terhadap *KNN Euclidean Distance* dan *Cosine Similarity*.

Analisis pertama akan membandingkan pengaruh penerapan *Max TF-IDF* terhadap *KNN Euclidean Distance*. Hasil pengujian *Raw TF-IDF* akan ditampilkan pada Tabel 6.5 sedangkan *Max TF-IDF* akan ditampilkan pada Tabel 6.6.

Tabel 6.5 Akurasi dan F-Measure *Raw TF-IDF + KNN Euclidean* Dataset Positif Negatif

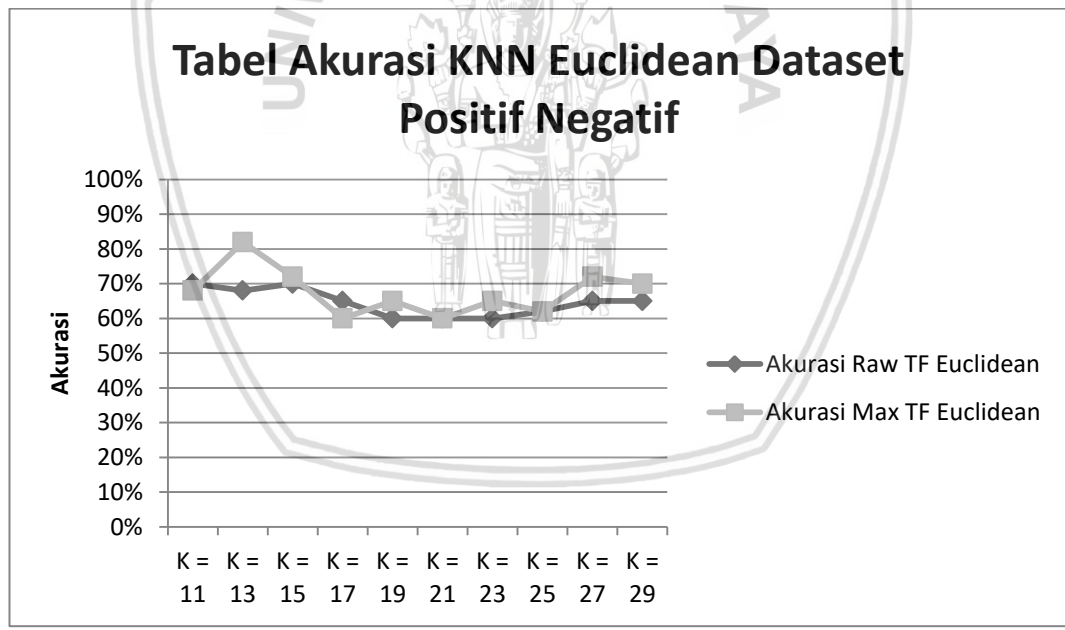
K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
11	80	60	70	0.71	0.70	0.70
13	77	60	68	0.69	0.68	0.68
15	80	60	70	0.71	0.70	0.70
17	90	40	65	0.70	0.65	0.63
19	87	33	60	0.64	0.60	0.57
21	87	33	60	0.64	0.60	0.57
23	87	33	60	0.64	0.60	0.57
25	83	40	62	0.64	0.62	0.60
27	87	43	65	0.68	0.65	0.63
29	83	47	65	0.67	0.65	0.64

Tabel 6.6 Akurasi dan F-Measure *Max TF-IDF + KNN Euclidean* Dataset Positif Negatif

K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
11	53	83	68	0.70	0.68	0.68
13	73	90	82	0.83	0.82	0.82
15	67	77	72	0.72	0.72	0.72
17	60	60	60	0.60	0.60	0.60
19	80	50	65	0.66	0.65	0.64

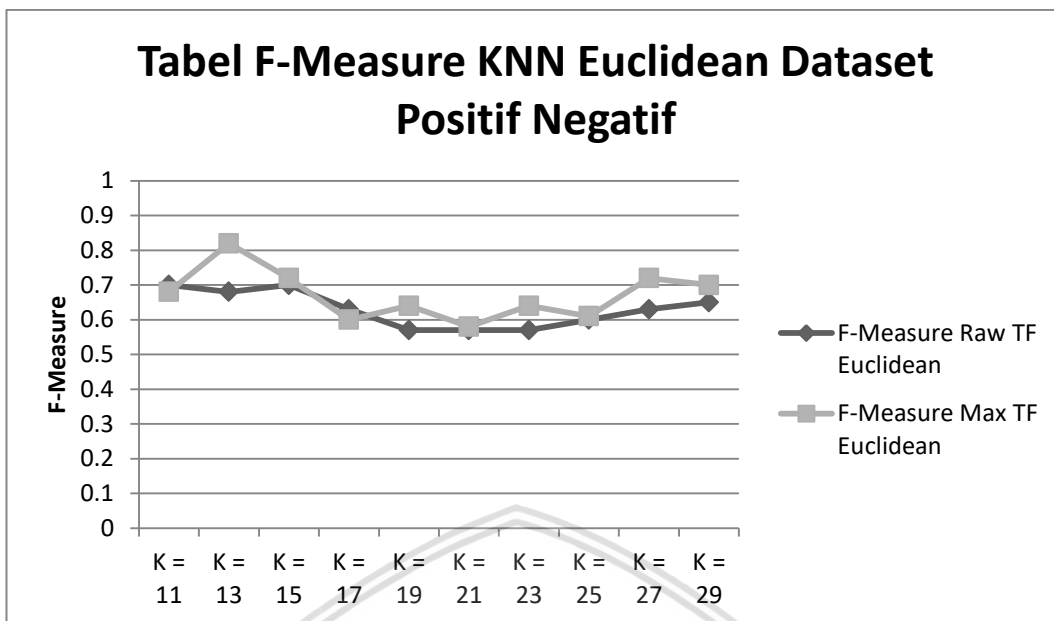
K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
21	80	40	60	0.62	0.60	0.58
23	80	50	65	0.66	0.65	0.64
25	70	53	62	0.62	0.62	0.61
27	73	70	72	0.72	0.72	0.72
29	67	73	70	0.70	0.70	0.70

Dari hasil pengujian, dapat dibandingkan akurasi dan F-Measure yang didapat untuk KNN *Euclidean Distance*, menggunakan kedua jenis TF-IDF. Dari 10 pengujian, hanya pada 2 pengujianlah akurasi dan F-Measure dan *Max TF-IDF* dan KNN *Euclidean* memberikan hasil yang lebih buruk daripada *Raw TF-IDF* dan KNN *Euclidean*. 8 hasil lainnya menunjukkan bahwa menggunakan dataset yang sama, *Max TF-IDF* memberikan nilai akurasi dan F-Measure yang lebih tinggi. *Max TF-IDF* dapat meningkatkan hasil akurasi dan F-Measure secara cukup besar pada KNN *Euclidean*, hingga 12% pada kondisi optimal masing masing metode, membuat *Max TF-IDF* ini metode yang efektif apabila digunakan bersamaan dengan KNN *Euclidean Distance*. Perbandingan lebih jelas dapat dilihat pada Gambar 6.1 dan Gambar 6.2.



Gambar 6.1 Tabel perbandingan Akurasi *Raw TF-IDF* dan *Max TF-IDF* terhadap KNN *Euclidean Distance*

Pada Gambar 6.1, terlihat bahwa posisi dimana nilai akurasi *Max TF-IDF* berada atas akurasi *Raw TF-IDF* 8 dari 10 percobaan. Diantara 8 percobaan tersebut, termasuk didalamnya kedua K yang menjadi K optimal masing masing metode. Hal ini menunjukkan bahwa dalam kondisi *Raw TF-IDF* dengan K optimal pun, *Max TF-IDF* masih menghasilkan nilai akurasi yang lebih baik.



Gambar 6.2 Tabel perbandingan F-Measure *Raw TF-IDF* dan *Max TF-IDF* terhadap KNN *Euclidean Distance*

Gambar 6.2 menunjukkan perbandingan antara F-Measure *Raw TF-IDF* KNN *Euclidean* dengan *Max TF-IDF* KNN *Euclidean*. Hasil yang serupa dengan Gambar 6.1 didapatkan pada Gambar 6.2, dengan *Raw TF-IDF* memiliki hasil yang lebih baik hanya 2 kali dari 10 percobaan, tidak termasuk didalamnya nilai K optimal untuk KNN *Euclidean* (13 untuk *Max TF-IDF* KNN *Euclidean Distance* dan 15 untuk *Raw TF-IDF* KNN *Euclidean Distance*). Hasil pengujian menunjukkan bahwa dengan dataset positif negatif, *Max TF-IDF* dapat sangat membantu meningkatkan F-Measure dan akurasi dari metode KNN *Euclidean Distance*.

Analisis kedua akan membandingkan hasil penerapan metode *Max TF-IDF* terhadap KNN *Cosine Similarity*, untuk melihat seberapa besarkah pengaruh perubahan *Raw TF-IDF* menjadi *Max TF-IDF* terhadap metode KNN *Cosine Similarity*. Hasil pengujian akan ditampilkan pada Tabel 6.7 dan Tabel 6.8.

Tabel 6.7 Akurasi dan F-Measure *Raw TF-IDF* + KNN *Cosine* Dataset Positif Negatif

K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
11	83	77	80	0.80	0.80	0.80
13	87	77	82	0.82	0.82	0.82
15	87	80	83	0.83	0.83	0.83
17	90	77	83	0.84	0.83	0.83
19	90	77	83	0.84	0.83	0.83
21	90	80	85	0.85	0.85	0.85
23	90	80	85	0.85	0.85	0.85
25	90	80	85	0.85	0.85	0.85
27	90	80	85	0.85	0.85	0.85

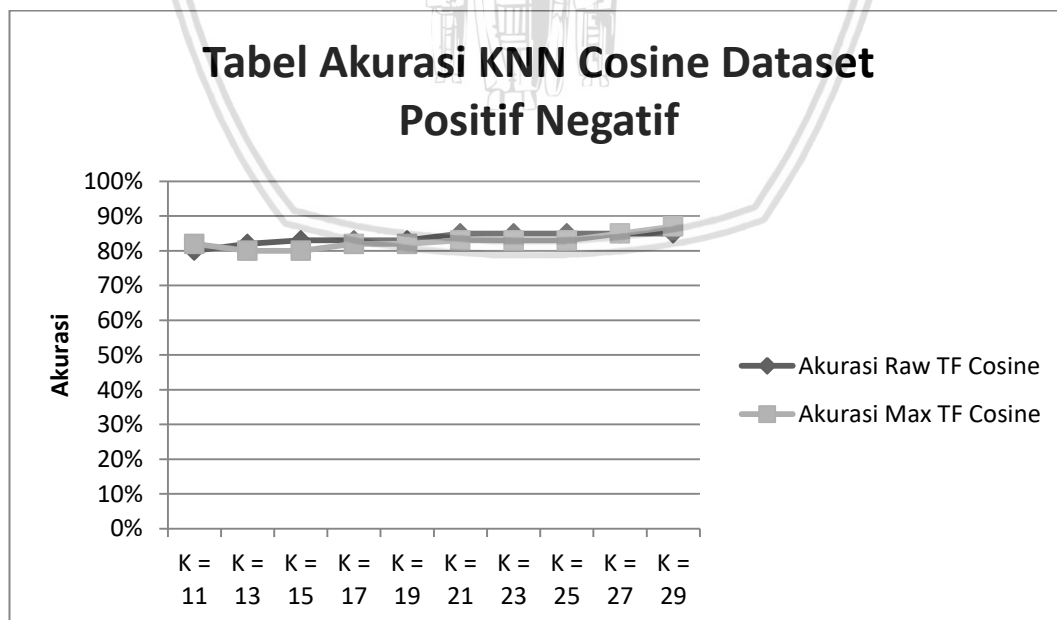


K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
29	90	80	85	0.85	0.85	0.85

Tabel 6.8 Akurasi dan F-Measure *Max* TF-IDF + KNN *Cosine* Dataset Positif Negatif

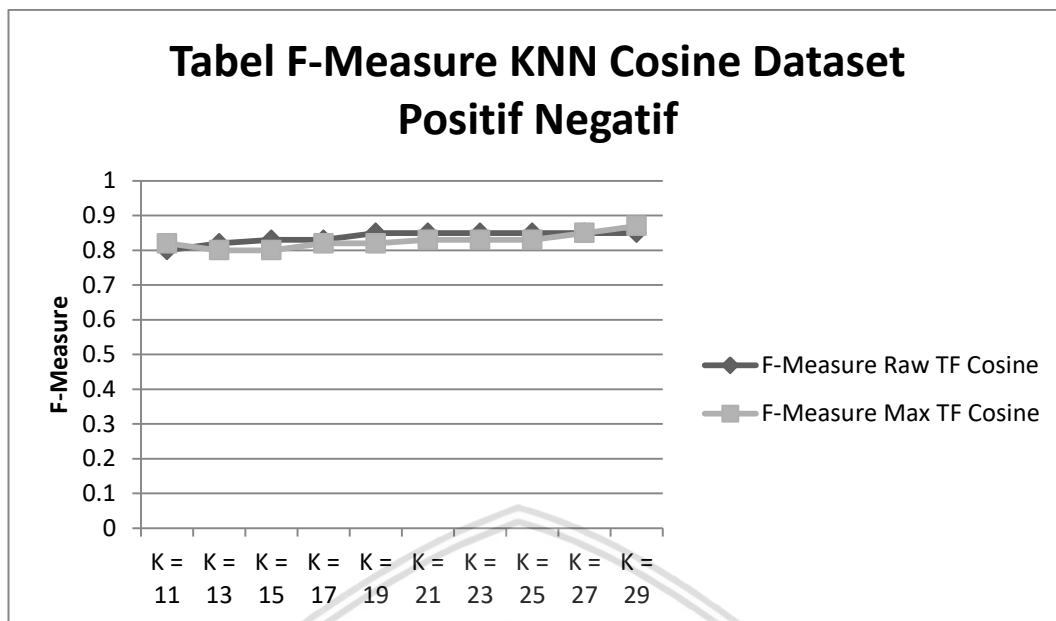
K	Accuracy (%)			Macro		
	Positive	Negative	Total	Precision	Recall	F-Measure
11	80	83	82	0.82	0.82	0.82
13	77	83	80	0.80	0.80	0.80
15	83	77	80	0.80	0.80	0.80
17	87	77	82	0.82	0.82	0.82
19	87	77	82	0.82	0.82	0.82
21	87	80	83	0.83	0.83	0.83
23	87	80	83	0.83	0.83	0.83
25	87	80	83	0.83	0.83	0.83
27	90	80	85	0.85	0.85	0.85
29	90	83	87	0.87	0.87	0.87

Kedua tabel diatas menunjukkan hasil penerapan *Raw* TF-IDF terhadap KNN *Cosine Similarity*. Dari kesepuluh pengujian yang dilakukan, terdapat 2 pengujian dimana *Max* TF-IDF memberikan nilai akurasi dan F-Measure yang lebih tinggi daripada *Raw* TF-IDF, 7 dimana *Raw* TF-IDF memberikan hasil yang lebih tinggi dan 1 dimana hasilnya benar benar sama. Perbandingan lebih jelas dapat dilihat pada Gambar 6.3 dan Gambar 6.4.



Gambar 6.3 Tabel perbandingan Akurasi *Raw* TF-IDF dan *Max* TF-IDF terhadap KNN *Cosine Similarity* Dataset Positif Negatif





Gambar 6.4 Tabel perbandingan F-Measure *Raw TF-IDF* dan *Max TF-IDF* terhadap KNN *Cosine Similarity* Dataset Positif Negatif

Gambar 6.3 dan Gambar 6.4 menunjukkan hasil pengujian dimana *Raw TF-IDF* dan KNN *Cosine Similarity* memberikan hasil yang relatif lebih tinggi dibandingkan *Max TF-IDF* dan KNN *Cosine Similarity*. Namun, pada nilai K Optimal, *Max TF-IDF* memberikan hasil yang lebih baik, walau tidak signifikan, dibandingkan dengan *Raw TF-IDF*. Pada pengujian KNN *Euclidean Distance* di awal, nilai akurasi K Optimal dapat meningkatkan akurasi sebesar 12% apabila dibandingkan dengan penggunaan *Raw TF-IDF*, namun pada KNN *Cosine Similarity*, peningkatan terjadi hanya sebesar 2%. Hal ini, dan fakta bahwa *Raw TF-IDF* dan KNN *Cosine Similarity* memberikan rata-rata hasil yang lebih tinggi daripada *Max TF-IDF* dan KNN *Cosine Similarity*, menunjukkan bahwa *Max TF-IDF* tidaklah berpengaruh positif terhadap KNN *Cosine Similarity*. Terdapat perubahan dengan nilai yang kecil, namun tidak ada perubahan yang sangat signifikan sehingga membuat metode *Max TF-IDF* ini tidak cocok apabila digunakan dengan KNN *Cosine Similarity* untuk meningkatkan akurasi dan F-Measure dari KNN *Cosine Similarity*.

Dari kedua metode yang sudah diuji diatas, didapatkan kesimpulan bahwa *Max TF-IDF* dapat meningkatkan dan membawa perubahan yang positif terhadap KNN *Euclidean Distance*, namun tidak membawa perubahan yang berarti dan signifikan terhadap metode KNN *Cosine Similarity*. Selain itu, secara umum, KNN *Cosine Similarity* memiliki akurasi dan F-Measure yang lebih tinggi daripada KNN *Euclidean Distance*, dengan akurasi yang tidak pernah menyentuh dibawah 80% untuk KNN *Cosine Similarity*, sedangkan KNN *Euclidean Distance* hanya sekali menyentuh nilai diatas 80%, yaitu pada nilai K Optimal dan menggunakan *Max TF-IDF*.

6.2 Dataset Panichella

Dari pengujian terhadap dataset positif negatif, diketahui bahwa *Max TF-IDF* tidak begitu berpengaruh terhadap metode *KNN Cosine Similarity*, namun secara signifikan dapat meningkatkan akurasi metode *KNN Euclidean Distance*, baik pada pengujian akurasi maupun F-Measure. Pada dataset kedua ini, dataset Panichella akan digunakan untuk menguji seberapa baik keempat metode tersebut dapat mengklasifikasikan dataset Panichella.

Setiap pengujian akan dilakukan menggunakan dataset Panichella yang memiliki 4 kelas, yaitu Problem Discovery, Information Seeking, Information Giving dan Feature Request. Pengujian diharapkan dapat memberikan hasil berupa akurasi dari setiap gabungan metode yang digunakan terhadap dataset, sehingga dapat dilihat kemampuan keempat metode tersebut dalam mengklasifikasikan dataset Panichella. Selain itu, analisis terhadap dataset Panichella sendiri akan dilakukan, melihat nilai akurasi dan F-Measure setiap kelas, untuk mencari apabila terdapat kelas tertentu yang memberikan hasil yang terlalu baik atau terlalu buruk, sehingga mempengaruhi nilai akurasi seluruh kelas.

6.2.1 Optimal K

Sama dengan proses pada dataset positif negatif, proses awal yang dilakukan adalah mencari nilai optimal K dari masing-masing metode. Sistem akan mencari nilai K yang paling optimal dari masing-masing gabungan *method*. Untuk *dataset* Panichella, total *dataset training* yang digunakan sebesar 400 data, sehingga nilai umum yang didapat adalah 20. Nilai K yang akan diujikan menggunakan 10 nilai ganjil disekitar nilai umum ini.

Pengujian pertama merupakan pengujian K Optimal menggunakan kedua *method* gabungan *Term Frequency - Inverse Document Frequency* (TF-IDF) dengan K-Nearest Neighbor (KNN). Hasil pengujian tertera pada Tabel 6.9 untuk akurasi dan Tabel 6.10 untuk F-Measure

Tabel 6.9 Akurasi Raw TF-IDF + KNN Euclidean Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	33	20	20	83	39
19	37	20	17	87	40
21	43	17	23	83	42
23	47	10	27	83	42
25	53	10	27	83	43
27	50	3	27	87	42
29	50	0	27	87	41
31	53	0	27	90	43
33	53	0	37	87	44
35	57	0	43	87	47

Tabel 6.10 F-Measure Raw TF-IDF + KNN Euclidean Dataset Panichella

K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
17	0.39	0.32	0.31	0.45	0.37
19	0.43	0.32	0.26	0.46	0.37
21	0.46	0.28	0.37	0.45	0.39
23	0.49	0.18	0.39	0.46	0.38
25	0.54	0.18	0.38	0.48	0.39
27	0.51	0.06	0.37	0.49	0.36
29	0.53	0	0.36	0.48	0
31	0.55	0	0.36	0.50	0
33	0.53	0	0.46	0.51	0
35	0.56	0	0.50	0.54	0

Hasil pada Tabel 6.9 dan Tabel 6.10 menunjukkan nilai akurasi dan F-Measure dari pengujian *Raw TF-IDF + KNN Euclidean* terhadap dataset Panichella. Nilai K yang paling optimal pada pengujian ini adalah K = 25 dengan akurasi sebesar 43%. Terdapat nilai akurasi yang melebihi 43%, yaitu pada K = 33 dan K = 35 dengan akurasi sebesar 44% dan 47%, namun nilai F-Measure tidak dapat diukur dikarenakan terdapat permasalahan dengan Precision dengan nilai K = 33 dan K = 35. Permasalahan akan dijelaskan pada subbab selanjutnya, dikarenakan subbab ini hanya berfokus untuk mencari nilai K optimal.

Pengujian selanjutnya dilakukan terhadap metode *Max TF-IDF* terhadap *KNN Euclidean Distance*. Hasil pengujian tertera pada Tabel 6.11 untuk akurasi dan Tabel 6.12 untuk F-Measure.

Tabel 6.11 Akurasi Max TF-IDF + KNN Euclidean Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	57	40	30	77	51
19	57	50	27	67	50
21	53	60	17	57	47
23	53	60	20	53	47
25	50	60	27	53	48
27	50	63	17	53	46
29	47	60	20	53	45
31	50	67	27	60	51
33	47	60	27	57	48
35	47	50	37	60	48

Tabel 6.12 F-Measure *Max TF-IDF + KNN Euclidean* Dataset Panichella

K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
17	0.57	0.34	0.45	0.66	0.50
19	0.57	0.38	0.40	0.66	0.50
21	0.54	0.40	0.27	0.63	0.46
23	0.54	0.40	0.31	0.60	0.46
25	0.54	0.39	0.38	0.64	0.49
27	0.55	0.39	0.26	0.65	0.46
29	0.52	0.37	0.32	0.64	0.46
31	0.56	0.42	0.41	0.69	0.52
33	0.52	0.39	0.41	0.63	0.49
35	0.52	0.36	0.52	0.60	0.50

Dari Tabel 6.11 dan Tabel 6.12, didapatkan nilai K yang paling optimal untuk metode *Max TF-IDF + KNN Euclidean Distance* terhadap dataset Panichella sebesar K = 31, dengan akurasi sebesar 51% dan F-Measure sebesar 0.52. Terdapat 2 K yang menghasilkan nilai akurasi 51%, yaitu K = 17 dan K = 31, namun K = 31 diambil karena nilai F-Measure yang dimiliki K = 31 lebih tinggi daripada K=17.

Pengujian ketiga dilakukan menggunakan metode *Raw TF-IDF + KNN Cosine Similarity*. Hasilnya akan ditampilkan pada Tabel 6.13 untuk nilai Akurasi dan Tabel 6.14 untuk nilai F-Measure.

Tabel 6.13 Akurasi *Raw TF-IDF + KNN Cosine* Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	83	43	70	67	66
19	80	47	70	67	66
21	77	47	73	63	65
23	80	43	70	63	64
25	83	47	70	67	67
27	80	40	70	67	64
29	77	30	70	67	61
31	80	27	67	63	59
33	87	33	70	60	63
35	83	27	70	60	60

Tabel 6.14 F-Measure Raw TF-IDF + KNN Cosine Dataset Panichella

K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
17	0.72	0.47	0.72	0.69	0.65
19	0.70	0.51	0.71	0.70	0.65
21	0.67	0.51	0.71	0.70	0.65
23	0.67	0.49	0.70	0.69	0.64
25	0.68	0.55	0.70	0.73	0.66
27	0.65	0.48	0.70	0.71	0.64
29	0.61	0.38	0.71	0.69	0.60
31	0.62	0.34	0.71	0.64	0.58
33	0.65	0.43	0.74	0.64	0.61
35	0.62	0.36	0.72	0.64	0.58

Tabel 6.13 dan Tabel 6.14 menunjukan hasil K optimal sebesar K = 25, dimana akurasi dari K = 25 adalah sebesar 67%, dan nilai F-Measurenya sebesar 0.66.

Pengujian terakhir dilakukan menggunakan metode *Max TF-IDF + KNN Cosine Similarity* terhadap dataset Panichella. Hasil akan ditampilkan pada Tabel 6.15 untuk nilai akurasi dan Tabel 6.16 untuk nilai F-Measure.

Tabel 6.15 Akurasi Max TF-IDF + KNN Cosine Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	73	47	70	67	64
19	77	47	67	67	64
21	73	47	70	67	64
23	80	43	67	67	64
25	80	43	67	67	64
27	80	37	70	67	63
29	80	33	70	67	63
31	77	30	67	63	59
33	87	33	70	60	63
35	87	20	67	60	58

Tabel 6.16 F-Measure Max TF-IDF + KNN Cosine Dataset Panichella

K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
17	0.66	0.48	0.72	0.70	0.64
19	0.68	0.49	0.70	0.69	0.64
21	0.66	0.49	0.70	0.71	0.64
23	0.67	0.49	0.69	0.70	0.64



K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
25	0.67	0.48	0.68	0.73	0.64
27	0.63	0.44	0.72	0.71	0.63
29	0.63	0.40	0.72	0.71	0.62
31	0.59	0.37	0.71	0.67	0.58
33	0.65	0.42	0.74	0.65	0.61
35	0.64	0.27	0.71	0.62	0.56

Tabel 6.15 dan Tabel 6.16 menunjukkan hasil pengujian dengan hasil akurasi yang sama besarnya dan F-Measure yang sama besarnya untuk K = 17, 19, 21, 23 dan 25. Nilai K yang diambil adalah nilai K yang mendekati peraturan umum, yaitu K = 21.

Dari keempat pengujian yang sudah dilakukan diatas, didapatkan nilai K yang mewakili masing masing metode pengujian, yaitu K = 25 untuk *Raw TF-IDF + KNN Euclidean Distance*, K = 31 untuk *Max TF-IDF + KNN Euclidean Distance*, K = 25 untuk *Raw TF-IDF + KNN Cosine Similarity* dan K = 21 untuk *Max TF-IDF + KNN Cosine Similarity*. Perbandingan akan tetap dilakukan untuk setiap 10 K yang sudah dihitung, namun nilai K optimal untuk masing masing metode akan menjadi acuan yang akan dibahas nantinya.

6.2.2 Akurasi dan F-Measure

Pada analisis ini, akan dibandingkan nilai akurasi dan F-Measure dari setiap metode, K dan kelas untuk mencari tahu pengaruh penerapan *Max TF-IDF* terhadap *KNN Euclidean Distance* dan *KNN Cosine Similarity*.

Analisis pertama akan membandingkan pengaruh penerapan *Max TF-IDF* terhadap *KNN Euclidean Distance*. Hasil pengujian *Raw TF-IDF* akan ditampilkan pada Tabel 6.17 untuk akurasi dan pada Tabel 6.18 untuk F-Measure, sedangkan *Max TF-IDF* akan ditampilkan pada Tabel 6.20 dan Tabel 6.21.

Tabel 6.17 Akurasi Raw TF-IDF + KNN Euclidean Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	33	20	20	83	39
19	37	20	17	87	40
21	43	17	23	83	42
23	47	10	27	83	42
25	53	10	27	83	43
27	50	3	27	87	42
29	50	0	27	87	41
31	53	0	27	90	43
33	53	0	37	87	44
35	57	0	43	87	47



Tabel 6.18 F-Measure Raw TF-IDF + KNN Euclidean Dataset Panichella

K	F-Measure				Macro
	PD	IS	IG	FR	F-Measure
17	0.39	0.32	0.31	0.45	0.37
19	0.43	0.32	0.26	0.46	0.37
21	0.46	0.28	0.37	0.45	0.39
23	0.49	0.18	0.39	0.46	0.38
25	0.54	0.18	0.38	0.48	0.39
27	0.51	0.06	0.37	0.49	0.36
29	0.53	0	0.36	0.48	0
31	0.55	0	0.36	0.50	0
33	0.53	0	0.46	0.51	0
35	0.56	0	0.50	0.54	0

Pada tabel 6.18, terlihat beberapa nilai F-Measure yang bernilai 0. Hal ini disebabkan karena adanya permasalahan pada nilai Precision yang bernilai 0 pada kelas *Information Seeking* (IS). Permasalahan dapat dilihat secara jelas pada Tabel 6.19 dibawah

Tabel 6.19 Precision dan Recall Information Seeking Raw TF-IDF + KNN Euclidean Dataset Panichella

K	Information Seeking				Precision	Recall	F-Measure
	TP	TN	FP	FN			
17	6	88	2	24	0.75	0.20	0.32
19	6	88	2	24	0.75	0.20	0.32
21	5	89	1	25	0.83	0.17	0.28
23	3	89	1	27	0.75	0.10	0.18
25	3	89	1	27	0.75	0.10	0.18
27	1	90	0	29	1	0.03	0.06
29	0	90	0	30	0	0	0
31	0	90	0	30	0	0	0
33	0	90	0	30	0	0	0
35	0	90	0	30	0	0	0

Pada Tabel 6.19, terlihat bahwa pada nilai K = 29, 31, 33 dan 35, nilai *True Positive* (TP) dan *False Positive* (FP) sebesar 0. Hal ini menunjukkan bahwa model sama sekali tidak mengklasifikasikan satupun data, baik benar ataupun salah, kedalam kelas *Information Seeking*. Nilai ini membuat nilai Precision kategori tersebut menjadi 0, sehingga berakhir dengan F-Measure yang juga 0. Hal ini menunjukkan bahwa terdapat permasalahan ketika sistem mencoba mengategorikan data kedalam kelas IS, permasalahannya sendiri dapat



bersumber dari sistem yang sulit mengkategorikan kelas, atau dataset yang sulit diklasifikasi dan dibedakan dari kelas lain.

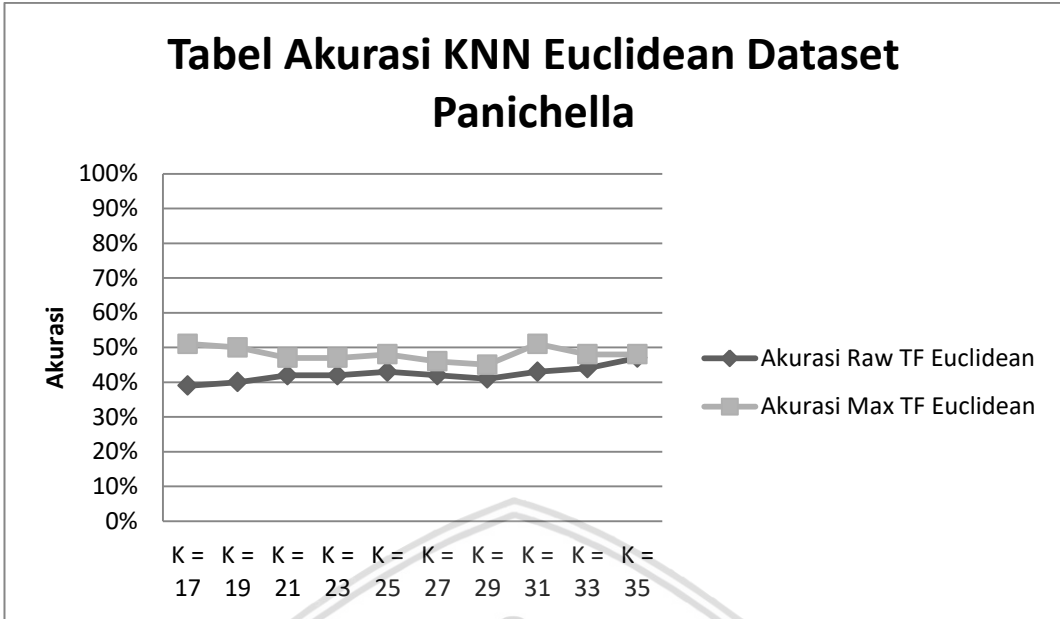
Tabel 6.20 Akurasi *Max TF-IDF + KNN Euclidean* Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	57	40	30	77	51
19	57	50	27	67	50
21	53	60	17	57	47
23	53	60	20	53	47
25	50	60	27	53	48
27	50	63	17	53	46
29	47	60	20	53	45
31	50	67	27	60	51
33	47	60	27	57	48
35	47	50	37	60	48

Tabel 6.21 F-Measure *Max TF-IDF + KNN Euclidean* Dataset Panichella

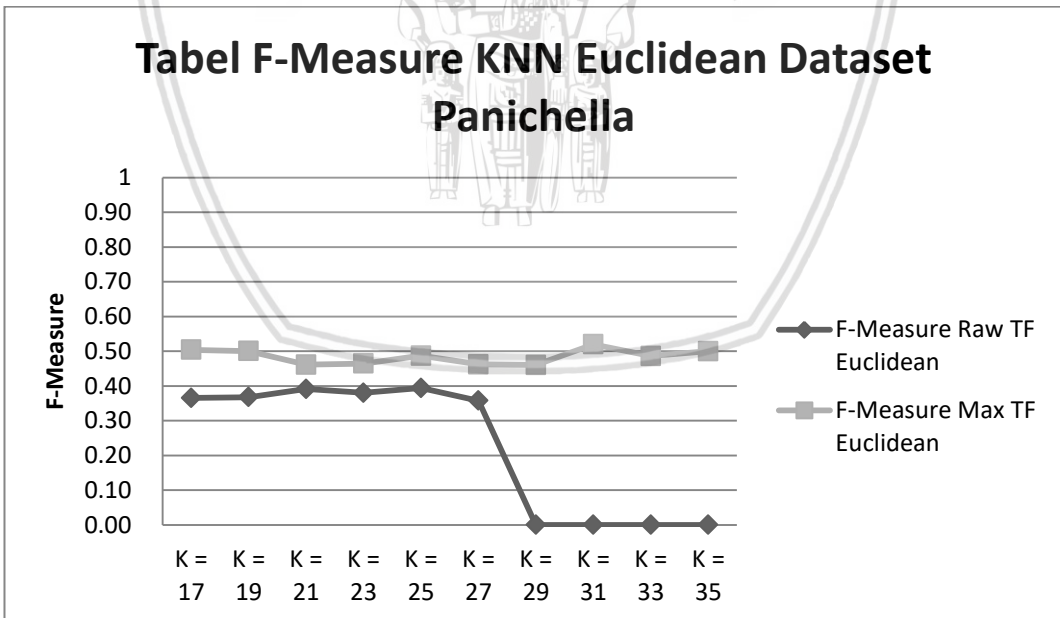
K	F-Measure				Macro F-Measure
	PD	IS	IG	FR	
17	0.57	0.34	0.45	0.66	0.50
19	0.57	0.38	0.40	0.66	0.50
21	0.54	0.40	0.27	0.63	0.46
23	0.54	0.40	0.31	0.60	0.46
25	0.54	0.39	0.38	0.64	0.49
27	0.55	0.39	0.26	0.65	0.46
29	0.52	0.37	0.32	0.64	0.46
31	0.56	0.42	0.41	0.69	0.52
33	0.52	0.39	0.41	0.63	0.49
35	0.52	0.36	0.52	0.60	0.50

Dari hasil pengujian, dapat dibandingkan akurasi dan F-Measure yang didapat untuk *KNN Euclidean Distance*, menggunakan kedua jenis TF-IDF. Dari 10 pengujian, semua 10 pengujian tersebut menghasilkan data dimana *Max TF-IDF + KNN Euclidean Distance* menghasilkan akurasi diatas *Raw TF-IDF + KNN Euclidean Distance*. Hal ini menunjukkan bahwa untuk dataset Panichella, sama dengan dataset positif negatif, *Max TF-IDF* lebih akurat dalam mengklasifikasikan data. Hasil perbandingan yang lebih jelas dapat dilihat pada Gambar 6.5 dan Gambar 6.6



Gambar 6.5 Tabel Akurasi KNN *Euclidean Distance* Dataset Panichella

Pada gambar 6.5, terlihat bahwa posisi dimana nilai Akurasi *Max* TF-IDF berada atas akurasi *Raw* TF-IDF pada semua percobaan. Hal ini menunjukkan bahwa penggunaan *Max* TF-IDF akan menghasilkan nilai yang lebih baik dalam kondisi apapun, apabila dibandingkn dengan *Raw* TF-IDF terhadap dataset Panichella.



Gambar 6.6 Tabel F-Measure KNN *Euclidean Distance* Dataset Panichella

Gambar 6.6 menunjukkan perbandingan antara F-Measure *Raw* TF-IDF KNN *Euclidean* dengan *Max* TF-IDF KNN *Euclidean*. Hasil yang serupa dengan Gambar 6.5 didapatkan pada gambar 6.6, dengan nilai F-Measure *Max* TF-IDF + KNN *Euclidean Distance* selalu berada di atas *Raw* TF-IDF + KNN *Euclidean Distance*.

Analisis ketiga akan membandingkan pengaruh penerapan *Max TF-IDF* terhadap *KNN Cosine Similarity*. Hasil pengujian *Raw TF-IDF* akan ditampilkan pada Tabel 6.22 untuk Akurasi dan Tabel 6.23 untuk F-Measure, sedangkan *Max TF-IDF* akan ditampilkan pada Tabel 6.24 untuk Akurasi dan Tabel 6.25 untuk F-Measure.

Tabel 6.22 Akurasi *Raw TF-IDF* + *KNN Cosine* Dataset Panichella

K	Accuracy				
	PD	IS	IG	FR	Total
17	83	43	70	67	66
19	80	47	70	67	66
21	77	47	73	63	65
23	80	43	70	63	64
25	83	47	70	67	67
27	80	40	70	67	64
29	77	30	70	67	61
31	80	27	67	63	59
33	87	33	70	60	63
35	83	27	70	60	60

Tabel 6.23 F-Measure *Raw TF-IDF* + *KNN Cosine* Dataset Panichella

K	F-Measure				Macro
	PD	IS	IG	FR	F-Measure
17	0.72	0.47	0.72	0.69	0.65
19	0.70	0.51	0.71	0.70	0.65
21	0.67	0.51	0.71	0.70	0.65
23	0.67	0.49	0.70	0.69	0.64
25	0.68	0.55	0.70	0.73	0.66
27	0.65	0.48	0.70	0.71	0.64
29	0.61	0.38	0.71	0.69	0.60
31	0.62	0.34	0.71	0.64	0.58
33	0.65	0.43	0.74	0.64	0.61
35	0.62	0.36	0.72	0.64	0.58

Tabel 6.22 dan Tabel 6.23 menunjukkan nilai akurasi dan F-Measure yang relatif lebih tinggi dibandingkan dengan hasil pengujian menggunakan *Raw TF-IDF* dan *KNN Euclidean Distance* pada Tabel 6.17 dan Tabel 6.18.

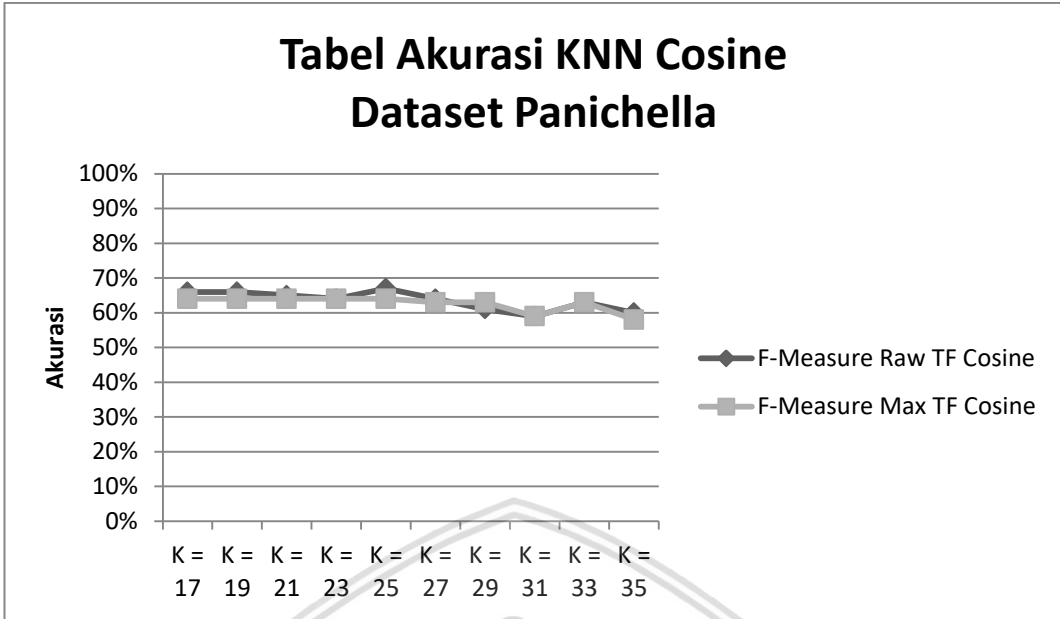
Tabel 6.24 Akurasi *Max TF-IDF + KNN Cosine* Dataset Panichella

K	Accuracy (%)				
	PD	IS	IG	FR	Total
17	73	47	70	67	64
19	77	47	67	67	64
21	73	47	70	67	64
23	80	43	67	67	64
25	80	43	67	67	64
27	80	37	70	67	63
29	80	33	70	67	63
31	77	30	67	63	59
33	87	33	70	60	63
35	87	20	67	60	58

Tabel 6.25 F-Measure *Max TF-IDF + KNN Cosine* Dataset Panichella

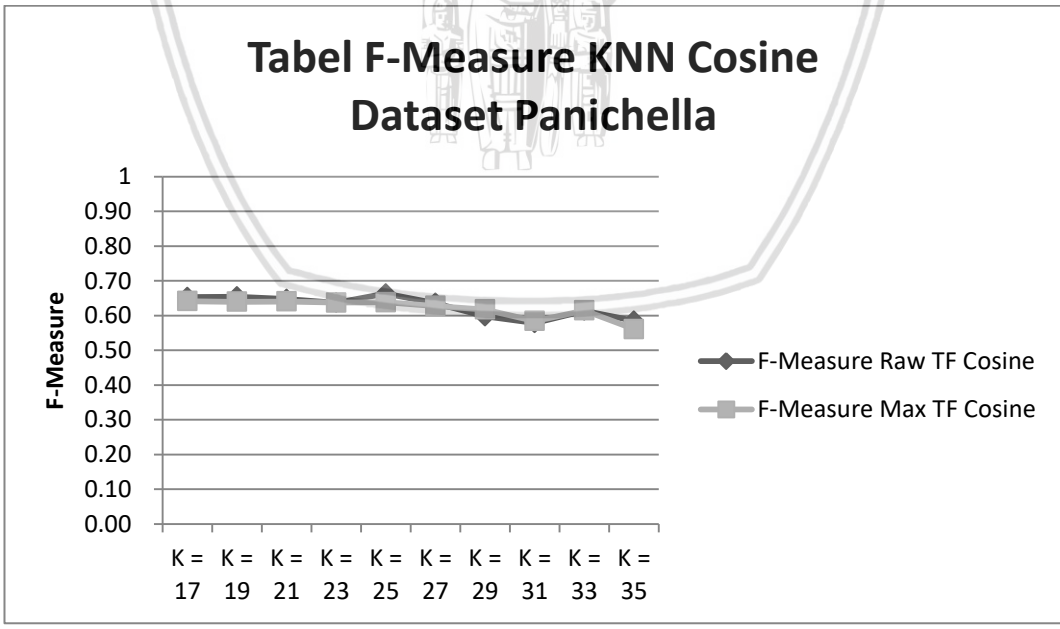
K	F-Measure				Macro
	PD	IS	IG	FR	F-Measure
17	0.66	0.48	0.72	0.70	0.64
19	0.68	0.49	0.70	0.69	0.64
21	0.66	0.49	0.70	0.71	0.64
23	0.67	0.49	0.69	0.70	0.64
25	0.67	0.48	0.68	0.73	0.64
27	0.63	0.44	0.72	0.71	0.63
29	0.63	0.40	0.72	0.71	0.62
31	0.59	0.37	0.71	0.67	0.58
33	0.65	0.42	0.74	0.65	0.61
35	0.64	0.27	0.71	0.62	0.56

Dari kedua hasil pengujian diatas, didapat hasil yang serupa dengan perbandingan pengujian kedua *KNN Cosine Similarity* menggunakan dataset positif negatif. Dari 10 pengujian, 6 hasil menunjukkan bahwa *Raw TF-IDF + KNN Cosine Similarity* menghasilkan akurasi yang lebih baik, 1 dimana *Max TF-IDF + KNN Cosine Similarity* menghasilkan nilai yang lebih baik dan 3 dengan nilai yang sama. Perbedaan akurasi terbesar hanyalah sebesar 3%, sehingga sama dengan pengujian dataset positif dan negatif, pada dataset Panichella ini, *KNN Cosine Similarity* menghasilkan hasil yang cenderung lebih baik tanpa menggunakan *Max TF-IDF*, dengan margin yang sangat kecil.



Gambar 6.7 Tabel Akurasi KNN *Cosine Similarity* Dataset Panichella

Pada gambar 6.7, terlihat bahwa posisi dimana nilai akurasi *Raw TF-IDF* berada atas akurasi *Max TF-IDF* pada 6 percobaan, namun dengan perbedaan yang sangat tipis. Akurasi juga berada diatas 60%, sehingga metode KNN *Cosine Similarity* ini masih menghasilkan nilai yang selalu lebih baik apabila dibandingkan dengan klasifier acak.

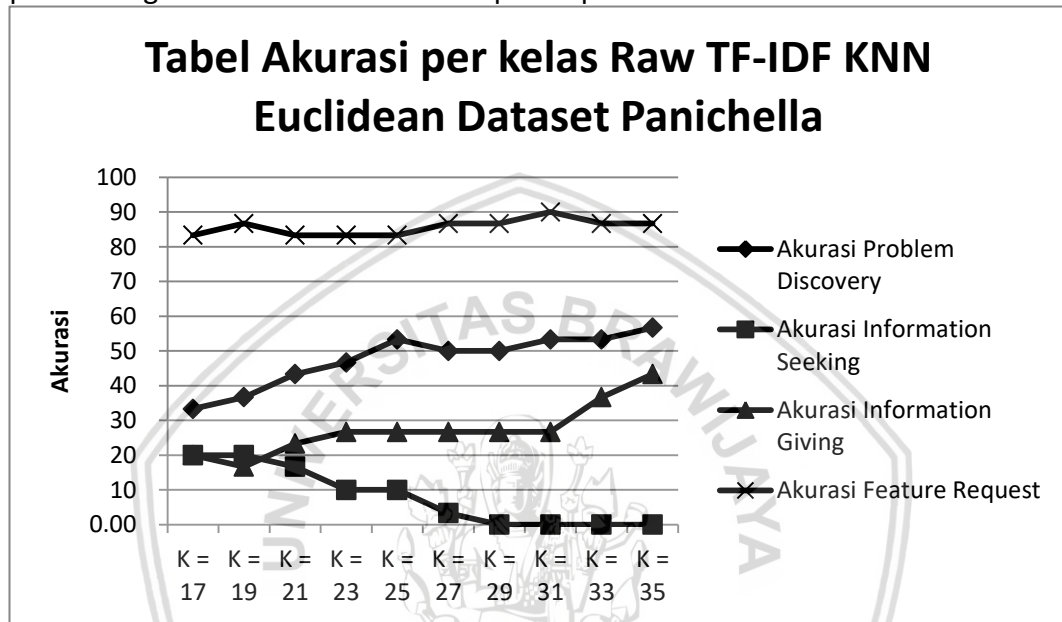


Gambar 6.8 Tabel F-Measure KNN *Cosine Similarity* Dataset Panichella

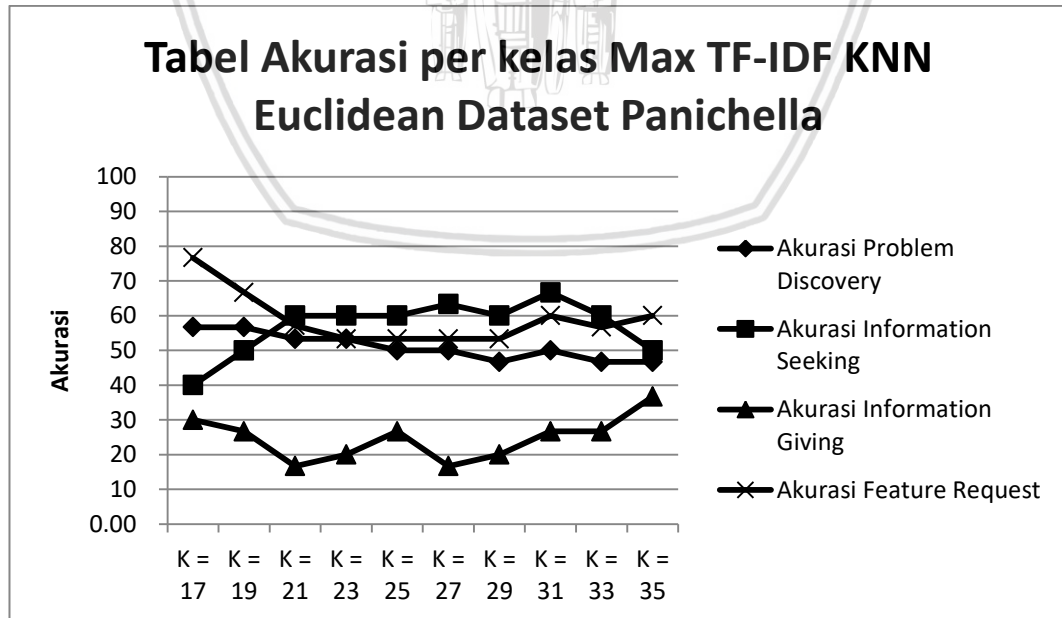
Gambar 6.8 menunjukkan perbandingan antara F-Measure *Raw TF-IDF* KNN *Cosine Similarity* dengan *Max TF-IDF* KNN *Cosine Similarity*. Hasil yang serupa dengan Gambar 6.7 didapatkan pada Gambar 6.8, dengan nilai F-Measure

Raw TF-IDF + KNN Cosine Similarity lebih sering Raw TF-IDF + KNN Cosine Similarity, dengan margin yang cukup tipis.

Dari keempat hasil pengujian, terlihat bahwa kelas Information Seeking sering mengalami permasalahan berupa akurasi yang rendah dan nilai F-Measure yang rendah apabila dibandingkan dengan kelas lain, kecuali pada pengujian Max TF-IDF + KNN Euclidean. Perbandingan akurasi untuk KNN Euclidean Distance akan ditampilkan pada Gambar 6.9 dan Gambar 6.10, sedangkan nilai perbandingan F-Measure akan ditampilkan pada Gambar 6.11 dan Gambar 6.12.



Gambar 6.9 Akurasi per kelas Raw TF-IDF + KNN Euclidean Distance

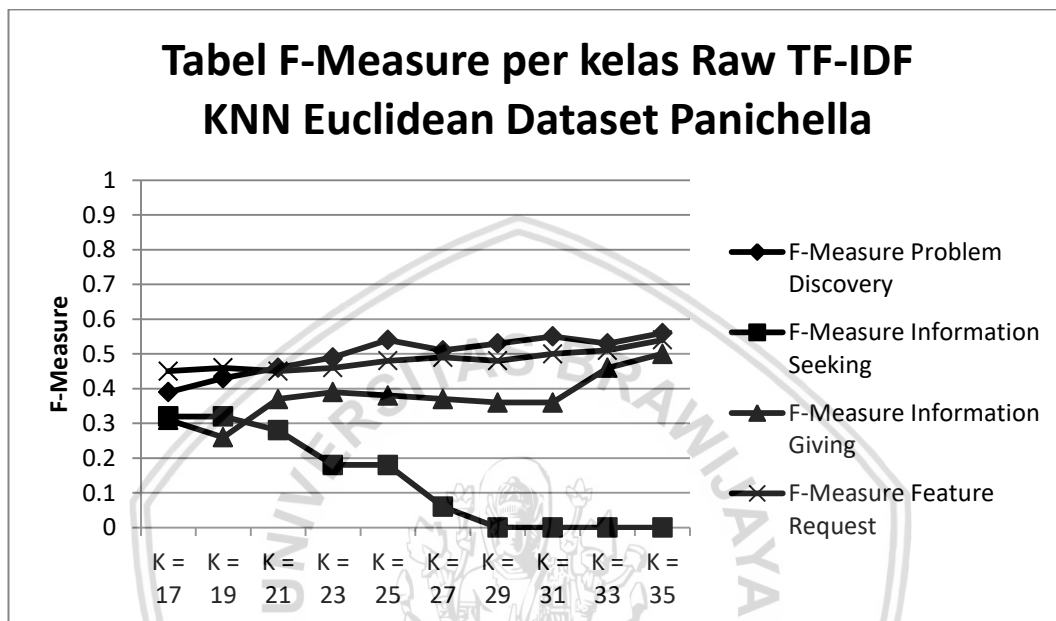


Gambar 6.10 Akurasi per kelas Max TF-IDF + KNN Euclidean Distance

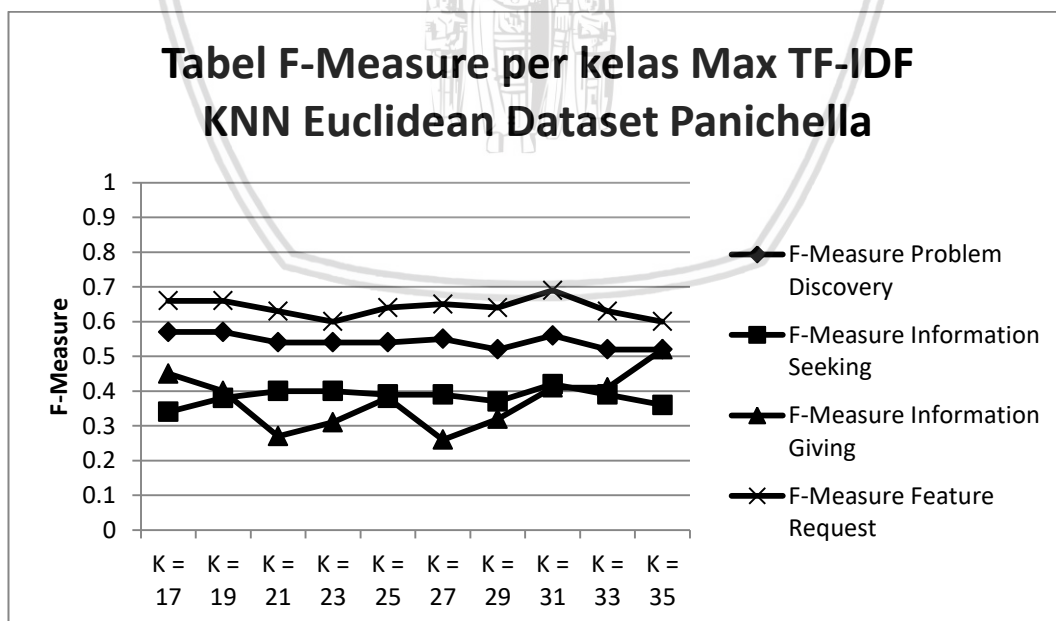
Gambar 6.9 dan Gambar 6.10 menunjukkan nilai perbandingan akurasi pada penerapan Max TF-IDF kedalam KNN Euclidean Distance. Terjadi perubahan



dimana nilai akurasi Information Seeking yang menjadi permasalahan pada penerapan KNN *Euclidean Distance* dan *Raw TF-IDF*. Pada penerapan *Max TF-IDF* dan KNN *Euclidean Distance*, nilai akurasi Information Seeking meningkat sehingga setara dengan nilai akurasi kelas kelas lainnya. Selain itu, tidak ada lagi permasalahan berupa nilai akurasi yang menyentuh angka 0, sehingga dapat dikatakan bahwa *Max TF-IDF* dapat menyelesaikan permasalahan yang dialami KNN *Euclidean Distance* dalam mengklasifikasikan dataset Panichella.



Gambar 6.11 F-Measure per kelas *Raw TF-IDF* + KNN *Euclidean Distance*

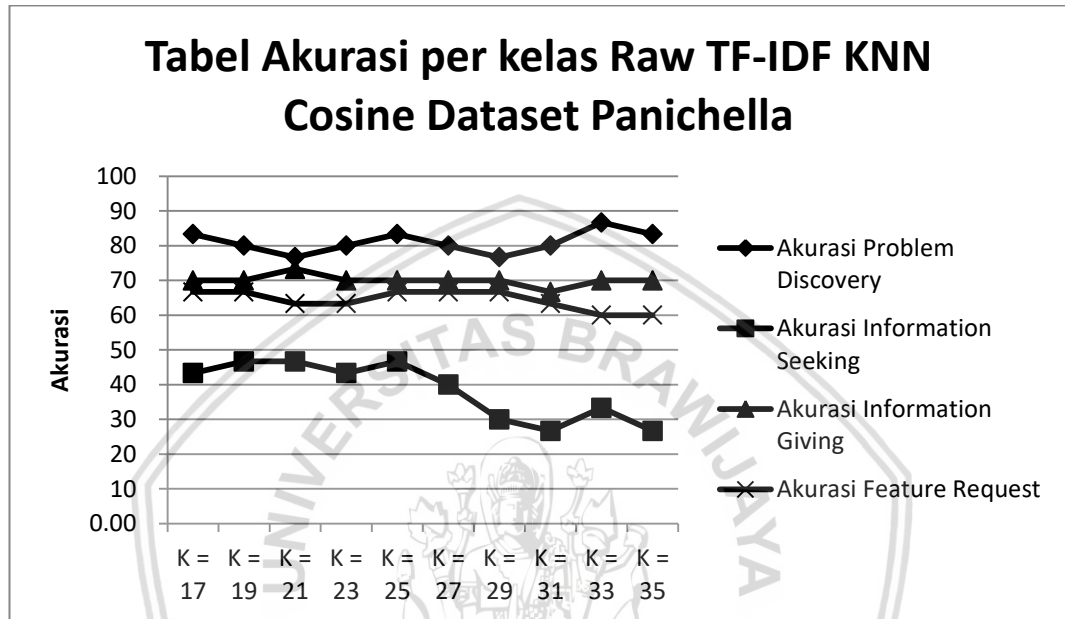


Gambar 6.12 F-Measure per kelas *Max TF-IDF* + KNN *Euclidean Distance*

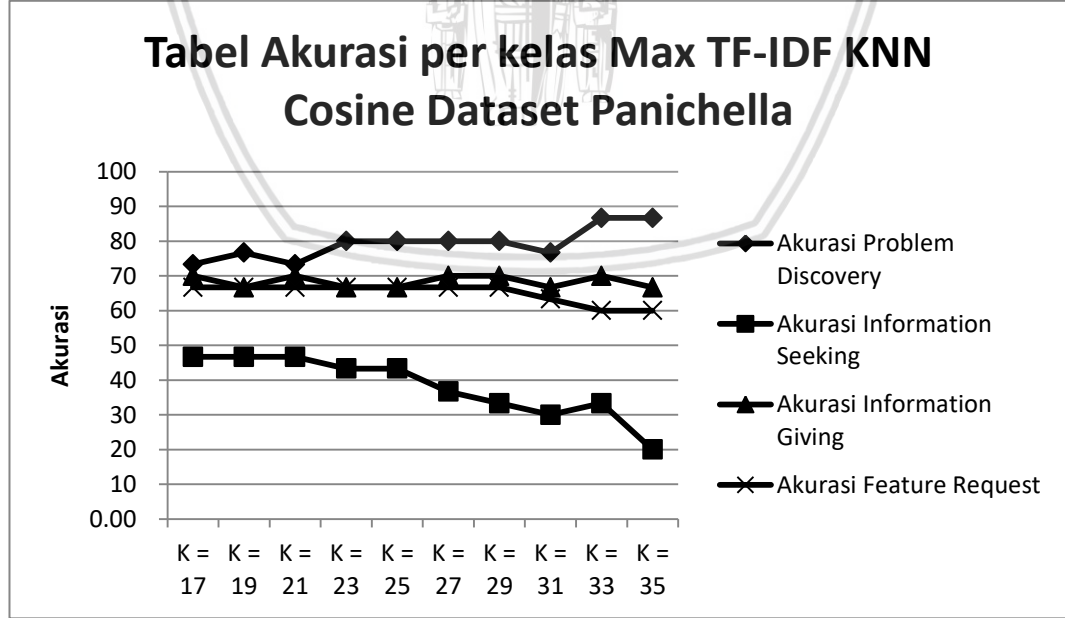
Gambar 6.11 dan Gambar 6.12 menunjukkan perbandingan hasil F-Measure untuk setiap kelas pada pengujian KNN *Euclidean Distance* menggunakan *Max TF-IDF*. Secara menyeluruh, terjadi kenaikan yang signifikan



pada saat *Max TF-IDF* diterapkan kedalam *KNN Euclidean*. Permasalahan yang sama juga diselesaikan oleh *Max TF-IDF* terhadap dataset Panichella dalam *KNN Euclidean Distance* ini, nilai F-Measure dari kelas Information Seeking tidak jatuh hingga menyentuh angka 0 pada *Max TF-IDF*, dimana nilai pernah jatuh ke angka 0 pada *Raw TF-IDF*. Secara menyeluruh, *KNN Euclidean Distance* sangat terbantu oleh *Max TF-IDF* apabila dibandingkan dengan *Raw TF-IDF*.

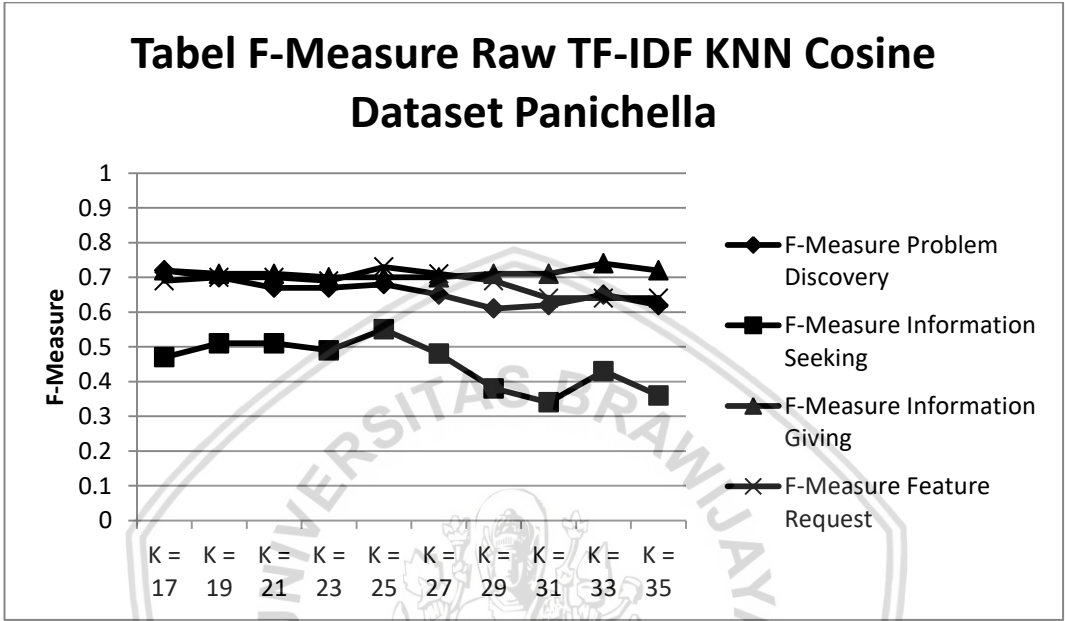


Gambar 6.13 Akurasi per kelas *Raw TF-IDF + KNN Cosine Similarity*

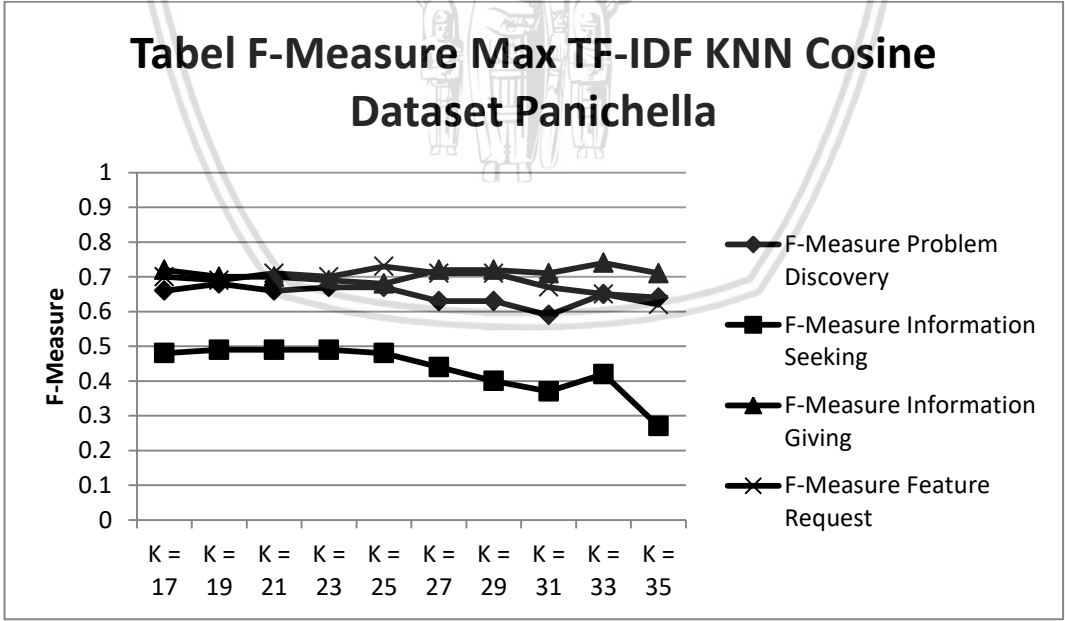


Gambar 6.14 Akurasi per kelas *Max TF-IDF + KNN Cosine Similarity*

Pada Gambar 6.13 dan Gambar 6.14, ditampilkan nilai Akurasi dari penerapan *Raw TF-IDF* dan *Max TF-IDF* terhadap *KNN Cosine Similarity*. Dari graph terlihat tidak terlalu ada perbedaan antara kedua akurasi, dimana hal ini sesuai dengan pengujian pada dataset positif negatif yang menyatakan bahwa *KNN Cosine Similarity* tidak terbantu dengan penerapan *Max TF-IDF* apabila dibandingkan dengan *Raw TF-IDF*.



Gambar 6.15 F-Measure per kelas *Raw TF-IDF* + *KNN Cosine Similarity*



Gambar 6.16 F-Measure per kelas *Max TF-IDF* + *KNN Cosine Similarity*

Pada Gambar 6.15 dan Gambar 6.16, ditampilkan nilai F-Measure dari penerapan *Max TF-IDF* terhadap *KNN Cosine Similarity*, dibandingkan dengan penerapan *Raw TF-IDF*, pada dataset Panichella. Mirip dengan perbandingan akurasi, F-Measure tidak mengalami perubahan signifikan dengan hasil yang

cenderung lebih baik pada *Raw* TF-IDF. Hal ini memperkuat kesimpulan bahwa *Max* TF-IDF tidak meningkatkan akurasi maupun F-Measure dari KNN *Cosine Similarity*, baik pada kasus dua kelas (dataset positif negatif) maupun pada kasus *multiclass* (dataset Panichella)



BAB 7 PENUTUP

Bab penutup berisi kesimpulan dan saran, dimana kesimpulan akan menjawab rumusan masalah yang sudah dirumuskan di rumusan masalah pada bab 1, dan saran akan berisi harapan mengenai penelitian kedepannya.

7.1 Kesimpulan

Berdasarkan hasil analisis dari pengujian yang tertera pada bab 6, dapat ditarik kesimpulan berupa:

1. Pada pengujian menggunakan dataset positif dan negatif, terdapat peningkatan akurasi sebesar 12% pada nilai K optimal masing masing metode pada saat pengujian dilakukan menggunakan *Max TF-IDF* dan KNN *Euclidean Distance* apabila dibandingkan dengan *Raw TF-IDF* dan KNN *Euclidean Distance*. Selain itu, dari 10 K yang diujikan, 8 diantaranya memberikan hasil dimana *Max TF-IDF* memberikan nilai akurasi yang lebih tinggi daripada *Raw TF-IDF* saat diaplikasikan kedalam KNN *Euclidean Distance*. F-Measure juga memberikan hasil serupa, dengan 8 percobaan menunjukkan nilai F-Measure yang lebih tinggi pada saat *Max TF-IDF* diterapkan kedalam KNN *Euclidean Distance* dibandingkan dengan penerapan *Raw TF-IDF* dan KNN *Euclidean Distane*. Hal ini membuktikan bahwa dengan menggunakan dataset positif negatif, *Max TF-IDF* sangat membantu meningkatkan nilai akurasi dan F-Measure dari KNN *Euclidean Distance*.

Hasil yang berbeda didapatkan saat pengujian dilakukan menggunakan KNN *Cosine Similarity*. Nilai akurasi yang didapat pada saat pengujian menggunakan *Max TF-IDF* menghasilkan nilai yang relatif lebih rendah daripada penggunaan *Raw TF-IDF*. Pengujian dengan nilai K optimal menghasilkan nilai akurasi dimana penggunaan *Raw TF-IDF* lebih baik daripada *Max TF-IDF*, walaupun dengan nilai perbedaan yang hanya sebesar 2%. Pengujian dengan nilai K lain juga menghasilkan nilai dimana 8 pengujian menggunakan *Raw TF-IDF* menghasilkan nilai akurasi yang lebih tinggi, 1 nilai akurasi yang sama dan 1 nilai akurasi dimana *Max TF-IDF* menghasilkan nilai akurasi yang lebih tinggi. Namun perlu diperhatikan bahwa perbedaan akurasi antar hasil percobaan pada KNN *Cosine Similarity* memiliki perbedaan hanya sebesar 2% dan 3%, membuat penerapan *Max TF-IDF* tidak membantu, bahkan sedikit mengurangi akurasi dari penerapan *Raw TF-IDF* terhadap KNN *Cosine Similarity*. Hal ini membuktikan bahwa penerapan *Max TF-IDF* tidaklah efektif apabila digunakan terhadap KNN *Cosine Similarity*, dengan dataset positif negatif.

2. Pada pengujian menggunakan dataset Panichella dan KNN *Euclidean Distance*, penggunaan *Max TF-IDF* menghasilkan nilai akurasi yang selalu



berada di atas *Raw TF-IDF* dari 10 percobaan, dengan perbedaan sebesar 4% pada kedua *K* optimal dengan *F-Measure* yang tidak diperhatikan, dan 8% ketika memperhitungkan *F-Measure* dimana terdapat beberapa kasus dengan nilai *F-Measure* 0 pada *Raw TF-IDF* dan *KNN Euclidean Distance*. Nilai *F-Measure* dari *Max TF-IDF* juga selalu berada di atas *Raw TF-IDF* untuk semua percobaan, membuktikan bahwa *Max TF-IDF* selalu menghasilkan nilai akurasi dan *F-Measure* yang lebih baik apabila dibandingkan dengan *Raw TF-IDF* pada percobaan *KNN Euclidean Distance* terhadap dataset Panichella.

Pada percobaan dataset Panichella terhadap *KNN Cosine Similarity*, didapati hasil yang serupa dengan percobaan pada dataset positif negatif. 6 dari 10 percobaan menunjukkan bahwa nilai akurasi *Raw TF-IDF* lebih tinggi dari nilai akurasi *Max TF-IDF*. Perbedaan pada masing masing nilai *K* optimal sebesar 3%, dengan perbedaan terjauh pada setiap *K* juga sebesar 3%. Hal ini sekali lagi membuktikan bahwa penggunaan *Max TF-IDF* terhadap *KNN Cosine Similarity* tidak berhasil meningkatkan nilai akurasi dari *KNN Cosine Similarity* terhadap dataset Panichella.

3. Pengujian menggunakan dataset positif negatif memberikan hasil perbandingan dengan pola yang relatif sama dengan pengujian menggunakan dataset Panichella. Pengujian terhadap dataset positif negatif menghasilkan nilai akurasi antara 60% - 82% menggunakan *KNN Euclidean Distance*, dan 80% - 87% menggunakan *KNN Cosine Similarity*. Pada dataset Panichella, nilai akurasi yang didapat berada diantara nilai 39% - 51% menggunakan *KNN Euclidean Distance*, dan 59% - 67% menggunakan *KNN Cosine Similarity*. Hal ini menunjukkan bahwa dalam semua dataset, walaupun *Max TF-IDF* dapat meningkatkan performa *KNN Euclidean Distance*, *KNN Cosine Similarity* selalu menghasilkan nilai yang lebih baik dari *KNN Euclidean Distance*, baik menggunakan *Raw TF-IDF* ataupun *Max TF-IDF*.

7.2 Saran

Saran pada penelitian ini untuk mengembangkan atau melakukan perbaikan adalah sebagai berikut:

1. Perubahan *Stopword*. *Stopword* yang digunakan pada penelitian ini masih menggunakan *Stopword* yang sudah disediakan. Apabila dapat dikembangkan *Stopword* pribadi yang didapat dari pengolahan dataset Panichella, akurasi dan *F-Measure* dataset Panichella dapat meningkat secara menyeluruh.
2. Perubahan dataset. Dataset yang digunakan memiliki nilai tetap sebesar 130 per kelas. Peningkatan dapat memperhalus nilai akurasi yang didapat, dan menampilkan hasil pengujian yang lebih *valid*.

DAFTAR PUSTAKA

- Baratloo, et al., 2015. *Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity*. Tehran: Emergency 48-49.
- Bird, S., Loper, E., 2002. *NLTK: The Natural Language Toolkit*. Philadelphia: Association for Computational Linguistics.
- Bruno, T., Donko, D., Sasa, M., 2014. *KNN with TF-IDF Based Framework for Text Categorization*. 24th DAAAM International Symposium on Intelligent.
- Gaigole, et al., 2011. *Preprocessing Techniques in Text Categorization*. National Conference on Innovative Paradigms in Engineering & Tehcnology. International Journal of Computer Applications.
- Guzzi, et al., 2012. *Facilitating enterprise software developer communication with CARES*. Software Maintenance (ICSM), 2012 28th IEEE.
- Hassanat, et al., 2014. *Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach*. International Journal of Computer Science and Information Security, International Journal of Computer Science and Information Security, Vol. 12, No. 8.
- Krendzelak, M., Jakab, F., 2012. *Text Categorization with Machine Learning and Hierarchical Status*. Stary Smokovec: 2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA).
- Koehrsen, W., 2018. *Beyond Accuracy: Precision and Recall*. <<https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>>. [Diakses 20 Maret 2018].
- Lahitani, A., Permanasari, A., Setiawan, N., 2016. *Cosine Similarity to deTermine similarity measure: Studi case in online essay assessment*. Bandung: 2016 4th International Conference on Cyber and IT Service Management.
- Manning, C., Raghavan, P., Schutze, H., 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- Mulak, P., Talhar, N., 2013. *Analysis of Distance Measures Using K-Nearest Neighbor Algorithm on KDD Dataset*. Pune: Computer Department, AISSMS, Collage of Engineering.
- Na, S., Kang, I., Lee, J., 2015. *Improving Term Frequency Normalization for Multi-topical Documents, and Application to Language Modeling Approaches*. Advances in Information Retrieval Lecture Notes in Computer Science Volume 4956.
- Panichella, et al., 2015. *How can i Improve my App? Classifying User Reviews for Software Maintenance and Evolution*. Bremen: 2015 IEEE

International Conference on Software Maintenance and Evolution (ICSME).

Porter, M., 1980. *New models in probabilistic information retrieval*. London: British Library

Powers, D M W., 2011. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation*. Journal of Machine Learning Technologies.

Quiros, et al., 2018. *A KNN-based approach for the machine vision of character recognition of license plate numbers*. Penang: TENCON 2017 - 2017 IEEE Region 10 Conference.

Rai, P., 2011. *Supervised Learning: K-Nearest Neighbors and Decision Trees*. Machine Learning, University of Utah.

Rajaraman, A., Leskovec, J., Ullman, J., 2011. *Mining of Massive Datasets*. Cambridge University Press.

Ramos, J., 2003. *Using TF-IDF to DeTermine Word Relevance in Document Queries*. Department of Computer Science, Rutgers University.

Qamar, et al., 2008. *Similarity Learning for Nearest Neighbor Classification*. Pisa: IEEE.

