ANALISIS SENTIMEN PADA ULASAN *"LAZADA"* BERBAHASA INDONESIA MENGGUNAKAN *K-NEAREST NEIGHBOR* (K-NN) DENGAN PERBAIKAN KATA MENGGUNAKAN *JARO WINKLER DISTANCE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yane Marita Febrianti
NIM: 145150201111141

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

Penguji 1

Nama : Suprapto, S.T., M.T
Email : spttif@ub.ac.id
Laboratorium : Laboratory of Intelligent Computing and Visualization
Jabatan : Head of Programme Information System

Penguji 2

Nama : Ratih Kartika Dewi, S.T., M.Kom
Email : ratihkartikad@ub.ac.id
Laboratorium : Laboratory of Media, Games, and Mobile Technologies
Jabatan : Lecturer of Informatics Engineering

PENGESAHAN

ANALISIS SENTIMEN PADA ULASAN *"LAZADA"* BERBAHASA INDONESIA MENGGUNAKAN *K-NEAREST NEIGHBOR* (K-NN) DENGAN PERBAIKAN KATA MENGGUNAKAN *JARO WINKLER DISTANCE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yane Marita Febrianti
NIM : 145150201111141

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Indriati, S.T., M.Kom	Agus Wahyu Widodo, S.T., M.Cs
NIP : 19831013 201504 2 002	NIP : 19740805 200112 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, 19 Januari 2018

Yane Marita Febrianti

NIM : 145150201111141

DAFTAR RIWAYAT HIDUP

Data Pribadi

Nama	: Yane Marita Febrianti
Tempat, Tanggal Lahir	: Jombang, 28 Februari 1996
Jenis Kelamin	: Perempuan
Agama	: Islam
Alamat Lengkap	: RT 04/RW 02, Dsn. Kedungcaluk, Ds. Kedungbogo, Kec. Ngusikan, Kab. Jombang
Nomor Telepon	: 081252414307
Gol. Darah	: A
Tinggi Badan	: 160 cm
Berat Badan	: 60 kg

Latar Belakang Pendidikan

Pendidikan Formal

2000 – 2002	: TK Budi Raharjo, Jombang
2002 – 2008	: Sekolah Dasar Negeri Kedungbogo, Jombang
2008 – 2011	: Sekolah Menengah Pertama Negeri Ngusikan, Jombang
2011 – 2014	: Sekolah Menengah Atas Negeri 1 Jombang
2014 – 2018	: S1 Teknik Informatika Universitas Brawijaya Malang

Pendidikan Non-Formal

2006	: Pelatihan Kader Tiwisada, Dinas Pendidikan Kabupaten Jombang
2008	: Juara Harapan 1 Lomba Kader Kesehatan Remaja, Dinas Kesehatan Kabupaten Jombang
2009	: English Olympic Committee of East Java
2008 – 2009	: Aktif dalam kegiatan OSIS SMPN Ngusikan Jombang
2011 – 2013	: Aktif sebagai anggota MPK SMAN 1 Jombang
2015	: Peserta acara Advokasi Cepat Tanggap
2016	: - Volunteer PTIIK Goes To School - Panitia Studi Ekskursi Informatika - Staff Departemen Administrasi Eksekutif Mahasiswa Informatika

Skill dan Keterampilan

Menguasai Microsoft Office
Administrasi

Pengalaman Kerja

Praktik Kerja Lapangan di Balai Penelitian Tanaman Pemanis dan Serat (BALITTAS)
Tim Dokumentasi di Startup Biji Inovasi
Sekertaris Karang Taruna Desa Kedungbogo tahun 2015

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi ini. Salawat dan salam semoga senantiasa tercurah kepada suri tauladan Rasullulah SAW yang telah memberikan ajaran berupa akhlakul karimah. Penulis sangat bersyukur, skripsi dengan judul "Analisis Sentimen Pada Ulasan *"Lazada"* Berbahasa Indonesia Menggunakan *K-Nearest Neighbor* (K-NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance*" dapat terselesaikan dengan baik sebagai syarat memperoleh gelar sarjana pada Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.

Dalam pembuatan skripsi ini tentunya masih banyak kekurangan dan kesalahan, penulis sangat mengharapkan saran yang bersifat membangun dari semua pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada :

1. Orangtua saya, yakni Ayah dan Ibu saya yang telah memberikan perhatian, kasih sayang, cinta, dukungan moral, materi, semangat, serta doa yang senantiasa ditujukan kepada saya demi kelancaran pembuatan skripsi ini. Dan juga keluarga besar Almarhum Kakek saya Suratno yang selalu menghibur dan memberikan motivasi untuk tidak mudah lelah dalam menjalani hidup.
2. Ibu Indriati, S.T, M.Kom selaku Pembimbing I yang telah banyak memberikan ilmu, saran, bimbingan, dukungan, serta perhatian bagi mahasiswa bimbingannya agar tetap semangat mengerjakan skripsi agar dapat selesai tepat waktu.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Pembimbing II yang juga telah memberikan ilmu, bimbingan, arahan, dukungan, serta berbagai macam masukan untuk menyelesaikan skripsi ini.
4. Bapak dan Ibu dosen yang telah memberikan ilmu selama penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya, beserta segenap karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang membantu dalam kelancaran pelaksanaan skripsi dari awal hingga pendaftaran sidang.
5. Sahabat-sahabat saya ketika menempuh pendidikan di Fakultas Ilmu Komputer yang tidak pernah bosan untuk mendengarkan keluh kesah saya selama mengerjakan skripsi dan selalu memberikan dukungan yakni Nurina Savanti W.G, Hamim Fathul Aziz, Eka Yuni .D, Vina Meilia, Amelia A. Putri, Anggita Mahardika, Mahdarani Dwi .L, Dwi Qunita Putri .A.P, serta Alfan Nur Hidayat sebagai seseorang yang spesial bagi saya yang selalu menghibur, memberikan support, saran, dan selalu ada saat saya membutuhkan.
6. Teman-teman Kost di Sengguruh 23 yang sudah saya anggap sebagai keluarga sendiri yang selalu memberikan perhatian ketika saya dalam keadaan sakit, dan menciptakan suasana ceria ketika kebingungan sedang melanda.
7. Teman-teman perjuangan program studi Teknik Informatika angkatan 2014 yang memberikan dukungan, dan informasi berharga selama penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.

8. Serta semua pihak yang mungkin belum tersebutkan diatas karena penulis tidak dapat menyebutkan satu per satu, terimakasih yang telah membantu baik secara langsung maupun tidak langsung dalam pembuatan skripsi ini.

Penulis menyadari jika dalam pembuatan skripsi ini masih banyak kekurangan baik dalam penulisan maupun isi, penulis mengharapkan kritik maupun saran yang membangun dari semua pihak. Semoga dari kritik dan saran yang diberikan penulis dapat memperbaiki agar menjadi lebih baik dan lebih bermanfaat. Terimakasih.

Malang, 19 Januari 2018

Yane Marita Febrianti

febriantiyane28@gmail.com

ABSTRAK

Perkembangan sebuah teknologi informasi saat ini membawa dampak yang cukup besar terhadap pola hidup masyarakat salah satunya pada daya beli. Saat ini daya beli masyarakat lebih cenderung berbelanja secara *online* karena dianggap lebih mudah, dan menghemat waktu karena dengan duduk dan memegang *smartphone* mereka sudah dapat memiliki sesuatu yang diinginkan, namun bagaimana konsumen mengetahui jika barang yang akan dibeli bagus atau sebaliknya. Oleh karena itu muncul adanya suatu ulasan atau komentar pada setiap barang yang dijual. Ulasan pada suatu barang membawa pengaruh yang cukup besar terhadap daya beli konsumen untuk mengetahui kualitas barang tersebut, tidak heran jika sebuah ulasan menjadi salah satu tujuan utama yang dilihat oleh konsumen setelah harga. Namun, tidak semua ulasan yang diberikan konsumen dapat dimengerti oleh konsumen lain dikarenakan penggunaan kata yang disingkat, penggunaan bahasa modern, salah dalam mengetik huruf, dan tidak bakunya kata yang digunakan dalam penulisan ulasan. Dengan latar belakang diatas, peneliti mengusulkan pembuatan sistem Analisis Sentimen Pada Ulasan *"Lazada"* Berbahasa Indonesia Menggunakan *K-Nearest Neighbor* (K-NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance*. Pengujian berdasarkan nilai *precision*, *recall*, dan *accuracy* pada masing-masing analisis sentimen tanpa perbaikan kata, maupun dengan perbaikan kata. Hasil pengujian dengan nilai *accuracy* yang baik terdapat pada analisis sentimen dengan perbaikan kata yakni 76 %, dengan nilai *precision* 0,76, dengan nilai *recall* yang optimal yaitu 1.

Kata Kunci : analisis sentimen, perbaikan kata, *K-Nearest Neighbor* (K-NN), *Jaro Winkler Distance*, *accuracy*, *precision*, *recall*

ABSTRACT

The development of an information technology currently carries a considerable impact against the pattern of life one on purchasing power. The current purchasing power are more likely to shop online because it's considered easier, it saved time because they sit and they hold their smartphone can already have something desirable, but how does a consumer know if the items to be purchased good or otherwise. Therefore it appears there is a review or comment on any goods sold. Review on items bring considerable influence against the purchasing power of consumers to know the quality of the goods, does not be surprised if a review into one of the main goals being viewed by consumers after the price. However, not all reviews provided the consumers can be understood by other consumers due to use the word is abbreviated, it use modern languages, in typing letters, the researcher proposes the creation of a system Analysis of Sentiment on the Reviews "Lazada" Berbahasa Indonesia Using the K-Nearest Neighbor (K-NN) and Repair Word Using Jaro Winkler Distance. Testing based on the value of precision, recall, and accuracy at each analysis sentiment without repair word, or with repair word. The test result with good accuracy value is present on the analysis sentiment with repair word is 76 %, with value of precision 0,76, and recall optimal 1.

***Keyword** : analysis of sentiment, repair word, K-Nearest Neighbor (K-NN), Jaro Winkler Distance, accuracy, precision, recall*

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi ini. Salawat dan salam semoga senantiasa tercurah kepada suri tauladan Rasullulah SAW yang telah memberikan ajaran berupa akhlakul karimah. Penulis sangat bersyukur, skripsi dengan judul "Analisis Sentimen Pada Ulasan *"Lazada"* Berbahasa Indonesia Menggunakan *K-Nearest Neighbor* (K-NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance*" dapat terselesaikan dengan baik sebagai syarat memperoleh gelar sarjana pada Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.

Dalam pembuatan skripsi ini tentunya masih banyak kekurangan dan kesalahan, penulis sangat mengharapkan saran yang bersifat membangun dari semua pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada :

1. Orangtua saya, yakni Ayah dan Ibu saya yang telah memberikan perhatian, kasih sayang, cinta, dukungan moral, materi, semangat, serta doa yang senantiasa ditujukan kepada saya demi kelancaran pembuatan skripsi ini. Dan juga keluarga besar Almarhum Kakek saya Suratno yang selalu menghibur dan memberikan motivasi untuk tidak mudah lelah dalam menjalani hidup.
2. Ibu Indriati, S.T, M.Kom selaku Pembimbing I yang telah banyak memberikan ilmu, saran, bimbingan, dukungan, serta perhatian bagi mahasiswa bimbingannya agar tetap semangat mengerjakan skripsi agar dapat selesai tepat waktu.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Pembimbing II yang juga telah memberikan ilmu, bimbingan, arahan, dukungan, serta berbagai macam masukan untuk menyelesaikan skripsi ini.
4. Bapak dan Ibu dosen yang telah memberikan ilmu selama penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya, beserta segenap karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang membantu dalam kelancaran pelaksanaan skripsi dari awal hingga pendaftaran sidang.
5. Sahabat-sahabat saya ketika menempuh pendidikan di Fakultas Ilmu Komputer yang tidak pernah bosan untuk mendengarkan keluh kesah saya selama mengerjakan skripsi dan selalu memberikan dukungan yakni Nurina Savanti W.G, Hamim Fathul Aziz, Eka Yuni .D, Vina Meilia, Amelia A. Putri, Anggita Mahardika, Mahdarani Dwi .L, Dwi Qunita Putri .A.P, serta Alfan Nur Hidayat sebagai seseorang yang spesial bagi saya yang selalu menghibur, memberikan support, saran, dan selalu ada saat saya membutuhkan.
6. Teman-teman Kost di Sengguruh 23 yang sudah saya anggap sebagai keluarga sendiri yang selalu memberikan perhatian ketika saya dalam keadaan sakit, dan menciptakan suasana ceria ketika kebingungan sedang melanda.
7. Teman-teman perjuangan program studi Teknik Informatika angkatan 2014 yang memberikan dukungan, dan informasi berharga selama penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.

8. Serta semua pihak yang mungkin belum tersebutkan diatas karena penulis tidak dapat menyebutkan satu per satu, terimakasih yang telah membantu baik secara langsung maupun tidak langsung dalam pembuatan skripsi ini.

Penulis menyadari jika dalam pembuatan skripsi ini masih banyak kekurangan baik dalam penulisan maupun isi, penulis mengharapkan kritik maupun saran yang membangun dari semua pihak. Semoga dari kritik dan saran yang diberikan penulis dapat memperbaiki agar menjadi lebih baik dan lebih bermanfaat. Terimakasih.

Malang, 19 Januari 2018

Yane Marita Febrianti

febriantiyane28@gmail.com

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Pada bab ini dijelaskan tentang masalah yang ada atau latar belakang yang mendasari untuk pembuatan sistem Analisis Sentimen Pada Ulasan "*Lazada"* Berbahasa Indonesia Menggunakan *K-Nearest Neighbor* (K-NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance*, serta beberapa teori yang mendukung dan berhubungan dengan penelitian yang akan dibuat.

1.1 Latar Belakang

Sebuah teknologi yang saat ini berkembang sangat pesat yakni internet membawa dampak sangat besar dalam masyarakat salah satunya adalah dalam kecenderungan berbelanja secara *online* pada sebuah toko yang memang dikhususkan untuk menjual secara *online* seperti *"Lazada"*, namun bagaimana konsumen mengetahui jika barang yang akan dibeli tersebut merupakan barang dengan kualitas yang bagus, maka dari permasalahan tersebut muncul adanya sebuah ulasan atau komentar dari konsumen pada masing-masing barang yang dijual yang bertujuan untuk membantu konsumen mengetahui kualitas barang yang akan dibeli. Ulasan dari konsumen di toko *online* seperti *"Lazada"* memang membawa pengaruh yang cukup besar terhadap daya beli konsumen berikutnya pada barang yang sama, dan juga merupakan suatu ukuran dimana konsumen lain akan melihat baik buruknya barang yang dijual tersebut berdasarkan ulasan dari pembeli sebelumnya. Tidak heran jika sebuah ulasan menjadi salah satu tujuan utama yang dilihat oleh konsumen setelah harga. Namun tidak semua ulasan yang diberikan konsumen dapat dimengerti oleh konsumen lain dikarenakan penggunaan kata yang seharusnya baku menjadi disingkat, kemudian penggunaan berbagai bahasa yang saat ini sedang *trend,* beberapa juga ada yang mungkin salah ketika mengetik beberapa huruf yang jarak antar huruf tersebut berdekatan, atau terdapat kata tidak baku yang digunakan dalam penulisan ulasan. Dari beberapa kesalahan dalam penulisan ulasan terdapat kesalahan yang disengaja maupun tidak sengaja atau yang biasa disebut *typo*. Pada kesalahan yang disengaja terdapat beberapa kesalahan yang memang konsisten yang dilakukan seseorang secara terus menerus dalam menuliskan suatu ulasan seperti contoh kata "bagus disingkat menjadi bgs", sehingga dibutuhkan sebuah perbaikan kata untuk kesalahan disengaja maupun tidak disengaja untuk mendapatkan nilai analisis sentimen dengan *accuracy* yang baik.

Dari permasalahan diatas, analisis sentimen memiliki peran yang sangat penting yakni untuk mengekstraksi polaritas ulasan yang diberikan antar kelas (positif dan negatif) dari ulasan teks, oleh karena itu dibutuhkan metode klasifikasi yang digunakan untuk mengekstraksi ulasan tersebut (Antinasari, Perdana, & Fauzi, 2017). Pengetahuan dan ilmu–ilmu yang telah dipelajari dalam *Text Mining* harus menjadi dasar dalam menyelesaikan sistem ini. Ada beberapa macam metode yang digunakan untuk klasifikasi teks yakni menggunakan metode yang umum digunakan yakni *Naive Bayes*, *K-Nearest Neighbor* (K-NN), *Minimum Edit Distance*, dan masih banyak lagi metode klasifikasi yang lain. Sedangkan metode yang digunakan dalam melakukan perbaikan kata diantaranya

metode *TF-IDF*, *Vektor Space*, *Search Matching*, ada juga metode lain yang ada pada *Aproximate String Matching* yakni yang pertama ada *Levenshtein Distance*, kedua *Hamming Distance*, ketiga *Damerau Levenshtein Distance*, dan yang keempat adalah *Jaro Winkler Distance.*

Terdapat penelitian untuk klasifikasi teks menggunakan *K-Nearest Neighbor* (K-NN) dengan alasan menggunakan penelitian K-NN menunjukkan *accuracy* yang cukup baik, dengan memberikan nilai *k*=3, K-NN dapat memberikan hasil persentase yang cukup baik yakni 88,29% (Samuel, Delima, & Rachmat, 2014). Selan itu, terdapat penelitian untuk membandingkan algoritme perbaikan kata dengan hasil penelitian yang menyebutkan jika algoritme *Jaro Winkler Distance* mampu memberikan nilai tertinggi dibanding algoritme pada *Aproximate String Matching* yang lainnya. Nilai yang diperoleh *Jaro Winkler Distance* yakni MAP 0,87 yang merupakan nilai terbesar dari ketiga algoritme yang lain, dimana nilai MAP terbagi menjadi empat kesalahan penulisan yaitu yang pertama kesalahan dalam penghapusan huruf dengan total nilai 0,92, kedua kesalahan penambahan huruf dengan total nilai 0,90, ketiga kesalahan penggantian huruf dengan total nilai 0,70, dan yang terakhir kesalahan penukaran huruf dengan total nilai 0,95 (Rocmawati & Kusumaningrum, 2015). Selanjutnya, terdapat penelitian yang juga menggunakan algoritme *Jaro Winkler Distance* yang digunakan untuk membandingkan kesamaan dokumen berbahasa indonesia dengan mendapatkan kesimpulan bahwa aplikasi akan berjalan dengan baik pada dokumen jika dokumen tersebut memiliki kemiripan dan urutan kata yang sama (Kurniawati, Pupitodjati, & Rahman, 2016).

Berdasarkan permasalahan dan alasan yang telah dipaparkan diatas, dan didukung oleh beberapa penelitian yang sudah dilakukan, maka pada penelitian ini akan dibangun sebuah sistem untuk melakukan analisis sentimen pada ulasan menggunakan metode *K-Nearest Neighbor* (K-NN) sebagai metode klasifikasi teks, kemudian hasil klasifikasi tersebut akan dibandingkan dengan hasil klasifikasi yang sudah melalui perbaikan kata menggunakan algoritme *Jaro Winkler Distance*. Dengan demikian sistem yang akan dibangun ini diharapkan dapat membantu seoptimal mungkin terhadap permasalahan untuk analisis sentimen khususnya pada ulasan dengan kata yang tidak baku dengan melalui perbaikan kata terlebih dahulu.

1.2 Rumusan Masalah

Berdasarkan latar diatas tentang permasalahan analisis sentimen dengan adanya beberapa ulasan kata tidak baku, maka didapatkan rumusan masalah sebagai berikut:

1. Bagaimana cara menerapkan algoritme *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance* untuk permasalahan analisis sentimen dan perbaikan kata pada ulasan *"Lazada"* ?
2. Bagaimana hasil *precision*, *recall*, dan *accuracy* yang diperoleh dari penerapan algoritme *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance* untuk analisis sentimen dan perbaikan kata pada ulasan *"Lazada"* ?

1.3 Tujuan

Tujuan dari perancangan sistem analisis sentimen dan perbaikan kata adalah sebagai berikut:

1. Menerapkan algoritme *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance* untuk analisis sentimen dan perbaikan kata pada ulasan *"Lazada"*.
2. Membangun suatu sistem yang mampu bekerja secara baik dengan hasil *accuracy* yang tinggi.
3. Memudahkan masyarakat dalam memahami maksud dari ulasan yang sebelumnya mempunyai makna yang ambigu.

1.4 Manfaat

Manfaat dari perancangan sistem analisis sentimen dan perbaikan kata, maka didapatkan manfaar sebagai berikut:

- Bagi Penulis
  - Dengan pembuatan sistem ini penulis mendapatkan ilmu untuk menerapkan pengetahuan baik yang sudah didapatkan selama melakukan perkuliahan maupun pengalaman-pengalaman baru yang sebelumnya belum pernah didapatkan.
  - Dengan pembuatan sistem ini penulis mendapatkan ilmu dalam memahami cara mengimplementasi algoritme *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance* untuk analisis sentimen dan perbaikan kata dalam ulasan berbahasa Indonesia.
- Bagi Pengguna
  - Sistem ini dapat memudahkan masyarakat umum dalam memahami maksud dari isi ulasan toko *online*.
  - Sistem ini dapat digunakan kapanpun dan dimanapun pengguna berada asalkan dapat terhubung dengan internet.

1.5 Batasan Masalah

Berdasarkan penjelasan – penjelasan sebelumnya, ruang lingkup / batasan yang digunakan dalam perancangan sistem analisis sentimen dan perbaikan kata adalah sebagai berikut:

1. Sistem ini dirancang untuk dapat digunakan oleh masyarakat umum agar lebih faham akan maksud dari kata yang memiliki arti ambigu.
2. *Input* yang digunakan untuk analisis sentimen dan perbaikan kata diambil dari ulasan toko *online "Lazada".*

1.6 Sistematika Pembahasan

Pada bab sistematika pembahasan ini digunakan untuk memberikan secara umum gambaran tentang apa saja yang nantinya akan dibahas pada sistem penelitian ini, sehingga secara sekilas pembaca akan mengetahui poin-poin penting dalam penelitian ini. Berikut sistematika dari penelitian ini:

BAB I PENDAHULUAN

Pada bab ini seperti pada umumnya yang pertama tentunya latar belakang, kedua membahas tentang rumusan masalah, ketiga tentang tujuan dibuatnya penelitian ini, yang keempat adalah manfaat penelitian baik dalam segi penulis maupun pembaca, kelima adalah batasan masalah yang digunakan untuk mengetahui batasan-batasan yang menjadi tujuan sistem, dan yang terakhir adalah sistematika pembahasan yang digunakan sebagai poin umum yang dilihat pembaca agar mengetahui poin-poin penting dalam penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Pada bab ini berisi teori yang sudah ada atau dari penelitian-penelitian yang sudah dilakukan dengan tujuan untuk mendukung serta menjadi pondasi dari sebuah sistem yang akan dibangun. Dasar teori sangat diperlukan demi tercapainya suatu tujuan sistem yang akan dibuat dan belajar dari kekurangan penelitian yang telah dibangun. Dasar teori yang akan digunakan pada penelitian ini adalah ulasan toko *online*, kesalahan penulisan, *text mining*, *information Retrieva*l, *Natural Language Processing*, metode *K-Nearest Neighbor* (K-NN), algorima *Jaro-Winkler Distance*.

BAB III METODE PENELITIAN

Pada bab ini terdapat langkah – langkah paling dasar yang akan digunakan dalam membangun penelitian ini, yang pertama adalah studi pustaka yakni dengan mempelajari teori dan penelitian terdahulu, kedua adalah analisis kebutuhan dimana kita harus tau apa saja yang akan sistem butuhkan dan pengguna butuhkan, ketiga adalah pengumpulan data yang digunakan sebagai bahas dalam pengujian, keempat adalah pengolahan data, keenam adalah perancangan sistem, ketujuh adalah implementasi, yang terakhir adalah pengujian, dan pengambilan kesimpulan.

BAB IV PERANCANGAN SISTEM

Pada bab ini berisi perancangan sistem secara detil yang digunakan dalam mengembangkan sistem untuk analisis sentimen dan perbaikan kata yang didasarkan dengan teori dan penelitian sebelumnya yang sudah dipelajari.

BAB V IMPLEMENTASI SISTEM

Bab ini menjelaskan tentang penerapan metode *K-Nearest Neighbor* (K-NN) dan algoritme *Jaro-Winkler Distance* dalam analisis sentimen dan perbaikan kata. Pada implementasi yang akan dibangun oleh sistem harus didasarkan dengan analisa dan perancangan yang sebelumnya sudah dilakukan dengan benar menurut teori yang digunakan.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini berisi hasil dari pengujian yang telah dilakukan kepada sistem sesuai dengan prosedur yang digunakan. Pengujian yang paling utama dan harus dilakukan adalah pengujian *accuracy* yang dihitung dari data uji dan data latih yang diberikan. Hasil pengujian berupa perbandingan *accuracy* antara hasil analisis sentimen ulasan tanpa perbaikan kata, dan analasis sentimen ulasan dengan perbaikan kata.

BAB VII PENUTUP

Pada bab ini berisi kesimpulan dari pembuatan sistem temu kembali informasi yaitu analisis sentimen dan perbaikan kata, dan juga terdapat saran yang diharapkan dapat menjadi masukan untuk menjadikan sistem ini lebih baik lagi untuk kedepannya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini dijelaskan beberapa teori yang berhubungan dengan penelitian yang akan dibuat yakni metode *K-Nearest Neighbor* (K-NN), dan metode *Jaro Winkler Distance*. Serta dasar yang digunakan dalam melakukan metode ini yakni *Text Mining*, dan beberapa penelitian yang berhubungan dengan implementasi dan pengujian sistem ini.

1.1 Kajian Pustaka

Berdasarkan penelitian yang sudah ada, penulis menggunakan beberapa penelitian sebelumnya yang merujuk kepada penelitian yang hampir sama untuk mendukung penelitian ini. Penelitian pertama dilakukan oleh Yosep Samuel, dkk pada tahun 2014 dengan judul, "Implementasi Metode *K-Nearest Neighbor* dengan *Decision Rule* untuk Klasifikasi Subtopik Berita". Penelitian kedua dilakukan oleh Yeny Rochmawati, dkk pada tahun 2015 dengan judul "Studi Perbandingan Algoritme Pencarian *String* dalam Metode *Approximate String Matching* Untuk Identifikasi Kesalahan". Penelitian ketiga oleh Anna Kurniawati, dkk tahun 2016 yang berjudul "Implementasi Algoritme *Jaro-Winkler Distance* Untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia". Kemudian penelitian keempat dilakukan oleh Hasnah Ariyana, dkk pada tahun 2016 dengan judul "Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode *Levenshtein Distance*".

Pada penelitian yang dilakukan Yosep Samuel, dkk mendapatkan hasil jika penggunaan *K-Nearest Neighbor* sebagai klasifikasi menunjukan persentasi yang baik, dengan nilai *k*= 3, menunjukan hasil persentase 88,29% itu artinya metode K-NN baik jika digunakan untuk klasifikasi teks. Berikutnya, penelitian dari Yeny Rochmawati, dkk pada tahun 2015 dengan judul "Studi Perbandingan Algoritme Pencarian *String* dalam Metode *Approximate String Matching* Untuk Identifikasi Kesalahan". Pada penelitian ini, peneliti membandingkan 4 algoritme yang digunakan dalam metode *Approximate String Matching* yaitu algoritme *Hamming Distance,* algoritme *Levenshtein Distance*, algoritme *Damerau Levenshtein Distance*, dan algoritme *Jaro Winkler Distance*. Hasil dari penelitian tersebut mengatakan jika algoritme *Jaro Winkler Distance* memiliki nilai tertinggi dibandingkan dengan ketiga algoritme lainnya (Rocmawati & Kusumaningrum, 2015). Penelitian selanjutnya oleh oleh Anna Kurniawati, dkk pada tahun 2016 dengan judul "Implementasi Algoritme *Jaro-Winkler Distance* Untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia". Pada penelitian ini digunakan untuk menghitung tingkat kesamaan antar dokumen dengan cepat, dengan menggunakan algoritme *Jaro Winkler Distance* untuk menghitung kesamaan dokumen dianggap baik, namun kekurangannya tidak mampu mendeteksi kemiripannya jika dokumen tersebut dalam urutan yang berbeda (Kurniawati, Pupitodjati, & Rahman, 2016). Penelitian selanjutnya oleh Hasnah Ariyana, dkk pada tahun 2016 dengan judul "Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode *Levenshtein Distance*". Pada penelitian ini

hampir sama dengan penelitian oleh Anna, namun metode yang digunakan dalam penelitian ini adalah *Levenshtein Distance* dengan tahapan *prepocessing* berupa *casefolding*, *tokenizing*, *filtering*, *stemming*, dan *sorting*. Hasil dari penelitian ini adalah nilai *similarity* yang tinggi yaitu diatas 85% sampai 100% untuk dokumen yang tingkat kemiripannya tinggi, sedangkan untuk dokumen dengan tingkat kemiripan yang rendah atau tidak berplagiat maka menghasilkan nilai *similarity* dibawah 40% (Na'firul, Sutardi, & Rahmat, 2016).

Perbedaan pada penelitian ini dengan penelitian sebelumnya adalah ditambahkannya proses perbaikan kata pada proses *prepocessing* dengan menggunakan algoritme *Jaro Winkler Distance*. Sistem yang dibuat akan memberikan hasil analisis sentimen tanpa perbaikan kata maupun analisis sentimen dengan perbaikan kata, dimana kita akan mengetahui perbandingan nilai akurasinya.

	Penulis	Objek	Metode	Hasil
1.	(Yosep Samuel, Rosa Delima, Antonius Rachmat, 2014)		Metode yang dipakai : a. *K-Nearest Neighbor* b. *Descision Rule*	Penggunaan *K-Nearest Neighbor* sebagai klasifikasi menunjukan persentasi yang baik, dengan nilai *k*=3, menunjukan hasil persentase 88,29% itu artinya metode K-NN baik jika digunakan untuk klasifikasi teks.
2.	(Yeny Rochmawati, dan Retno Kusumanungrum, 2015)	Membandingkan 4 algoritme yang ada pada metode *Approximate String Matching* yaitu algoritme *Hamming Distance,* algoritme *Levenshtein Distance*, algoritme *Damerau Levenshtein Distance*, dan algoritme *Jaro Winkler Distance*.	Metode yang dipakai Ekstraksi Kata dan *Mean Average Precision* : c. Pembentukan kamus d. Pengumpulan abstrak e. *Prepocessing* f. Pencocokan kata berdasarkan *distance*	Hasil dari penelitian ini bahwa algoritme *Jaro Winkler Distance* memiliki nilai tertinggi dibandingkan ketiga algoritme yang lain dengan nilai MAP 0,87 yang terbagi ke dalam empat jenis kesalahan penulisan yaitu jenis kesalahan penghapusan huruf 0,92, jenis kesalahan penambahan huruf 0,90, jenis kesalahan penggantian huruf 0,70 dan jenis kesalahan penukaran huruf 0,95.
3.	(Anna Kurniawati, Sulistyo Puspitodjati, dan Sazali Rahman, 2016)	menghitung tingkat kesamaan antar dokumen dengan cepat, untuk mengantisipasi adanya plagiat dalam penulisan dokumen.	Metode yang dipakai algoritme *Jaro Winkler Distance* : a. Menghitung panjang string b. Menemukan jumlah karakter yang sama di dalam dua string	Aplikasi yang dibuat telah berhasil menggunakan algoritme *Jaro-Winkler Distance* untuk mendukung kinerjanya. Dalam ujicobanya aplikasi dapat berjalan dengan baik untuk memeriksa kemiripan dokumen yang identik atau sama seratus persen. Hal

			$dj = \frac{1}{3} \times \left(\frac{m}{\|s1\|} + \frac{m}{\|s2\|} + \frac{m-t}{m}\right)$ Ket : m = jumlah karakter yang sama persis $\|s1\|$ = panjang string 1 $\|s2\|$ = panjang string 2 t = jumlah transportasi Jarak teoritis dua buah karakter : $\left(\frac{\max(\|s1\|, \|s2\|)}{s}\right) < -1$ c. Menemukan jumlah transposisi $dw = dj + \left(lp(1 - dj)\right)$ Ket : dj = Jaro distance untuk string s_1 dan s_2 l = panjang prefiks umum di awal string nilai maksimalnya 4 karakter. P = konstanta scaling faktor	ini dikarenakan urutan kata-kata yang dibandingkan sangat sesuai. Maka bisa disimpulkan bahwa aplikasi ini berjalan baik untuk dokumen yang memiliki kemiripan dan urutan kata yang sama.
4.	(Na'firul Hasna Ariyani, Sutardi, dan Rahmat Ramadhan, 2016)	Mengetahui kemiripan suatu dokumen untuk membantu menemukan plagiarisme.	Metode yang dipakai Metode *Levenshtein Distance* : a. Memasukkan dokumen asli dan dokumen pembanding b. *Prepocessing* : - *Case folding* - *Tokenizing* - *Filtering* - *Stemming* - *Sorting* c. Pencocokan menggunakan LD $f(i-1, j) + 1$	Hasil dari penelitian ini adalah nilai *similarity* yang tinggi yaitu diatas 85% sampai 100% untuk dokumen yang tingkat kemiripannya tinggi, sedangkan untuk dokumen dengan tingkat kemiripan yang rendah atau tidak berplagiat maka menghasilkan nilai *similarity* dibawah 40%.

			$f(i,j-1)+1$ $f(i-1,j-1)+1$ d. Pengukuran nilai *similarity* $\left\{1-\frac{diff}{\max(CS,ST)}\right\}\times 100$	

Tabel 2.1 Kajian Pustaka

1.2 Ulasan / Komentar

Merupakan suatu teks atau kalimat yang berisi penilaian atau komentar terhadap sesuatu hasil karya seseorang. Didalam sebuah ulasan biasanya terdapat berbagai macam pengungkapan seperti pujian, pertanyaan, tafsiran, maupun komentar-komentar yang lain. Pentingnya ulasan dalam produk karena sebagian besar dari konsumen *online* akan melihat terlebih dahulu ulasan yang diberikan konsumen terlebih dahulu sebelum membeli produk tersebut (Nanda, 2015).

1.3 Kesalahan Penulisan

Kesalahan penulisan merupakan suatu *human error* (kesalahan manusia) yang cukup sering dilakukan oleh seseorang dalam melakukan sebuah penulisan. Kesalahan penulisan sendiri dibagi menjadi dua yakni kesalahan penulisan yang dilakukan secara sengaja dan tidak sengaja. Untuk kesalahan penulisan yang tidak disengaja atau biasa disebut dengan kesalahan tipografi (*typographical error),* sedangkan untuk kesalahan penulisan yang dilakukan dengan sengaja karena kesalahan tersebut bersifat konsisten yang dilakukan secara berulang-ulang seperti kata "barang disingkat brg", "bagus disingkat bgs", serta beberapa kata lain yang sering dituliskan secara tidak baku (Fahma, Cholissodin, & Perdana, 2018).

1.4 *Text Mining*

1.4.1 Pengertian *Text Mining*

Populer dengan sebutan Pemrosesan Teks yang saat ini lebih dikenal dengan *Text Mining* merupakan satu dari banyak pengetahuan di bidang *Artificial Intelligence* dimana pada *Text Mining* ini menerapkan sebuah konsep yang disertai dengan teknik yang ada pada data mining yang digunakan dalam pencarian pola teks yang menggunakan proses ekstraksi terlebih dulu untuk mendapatkan informasi dari pola data teks yang tidak terstruktur. Pada penggunaan *Text Mining* teknik yang digunakan yakni melakukan pengambilan data berupa teks yang diambil dari beberapa sumber seperti dokumen, dengan tujuan untuk mencari inti dari dokumen tersebut sehingga dapat dilakukan proses analisis mengenai hubungan dan kemiripan dokumen satu dengan yang lain (Baskoro, Ahmad, & Furqon, 2015). Terdapat 7 tipe *Text Mining* yang ada saat ini yakni sebagai berikut (Abbott, 2013):

1. *Search and Information Retrieval* (*IR*)
   Penyimpanan dan sistem temu kembali dari dokumen teks, termasuk *search engines* dan *keyword search* serta teks mining seperti identifikasi kata *typo.*
2. *Document Clustering*
   Pengelompokan dan pengkategorian istilah, potongan kata, paragraf, atau dokumen yang menggunakan metode klasifikasi data mining.
3. *Document Classification*

Pengelompokan dan pengkategorian istilah, potongan kata, paragraf, atau dokumen yang menggunakan metode klasifikasi data mining, berdasarkan model yang telah terlatih.

4. *Web Mining*
   Data dan *Text Mining* pada internet dengan berfokus pada skala dan web yang saling berhubungan.
5. *Information Extraction* (*IE*)
   Identifikasi dan ekstraksi dari fakta-fakta yang relevan dan hubungan dari teks yang tidak terstruktur; proses pembuatan data terstruktur dari teks yang tidak terstruktur dan semi-terstruktur.
6. *Natural Language Processing* (*NLP*)
   Pemrosesan bahasa *low-level* (contohnya *tagging part of speech*); digunakan secara sinonim dengan Bahasa komputasional
7. *Concept Extraction*
   Pengelompokan kata dan frase ke dalam grup serupa secara semantik.

1.4.2 *Information Retrieval* (IR)

Information Retrieval (*IR*) atau biasa disebut Sistem Temu Kembali Informasi adalah pengetahuan atau sistem yang digunakan dalam menemukan kembali informasi yang diminta pengguna secara relevan sesuai dengan permintaan informasi yang dibutuhkan. Aplikasi yang sudah sering digunakan adalah mesin pencarian atau *search engine* dimana sistem harus terhubung dengan internet. Pada sistem temu kembali inormasi ini tidak semata-mata hanya untuk menemukan dokumen dengan *query* yang diberikan pengguna, namun sistem juga harus memperhatikan ukuran keefektifan suatu dokumen dengan menghitung *precision* dan *recall. Precision* merupakan jumlah dokumen relevan yang diperoleh dengan jumlah seluruh dokumen yang diperoleh mesin pencari. Sedangkan *Recall* merupakan penghitungan yang lebih sempit lagi karena dihitung dari jumlah dokumen relevan setelah dokumen yang relevan tersebut diperoleh (Fahma, Cholissodin, & Perdana, 2018).

$$Precision = \frac{|\{relevant\ documants\} \cap \{document\ retrieved\}|}{|\{documents\ retrieved\}|} \quad (2.1)$$

$$Recall = \frac{|\{relevant\ documents\} \cap \{documents\ retrieved\}|}{|\{relevant\ documents\}|} \quad (2.2)$$

$$\boldsymbol{Accuracy} = \frac{|\{\boldsymbol{relevant\ document + not\ relevant\ document}\}|}{|\{\boldsymbol{seluruh\ document}\}|} \quad (2.3)$$

1.4.3 *Natural Language Processing* (NLP)

Natural language Processing (NLP) merupakan penerapan ilmu linguistik pada ilmu komputer atau *computational linguistics* yang berkonsentrasi dalam menjembatani interaksi antara komputer dengan bahasa (alami) manusia. NLP

berusaha memecahkan masalah dengan berbagai aturan gramatika serta semantik dan merubah bahan menjadi representasi formal yang kemudian diproses komputer (Pustejovsky & Stubbs, 2012).

Menurut (Pustejovsky & Stubbs, 2012) menyebutkan tantangan-tantangan dalam penerapan NLP untuk memahami bahasa manusia, antara lain:

1. Penandaan kelas kata (*part-of-Speech tagging*). Kesulitan dalam proses penandaan pada kelas (kata benda, kata kerja, kata sifat dll) karena kata pada teks bergantung pada konteks penggunaannya.
2. Segmentasi teks (*text segmentation*) segmentasi sulit direpresentasikan pada bahasa yang tak memiliki pembatas kata spesifik contohnya bahasa Mandarin, Jepang dan Thailand.
2. Disambiguasi makna kata (*word sense disambiguation*). Beberapa kata memiliki makna lebih dari satu makna baik dalam bentuk homonim, polisemi.
3. Ambiguitas Sintaksis (*Syntactic Ambiguity*). Setiap bahasa memiliki banyak kemungkinan struktur kalimat. Untuk membuat struktur kalimat yang tepat perlu gabungan informasi secara semantik maupun kontektual.
4. Masukan yang tidak sempurna atau tidak teratur. Sering terjadi kesalahan pada ejaan, aksen bahasa, dan gramatikal dalam bahasa tulis menyulitkan pemrosesan bahasa alami.
5. Pertuturan (*speech act*). Gaya bahasa serta konteks menentukan maksud yang diinginkan. Karena kadang kalimat tidak manggambarkan maksud penulis.

1.4.4 Teks *Prepocessing*

Preprocessing dokumen terdiri dari *tokenisasi*, *stemming*, *part-of-speech* (POS) *tagger*, dan *stopwords*. Setelah dokumen dimasukkan ke dalam program, fungsi dari *preprocessing* kemudian membagi teks menjadi daftar kata-kata dengan menggunakan fungsi tokenisasi.

1.4.4.1 *Tokenisasi*

Proses *tokenisasi* ada tiga tahapan yaitu pertama melakukan penghapusan seluruh tanda baca yang terdapat dalam dokumen *termasuk* kata-kata yang tidak terlihat misalnya tab, enter kecuali spasi. Kedua, melakukan proses pemisahan kata-kata dari seluruh kalimat dari data dokumen, pemisahan ini menggunakan spasi sebagai tanda pemisahnya. Ketiga melakukan proses *stopword removal* yaitu penghapusan kata-kata yang biasanya tidak digunakan sebagai kata kunci dalam pencarian dokumen. Pada proses *tokenisasi* dilakukan juga *term weighting* atau pembobotan kata yaitu setiap kata dalam dokumen diberikan bobot berdasarkan banyaknya kata pada tiap dokumen. Setelah proses *tokeniasi* dan pembobotan kata selesai hasilnya berupa kumpulan kata-kata yang berasal dari

data dokumen, dan bobot tiap-tiap kata kemudian disimpan dalam database *term* (Sasongko & Hartati, 2011).

1.4.4.2 *Stopword Removal*

Stopword removal digunakan untuk membuang kata-kata yang sering muncul dan bersifat umum, kurang menunjukan relevansinya dengan teks. Semua *stopwords* dihapus untuk mencegah ambiguitas. *Stopword* pada dasarnya adalah serangkaian kata yang umum digunakan, tidak hanya dalam bahasa inggris namun dalam bahasa apapun. *Stopword* penting untuk banyak aplikasi karena, jika kata-kata yang sangat umum digunakan dalam bahasa dihapus, ada baiknya kita fokus pada kata-kata penting. Untuk penghapusan *stopword* kita menggunakan algoritme pencocokan pola. Algoritme pencocokan pola dapat dijelaskan dengan kode pseudo yang diberikan di bawah ini:

// Pseudo Code: lakukan

Jika (surat teks == huruf pola)

Bandingkan huruf berikutnya dari pola ke huruf berikutnya dari teks yang lain.

Pindahkan pola teks ke bawah dengan satu huruf sementara (seluruh pola Ditemukan atau akhir teks).

1.4.4.3 *Stemming*

Tahap *stemming* adalah tahap mencari akar kata dari tiap kata hasil penyaringan. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama (Dao & Simpson, 2006). Proses mencari akar kata dan menghilangkan imbuhan pada kata. *Stemming* bertujuan mengurangi variasi kata yang memiliki kata dasar sama.

1.5 *Term Frequency-Inverse Document Frequency* (TF-IDF)

Merupakan statistik numerik yang mencerminkan betapa pentingnya sebuah kata dalam sebuah dokumen dalam koleksi atau *corpus*. TF merupakan frekuensi *term* pada suatu dokumen, dimana *term* adalah sebuah kata atau frase. Frekuensi merupakan atribut penting *term* untuk membedakan dirinya dari *term* lain (Xia & Chai, 2011). Artinya, nilai TF dari *term* i adalah:

$$TF_i = tf_{ij} \tag{2.4}$$

Dimana tf_{ij} menunjukkan frekuensi *term* i dalam dokumen j. Karena jumlah frekuensi *term* mungkin sangat besar, rumus berikut ini juga sering digunakan untuk menghitung nilai TF (Xia & Chai, 2011).

$$TF_i = log2\ (tf_{ij}) \tag{2.5}$$

Sedangkan IDF pada dasarnya mengukur jumlah informasi yang diberikan oleh sebuah kata, yaitu apakah *term* itu biasa atau jarang terjadi di semua dokumen. Formula dasar diberikan oleh (Robertson, 2004). Sehingga menghasilkan formula IDF berikut ini:

$$IDF_i = log2\left(\frac{N}{n_j}\right) + 1 = log2(N) - log2(N_j) + 1 \quad (2.6)$$

Dimana N adalah jumlah total dokumen dalam koleksi dan n_j adalah jumlah dokumen yang mengandung setidaknya satu kejadian dari *term* i (Xia & Chai, 2011). Tujuan penghitungan IDF adalah untuk mencari kata-kata yang benar-benar merepresentasikan suatu dokumen teks pada suatu koleksi. Metode pembobotan kata yang digunakan dalam penelitian ini adalah metode TF-IDF. Metode ini digunakan karena metode ini paling baik dalam perolehan informasi. Perhitungan bobot yaitu, frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu yang disebut *Term Frequency* (TF) dan inverse frekuensi dokumen yang mengandung kata yang disebut *Inverse Document Frequency* (IDF). Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata tersebut di dalam dokumen. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tinggi di dalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut akan rendah pada kumpulan dokumen. Kemudian TF-IDF dihitung dengan cara:

$$TF - IDF = TF\ x\ IDF \quad (2.7)$$

Dengan konvensi, nilai TF-IDF meningkat secara proporsional dengan sebuah kata berapa kali muncul dalam sebuah dokumen, namun diimbangi oleh frekuensi kata di dalam korpus, yang membantu mengendalikan fakta bahwa beberapa kata lebih umum daripada yang lain (Christian, Agus, & Suhartono, 2016).

1.6 *Cosine Similarity*

Cosine Similarity atau biasa disebut dengan ukuran kesamaan yang umum digunakan dalam mengukur sudut antara vektor dokumen dengan vektor *query*. Setiap vektor merepresentasikan setiap kata dalam setiap dokumen. Perhitungan yang dihasilkan dari *Cosine Similarity* berupa tingkat kesamaan antara dokumen dengan *query* yang di*input*kan apakah sesuai atau tidak. Semakin tinggi nilai *cosine similarity* maka tingkat kemiripan dokumen dan *query* semakin baik. Sehingga didapatkan rumus yang digunakan untuk *cosine similarity* adalah sebagai berikut (Fahma, Cholissodin, & Perdana, 2018):

$$CosSim\left(d_j, q\right) = \overrightarrow{d_j}.\vec{q} = \sum_{i=1}^{t}(W_{ij}\,.\,W_{iq}) \quad (2.8)$$

1.7 Metode *K-Nearest Neighbor* (K-NN)

Untuk proses pemecahan masalah klasifikasi metode *K-Nearest Neighbor* (K-NN) inilah yang menjadi metode paling sederhana diantara metode klasifikasi lainnya. Teknik yang digunakan pada K-NN ini yakni dengan mengklasifikasikan data dengan obyek yang memiliki nilai tetangga yang paling dekat. hasil yang didapatkan dari proses K-NN ini akan menjadi lebih tinggi atau yang paling optimal ketika dalam perhitungan setiap *term* digunakan pembobotan *cosine similarity* pada masing-masing dokumen yang akan diklasifikasikan (Nurjanah, Pradana, & Fauzi, 2017):

1. *Term Frequency* (TF)

   Merupakan frekuensi kemunculan kata pada suatu dokumen teks. *Term Frequency* (tft,d) didefinisikan jumlah kemunculan *term* t pada dokumen d.

2. *Document Frequency* (DF)

   Merupakan kata-kata yang banyak terdapat pada dokumen, kata tersebut tidak informatif, seperti kata dan, di, atau, bisa, merupakan.

3. *Invers Document Frequency* (IDF)

   Merupakan frekuensi kemunculan *term* pada keseluruhan dokumen teks. *Term* yang jarang muncul pada keseluruhan dokumen teks memiliki nilai *Invers Document Frequency* lebih besar dibandingkan dengan *term* yang sering

4. *Term Frequency - Invers Document Frequency* (TF-IDF)

   Nilai tf-idf dari sebuah kata merupakan kombinasi dari nilai tf dan nilai idf dalam perhitungan bobot.

1.8 Metode *Jaro Winkler Distance*

Merupakan algoritme yang memiliki nilai tertinggi dari algoritme lainnya, meskipun sama-sama digunakan untuk mengukur kesamaan pada dua string. Jika nilai *Jaro Winkler Distance s*emakin tinggi atau mendekati dengan 1 maka string tersebut semakin mirip dengan string acuhan (Kurniawati, Pupitodjati, & Rahman, 2016). Algoritme ini memiliki tiga bagian penyelesaian yaitu : (1) Menghitung panjang string, (2) Menemukan jumlah karakter yang sama didalam dua string, (3) Menemukan jumlah transposisi.

1. Menghitung panjang string
2. Menemukan jumlah karakter yang sama di dalam dua string

$$dj = \frac{1}{3} \times \left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m} \right) \quad (2.9)$$

Ket :
m = jumlah karakter yang sama persis
$|s1|$ = panjang string 1

$|s2|$ = panjang string 2
t = jumlah transportasi

Jarak teoritis dua buah karakter:

$$\left(\frac{\max(|s1|,|s2|)}{s}\right) < -1 \quad (2.10)$$

3. Menemukan jumlah transposisi

$$dw = dj + \left(lp(1 - dj)\right) \quad (2.11)$$

Ket :
dj = Jaro distance untuk string s_1 dan s_2
l = panjang prefiks umum di awal string nilai maksimalnya 4 karakter.
P = konstanta scaling faktor

1.9 *Precision* dan *Recall*

Pengukuran suatu efektifitas dari *information retrieval* dapat dilakukan dengan melakukan perhitungan nilai ketepatan (*precision*), dan nilai perolehan (*recall*). *Precision* dapat diartikan sebagai kecocokan antara permintaan informasi dengan query yang diminta, ketika seseorang mencari informasi di sebuah sistem, dan sistem menampilkan beberapa dokumen, maka kecocokan dokumen tersebut disebut relevan. Sedangkan *Recall* adalah proporsi jumlah dokumen yang dapat ditemukan kembali oleh sebuah proses pencarian di sistem *Informatiom Retrieval* (Lestari, 2015).

Rasio ketepatan (*precision*) adalah perbandingan antara dokumen relevan dengan jumlah dokumen yang ditemu balik dalam penelusuran. Ketepatan (*precision*) berkaitan dengan kemampuan sistem untuk tidak memanggil dokumen yang tidak relevan. Untuk menghitung nilai ketepatan (*precision*) digunakan rumus sebagai berikut (Hasugian, 2006):

$$precision\ (P) = \frac{Jumlah\ dokumen\ relevan\ yang\ terambil}{Jumlah\ dokumen\ terambil\ dalam\ pencarian} \quad (2.12)$$

Rasio perolehan (*recall*) adalah perbandingan dokumen ditemukan dengan jumlah total dokumen relevan dalam sistem. Perolehan (*recall*) berhubungan dengan kemampuan sistem untuk memanggil dokumen yang relevan. Untuk menghitung nilai perolehan (*recall*) digunakan rumus sebagai berikut (Hasugian, 2006):

$$recall\ (R) = \frac{Jumlah\ dokumen\ relevan\ yang\ terambil}{jumlah\ dokumen\ relevan\ pada\ databse} \quad (2.13)$$

Berdasarkan persamaan 2.12 tentang *precision*, dan persamaan 2.13 tentang *recall* maka didapatkan tabel kontingensi sebagai berikut:

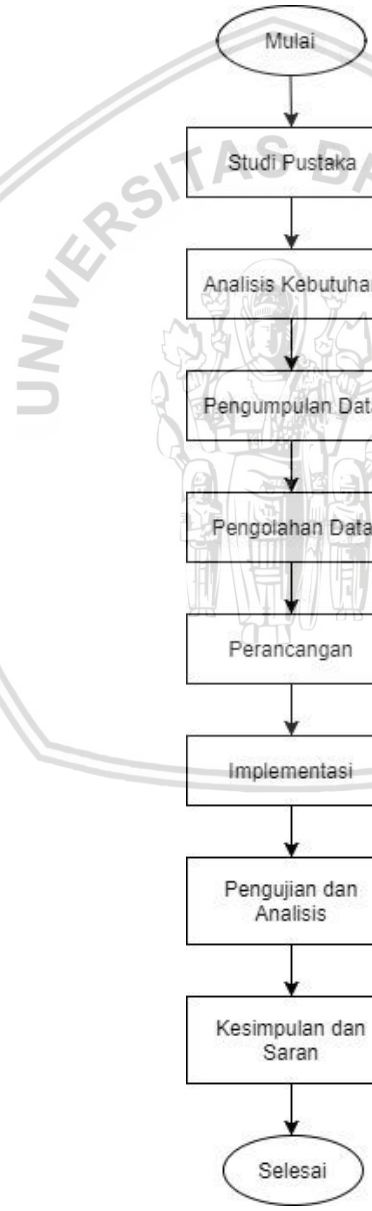
Tabel 2.2 Tabel Kontingensi

		Nilai Sebenarnya	
		Relevance	*Not Relevance*
Nilai Prediksi	*Retrieved*	TP (*True Positive*)	FP (*False Positive*)
	Not Retrieved	FN (*False Negatif*)	TN (*True Negatif*)

Suatu sistem *Information Retrieval* dinyatakan efektif apabila hasil penelusuran mampu menunjukkan ketepatan (*precision*) yang tinggi sekalipun perolehannya rendah. Kondisi ideal dari keefektifan suatu sistem *Information Retrieval* adalah apabila rasio recall dan precision sama besarnya (1:1). Tetapi, karena rasio dari *recall* sulit diukur karena jumlah seluruh dokumen yang relevan dalam database sangat besar, oleh karena itu nilai *precision* yang menjadi salah satu ukuran yang digunakan untuk menilai keefektifan suatu sistem *Information Retrieval* (Hasugian, 2006).

BAB 3 METODOLOGI

Pada bab Metodologi Penelitian ini terdapat langkah – langkah paling dasar yang akan digunakan dalam membangun penelitian, yang pertama adalah studi pustaka, kedua adalah analisis kebutuhan dimana kita harus tau apa saja yang akan sistem butuhkan dan pengguna butuhkan, ketiga adalah pengumpulan data yang digunakan sebagai bahas dalam pengujian, keempat adalah pengolahan data, keenam adalah perancangan sistem, ketujuh adalah implementasi, yang terakhir adalah pengujian, dan pengambilan kesimpulan. Langkah-langkah metode penelitian akan dijelaskan pada Gambar 3.1 .



Gambar 3.1 Diagram Alir Metode Penelitian

1.1. Studi Pustaka

Pengumpulan teori dan referensi dari penelitian sebelumnya yang menjadi dasar dalam membantu melakukan penelitian, atau kini lebih populer dengan sebutan studi pustaka. Dilihat dari fungsinya yakni untuk mempelajari penelitian terdahulu agar penelitian yang akan digunakan bisa memperbaiki kekurangan dari penelitian sebelumnya, maka sudah seharusnya studi pustaka harus sesuai dengan permasalahan yang akan diangkat pada penelitian diantaranya:

- Ulasan atau komentar
- *Text Mining*
- *K-Nearest Neighbor* (K-NN)
- Algoritme *Jaro Winkler Distance*

1.2. Analisis Kebutuhan

Sebuah sistem yang akan dikembangkan harus sesuai dengan apa yang sistem butuhkan maupun yang pengguna butuhkan. Itulah mengapa sebuah analisa kebutuhan sangatlah penting dalam pembuatan sistem agar sesuai target permasalahan, manfaat, dan tujuan dari pembuatan sistem tersebut. Analisa kebutuhan secara umum dibagi menjadi beberapa kebutuhan sistem meliputi :

3.2.1 Kebutuhan Fungsional

- Sistem dapat melakukan analisis sentimen dan perbaikan kata dalam ulasan berbahasa indonesia.
- Sistem dapat menampilkan perbaikan kata yang bermakna ambigu tersebut agar dimengerti konsumen.

3.2.2 Kebutuhan Non-Fungsional

Selain sistem harus memenuhi kebutuhan utama, sistem juga harus memenuhi kebutuhan selain yang utama atau disebut sebagai kebutuhan non-fungsional. Sama-sama merupakan kebutuhan yang harus ada pada sistem seperti memberikan *user interface* yang baik dan tidak asing bagi pengguna sehingga pengguna tidak akan mengalami kesulitan.

3.2.3 Kebutuhan Sistem

- Kebutuhan Data
  - Ulasan atau komentar yang ada situs belanja *online "Lazada"*.
  - Kamus Besar Bahasa Indonesia.
  - Kamus singkatan yang berisi daftar kata yang biasa disingkat ketika menuliskan ulasan yang digunakan sebagai string acuan perbaikan kata.

3.2.4 Kebutuhan Perangkat

- Kebutuhan Hardware
  - Laptop dengan *processor* Intel Core i5-5200U 2.7 GHz
  - *Memory* 4 GB
  - *Hardisk* 500 GB
  - Layar monitor berukuran 14”
- Kebutuhan Software
  - Sistem Operasi *Windows* 8 64-bit
  - *Phyton*
  - *Microsoft Word 2010*
  - *Microsoft Excel 2010*
  - *Notepad*

3.3 Pengumpulan Data

Pada Pengumpulan data yang digunakan untuk sistem “Analisis Sentimen Pada Ulasan *“Lazada”* Berbahasa Indonesia Menggunakan *K-Nearest Neighbor* (K-NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance”,* dilakukan dengan cara mengambil ulasan dari toko *online “Lazada”* yang disimpan dalam bentuk .txt .

3.4 Pengolahan Data

Berdasarkan penjelasan sebelumnya, data yang digunakan pada penelitian ini adalah dengan mengambil ulasan dari toko *online “Lazada”* yang disimpan dalam bentuk .txt, yang nantinya akan diproses melalui hitungan *K-Nearest Neighbor* (K-NN) sebagai klasifikasi teks, kemudian akan dibandingan dengan hasil klasifikasi yang sebelumnya sudah diperbaiki oleh algoritme *Jaro Winkler Distance*.

3.5 Perancangan

Setelah melakukan analisa kebutuhan kita akan mengetahui apa saja yang diperlukan oleh sistem sehingga kita bisa masuk ketahapan selanjutnya yakni melakukan perancangan sistem. Proses ini akan menjelaskan proses klasifikasi dengan metode *K-Nearest Neighbor* (K-NN), dan setelah mendapatkan hasil klasifikasi, maka akan dibandingan dengan hasil klasifikasi yang sebelumnya sudah diperbaiki oleh algoritme *Jaro Winkler Distance*.

3.6 Implementasi

Setelah proses perancangan selesai dilakukan, tahapan selanjutnya dalaha melakukan implementasi sesuai dengan apa yang sudah dirancang sebelumnya. Pada sistem ini Implementasi yang akan diterapkan adalah dengancmenggunakan bahasa pemrograman *phyton*. Selain implementasi yang dilakukan dengan sistem, implementsi juga disertai dengan perhitungan sehingga akan memudahkan pada proses pengujian nanti. Tahapan – tahapan yang ada pada implementasi sistem adalah:

- Implementasi metode *K-Nearest Neigboar*.
- Implementasi algoritme *Jaro Winkler Distance.*

3.7 Pengujian dan Analisis

Tahap pengujian sistem merupakan tahapan dimana kita dapat mengetahui seberapa baikkah sistem yang sudah dibuat, apakah sistem sudah berjalan dengan baik sesuai dengan yang direncanakan apakah sudah sesuai dengan implementasi pada sistem, dan mengantisipasi adanya beberapa *human*. Setelah melakukan pengujian dengan sistem, maka hasil yang didapat akan diolah dengan membandingkan nilai *accuracy* yang didapatkan dari klasifikasi ulasan tanpa menggunakan perbaikan kata, dengan *accuracy* dari klasifikasi ulasan menggunakan perbaikan kata.

3.8 Kesimpulan dan Saran

Pada tahapan ini akan bisa dijalankan dan diambil kesimpulan ketika seluruh proses mulai dari studi pustaka hingga tahap pengujian sudah dilakukan dengan baik sesuai dengan ketentuan yang ada. Secara umum sebuah kesimpulan berisi hasil dari pengujian dan analisis yang mengarah pada kesesuaian antara teori yang menjadi dasar dengan penerapannya kepada sistem. Dan juga pada kesimpulan ini terdapat beberapa saran dan masukan yang diharapkan dapat digunakan untuk memperbaiki jika pada sistem terdapat banyak *bug* atau kesalahan dalam proses pendokumentasian. Diharapkan dari saran tersebut dapat menjadi pengalaman sehingga untuk membangun sistem yang baru bisa lebih baik lagi.

BAB 4 PERANCANGAN

Pada bab ini dijelaskan perancangan sistem secara detil yang digunakan dalam mengembangkan sistem untuk analisis sentimen dan perbaikan kata yang didasarkan dengan teori dan penelitian sebelumnya yang sudah dipelajari. Yang mana hasil perancangan ini akan menjadi dasar dalam pengimplementasian sistem nantinya.

4.1 Deskripsi Permasalahan

Saat ini kita mengetahui bahwa proses jual beli baik dalam kebutuhan pokok maupun kebutuhan lainnya sudah dapat dijual secara *online* salah satunya toko *online* terbesar seperti *"Lazada",* dengan kemudahan akses internet apa yang dibutuhkan seseorang akan dengan mudah terpenuhi hanya dengan duduk dan menggunakan *smartphone*. Guna memberikan pelayanan pada konsumen tentang barang yang dijual maka disediakannya sebuah ulasan atau komentar untuk menampung kritik dan saran dari konsumen. Namun setiap konsumen yang memberikan ulasan memiliki penulisan kata yang beragam, dan tidak sedikit dari mereka yang menuliskan dengan kata yang tidak baku, sehingga menimbulkan kesalahfahaman terhadap isi ulasan tersebut. oleh karena itu diperlukannya analisis sentimen untuk mengetahui ulasan tersebut termasuk ulasan positif atau negatif dengan menggunakan metode klasifikasi yakni *K-Nearest Neighbor*. Kemudian hasil dari klasifikasi ulasan tersebut akan dibandingan dengan hasil klasifikasi ulasan dimana kata yang dianggap ambigu atau tidak baku akan melalui proses perbaikan kata terlebih dahulu menggunakan algoritme *Jaro Winkler Distance*. Banyak metode yang digunakan untuk mengidentifikasi dan memperbaiki kata, salah satunya adalah *Jaro Winkler Distance*. Hasil yang akan ditampilkan berupa nilai perbandingan dari *accuracy* sistem klasifikasi dengan dan tanpa menggunakan perbaikan kata yang dianggap ambigu atau tidak baku.

4.2 Deskripsi Umum Sistem

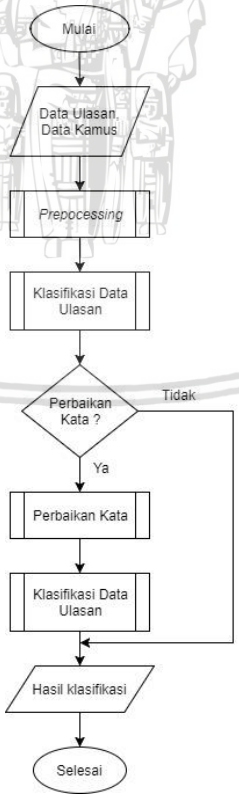
Pada deskripsi sistem ini akan membahas seperti apa sistem yang akan dibangun nantinya dalam menyelesaikan suatu permasalahan analisis sentimen yang ada pada ulasan *"Lazada"* dengan menggunakan metode klasifikasi *K-Nearest Neighbor* dan perbaikan kata menggunakan algoritme *Jaro Winkler Distance*. Proses analisis sentimen akan didefinisikan menjadi 2 kelas yakni kelas positif, dan kelas negatif. Sebelum data *input* diklasifikasikan tentunya harus melewati proses *prepocessing* terlebih dahulu. Setelah proses *prepocessing* telah selesai dilakukan, maka akan masuk ke proses selanjutnya yaitu melakukan pembobotan *term*, dan mengambil 1 *sample* data *training* untuk dihitung berdasarkan nilai *cosine similarity*nya. Kemudian diambil nilai *k*=3 sehingga didapatkan hasil klasifikasi ulasan positif dan negatif. Setelah diketahui hasil klasifikasinya maka hasil klasifikasi tersebut akan dibandingkan dengan hasil klasifikasi ulasan dimana kata yang dianggap ambigu atau tidak baku akan

melalui proses perbaikan kata terlebih dahulu menggunakan algoritme *Jaro Winkler Distance* dengan cara mencocokkan setiap string dari kata tidak baku dengan kata acuan yang diambil dari kamus singkatan yang berisi daftar kata yang biasa disingkat ketika menuliskan ulasan.

Hasil keluaran sistem yang akan dibangun ini berupa hasil analisis sentimen dari masing-masing klasifikasi dengan menggunakan perbaiakan kata, dan tanpa menggunakan perbaikan kata yang dianggap ambigu atau tidak baku.

4.3 Alur Klasifikasi Data *Input* dan Perbaikan Kata

Alur proses klasifikasi data *input* menjelaskan gambaran dari proses pengelompokkan berdasarkan data *training* yang diberikan, hasil yang didapat dihitung berdasarkan nilai *cosine similarity* yang diambil sejumlah *k*. Proses klasifikasi dilakukan dengan menggunakan metode *K-Nearest Neighbor*, dimana data *input* harus melalui proses *prepocessing* dan pembobotan kata terlebih dahulu sebelum dilakukan perhitungan nilai *cosine*nya. Setelah seluruh data *input* terklasifikasi, kemudian dilakukan tahapan *prepocessing* kembali dengan perbaikan kata yang dianggap ambigu atau tidak baku menggunakan algoritme *Jaro Winkler Distance*. Hasil klasifikasi pada kelas yang sudah didefinisikan yakni kelas positif, dan kelas negatif yang nantinya menjadi *output* sistem. Gambar dari alur klasifikasi data *input* ditunjukkan pada Gambar 4.1 .

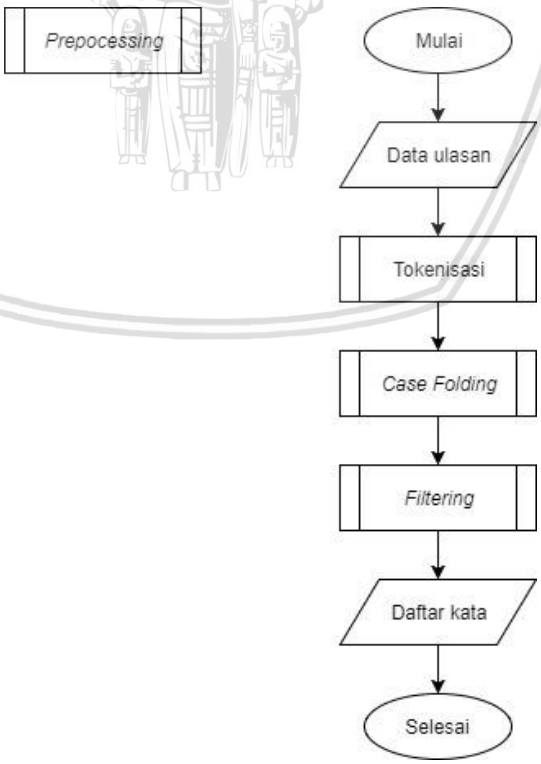


Gambar 4.1 Alur Klasifikasi Data *Input* dan Perbaikan Kata

Berdasarkan pada Gambar 4.1 tentang alur klasifikasi data *input* dan perbaikan kata, terdapat beberapa proses kerja sistem. Proses diawali dengan pemberian data *input* berupa data ulasan dan data kamus yang diambil dari ulasan toko *online* "*Lazada*" kemudian ulasan tersebut disimpan dalam bentuk format lain misalnya .txt agar memudahkan proses input data.

4.3.1 *Prepocessing* Data *Input*

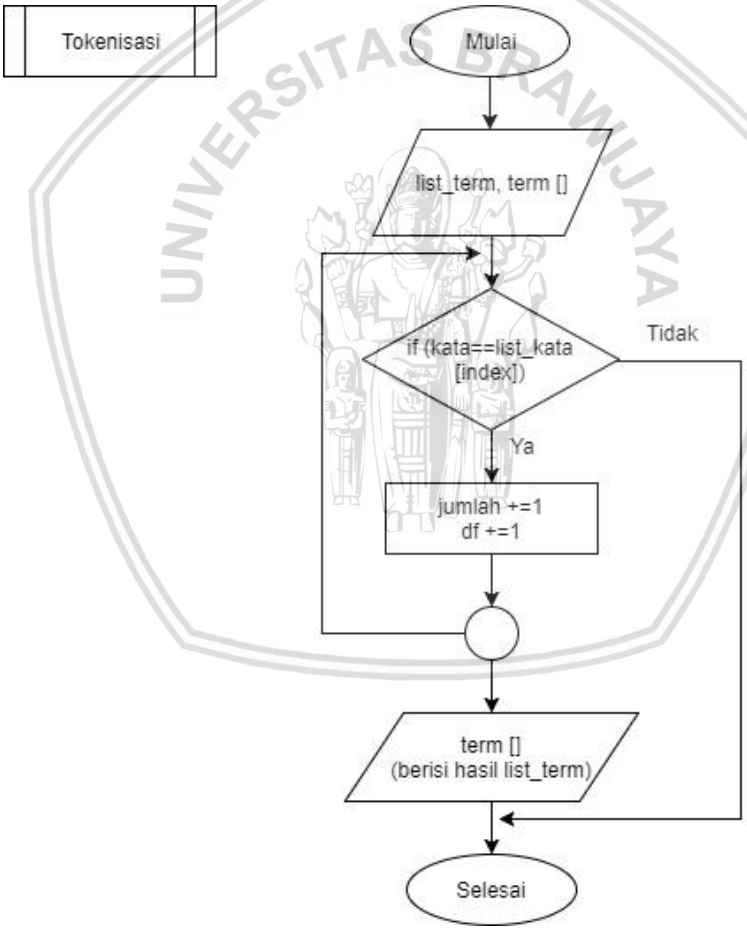
Pada tahapan *prepocessing* digunakan dalam mengekstrak sebuah data teks yang sifatnya tidak terstruktur menjadi sebuah data teks yang lebih terstruktur agar memudahkan sistem dalam melakukan proses setelah proses ini. Dalam *prepocessing* terdapat beberapa tahap yakni tahap pertama *tokenisasi*, tahap kedua *case folding*, dan yang terakhir adalah *filtering*. Namun, *prepocessing* yang digunakan pada sistem ini tidak menggunakan proses *stemming* karena pada proses pengubahan ke kata dasar akan menghilangkan imbuhan, sedangkan untuk proses identifikasi kata dibutuhkan kata yang sesuai tanpa adanya perubahan dari kata asli. Hasil keluaran dari tahapan ini adalah daftar kata atau *term list* yang mewakili setiap data *input* yang diberikan. Gambar dari alur tahapan *prepocessing* data *input* ditunjukkan Gambar 4.2 .



Gambar 4.2 *Prepocessing* Data *Input*

Berdasarkan pada Gambar 4.2 tentang alur tahapan *prepocessing* data *input* maka data ulasan pada *"Lazada"* harus melalui proses tokenisasi, *case folding*, dan *filtering* terlebih terdahulu sebelum diproses ke tahapan selanjutnya. Data ulasan terdiri dari sekumpulan kata yang menjadi satu kalimat atau lebih yang didalamnya terdapat penggunaan kata yang tidak baku, sehingga dibutuhkan proses *prepocessing* untuk mengambil kata atau *term* yang mewakili setiap data *input* yang diberikan.

Tahapan pertama pada *prepocessing* adalah tahap *tokenisasi* dimana dalam proses ini terdapat pemisahan antara kata dari kalimat sehingga memudahkan pada tahapan selanjutnya. Alur proses tokenisasi ditunjukkan Gambar 4.3 .

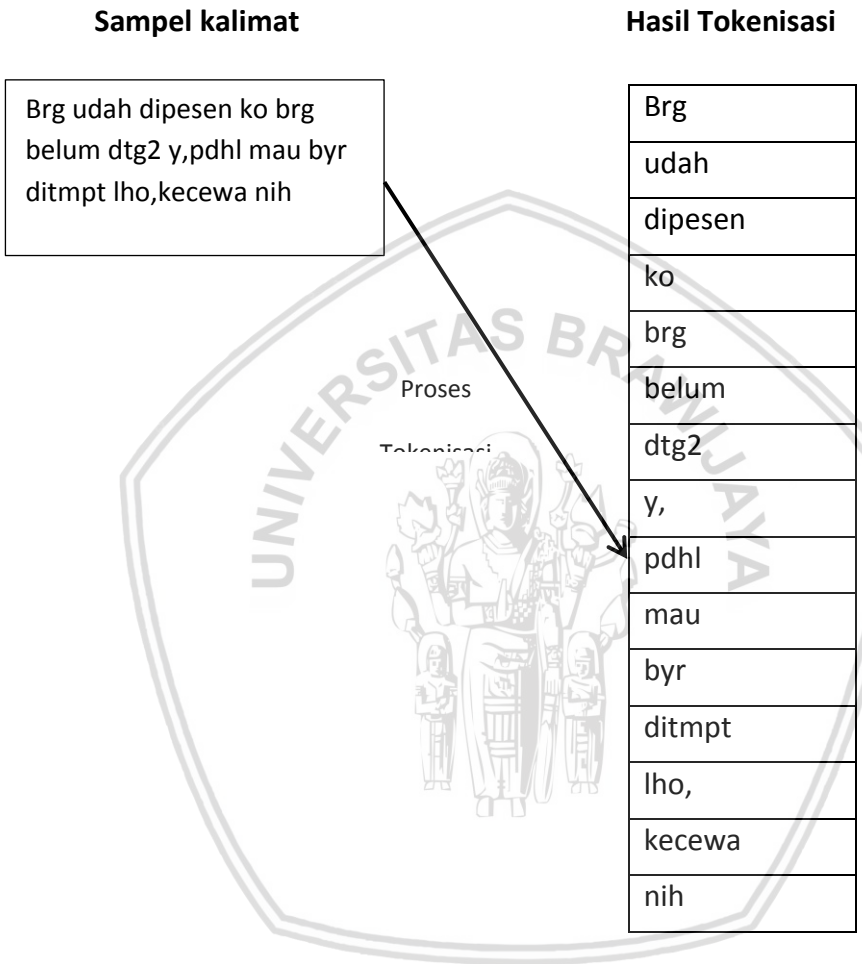


Gambar 4.3 Alur Tahapan Tokenisasi

Berdasarkan Gambar 4.3 tentang alur proses *tokenisasi* untuk memisahkan antar kata dari kalimat untuk menjadi *input*. Langkah awal adalah mendeklarasikan variabel *list_term* dan *term[]* untuk menyimpan hasil

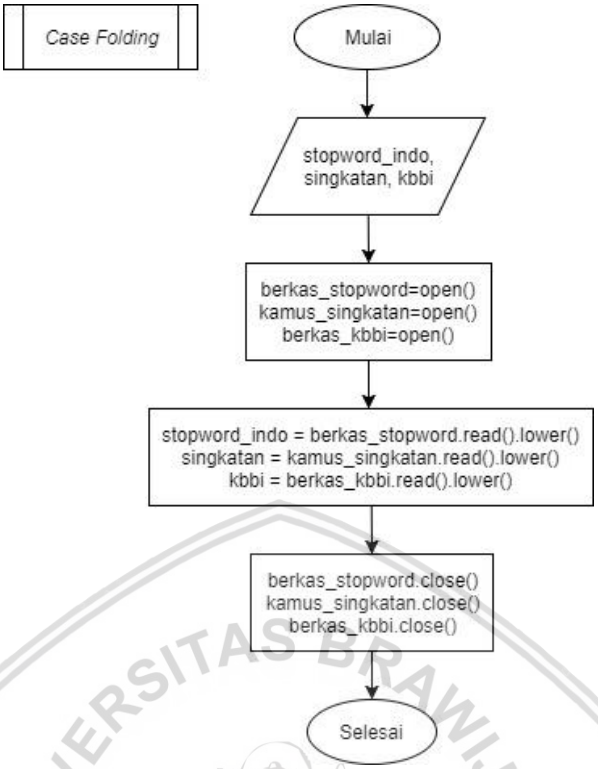
pemisahan kata. Kemudian kondisi perulangan yang digunakan jika kalimat ulasan masih terbaca, maka proses penambahan kata pada nilai jumlah dan nilai df akan terus dijalankan. Setelah perulangan berhenti, hasil dari pemisahan tersebut yaitu kumpulan *list_term* akan disimpan pada variabel *term[]* yang juga sudah dideklarasikan.

Tahapan tokenisasi dengan sampel kalimat ulasan pada *"Lazada"* ditunjukkan pada Gambar 4.4 .



Gambar 4.4 Hasil Tokenisasi

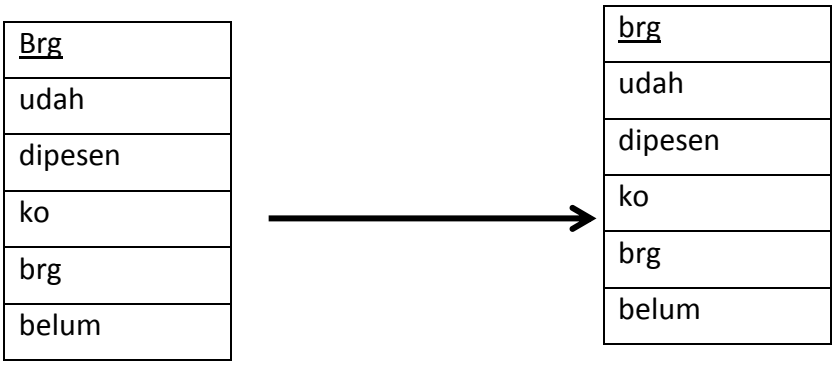
Hasil tokenisasi pada Gambar 4.4 akan masuk pada tahap kedua yaitu *case folding*. Pada tahap kedua ini digunakan untuk mengolah hasil pemisahan kata dari proses sebelumnya yakni tokenisasi dengan langkah kerja yang dilakukan *case folding* yaitu mengubah seluruh *term* yang berawalan kapital menjadi huruf kecil (*lowercase*). Alur proses *case folding* ditunjukkan Gambar 4.5 .



Gambar 4.5 Alur Tahapan *Case Folding*

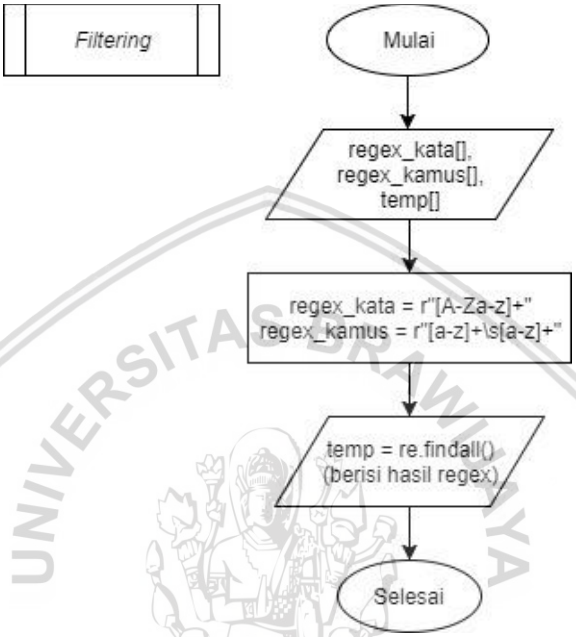
Berdasarkan pada Gambar 4.5 tentang alur tahapan case *folding* untuk mengubah semua huruf menjadi huruf kecil. Langkah awal adalah mendeklarasikan variabel *stopword_ indo,* singkatan, dan kbbi untuk menyimpan hasil *case folding* dari masing-masing berkas. Setiap berkas yang akan dilakukan *case folding* harus dibuka terlebih dahulu dengan perintah *open*, kemudian masing-masing berkas dibaca sekaligus diubah menjadi huruf kecil dengan fungsi *lower()* yang hasilnya akan otomatis tersimpan pada variabel yang telah dideklarasikan sebelumnya. Hasil dari alur *Case Folding* diatas ditunjukkan pada Gambar 4.6 .

Hasil Tokenisasi **Hasil *Case Folding***



Gambar 4.6 Hasil *Case Folding*

Hasil *case folding* pada Gambar 4.6 akan masuk pada tahapan selanjutnya yaitu *filtering*. Tahapan *filtering* digunakan untuk melakukan penghapusan beberapa karakter seperti seluruh angka, seluruh tanda baca, baik simbol, dan juga kata hubung, serta membuang seluruh karakter yang dianggap tidak efisien dalam memproses sebuah data teks. Alur tahapan *filtering* ditunjukkan Gambar 4.7 .



Gambar 4.7 Alur Tahapan *Filtering*

Berdasarkan pada Gambar 4.7 tentang alur tahapan *filtering* untuk menghapus seluruh angka, seluruh tanda baca, kata hubung, termasuk simbol, dan segala bentuk karakter yang dianggap tidak efisien dalam pemrosesan data teks. Langkah awal adalah mendeklarasikan variabel *regex_kata[]*, *regex_kamus[]*, dan *temp[]* untuk menyimpan hasil *case folding.* Fungsi yang digunakan dalam proses penghapusan adalah fungsi r"[A-Za-z]+", dan r"[a-z]+\s[a-z]+" yang berarti selain huruf, spasi, dan enter maka akan dihapus. Hasil dari alur *filtering* diatas ditunjukkan pada Gambar 4.8 .

Hasil *Case Folding*

brg
udah
dipesen

Hasil *Filtering*

brg
udah
dipesen

ko
brg
belum

brg
belum

Gambar 4.8 Hasil *Filtering*

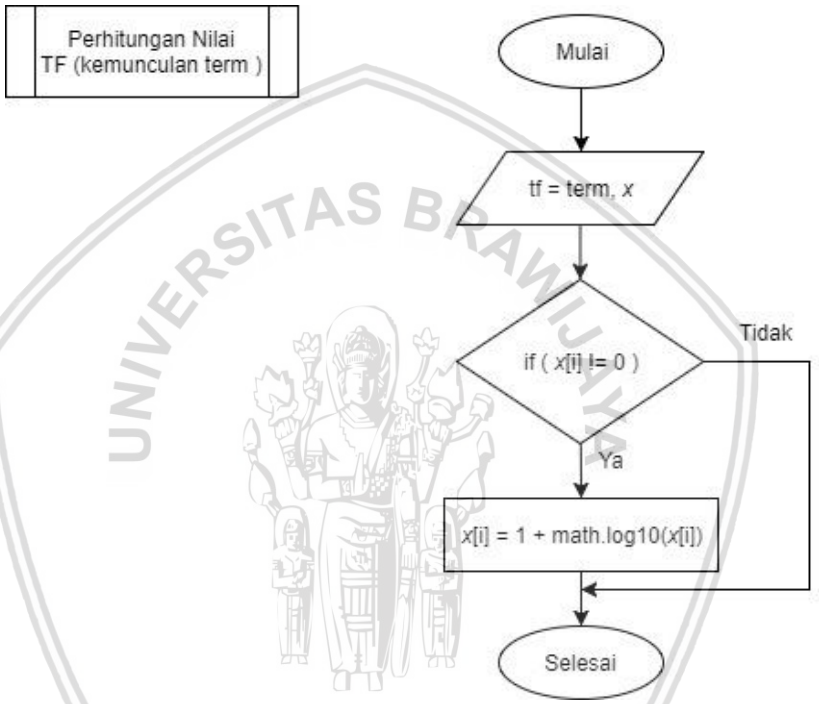
4.3.2 Klasifikasi Data Ulasan

Pada tahapan klasifikasi data ulasan digunakan untuk mengelompokkan data ulasan yang di*input*kan menjadi dua kelas yaitu kelas ulasan dengan kata baku, dan kelas ulasan dengan kata tidak baku. Proses ini terdapat beberapa tahapan didalamnya yang pertama adalah menentukan nilai *k* tetangga terdekat, kemudian melakukan proses *prepocessing*, perhitungan nilai TF-IDF, menentukan kemiripan vektor dokumen menggunakan *cosine similarity*, dan mengurutkan nilai dari yang terbesar ke terkecil berdasarkan nilai *cosine*nya. Hasil keluaran dari tahapan ini adalah klasifikasi ulasan berdasarkan kelas yang didefinisikan. Gambaran alur tahapan klasifikasi data ulasan ditunjukkan pada Gambar 4.9 .

Gambar 4.9 Klasifikasi Data Ulasan

Berdasarkan pada Gambar 4.9 tentang gambaran alur klasifikasi data ulasan, terdapat beberapa proses kerja sistem. Proses diawali dengan penentuan nilai *k* tetangga. Setelah penentuan nilai *k* maka tahapan selanjutnya adalah proses *prepocessing* yang sudah dijelaskan pada Gambar 4.2, dari proses tersebut dihasilkan sebuah *term* atau kata yang akan dijadikan *input* pada proses selanjutnya yaitu untuk menghitung TF-IDF nya.

Pertama yang dilakukan ialah menghitung kemunculan *term* yang disebut TF (*Term Frequency*) yang akan dijelaskan pada Gambar 4.10 .



Gambar 4.10 Alur Perhitungan TF (*Term Frequency*)

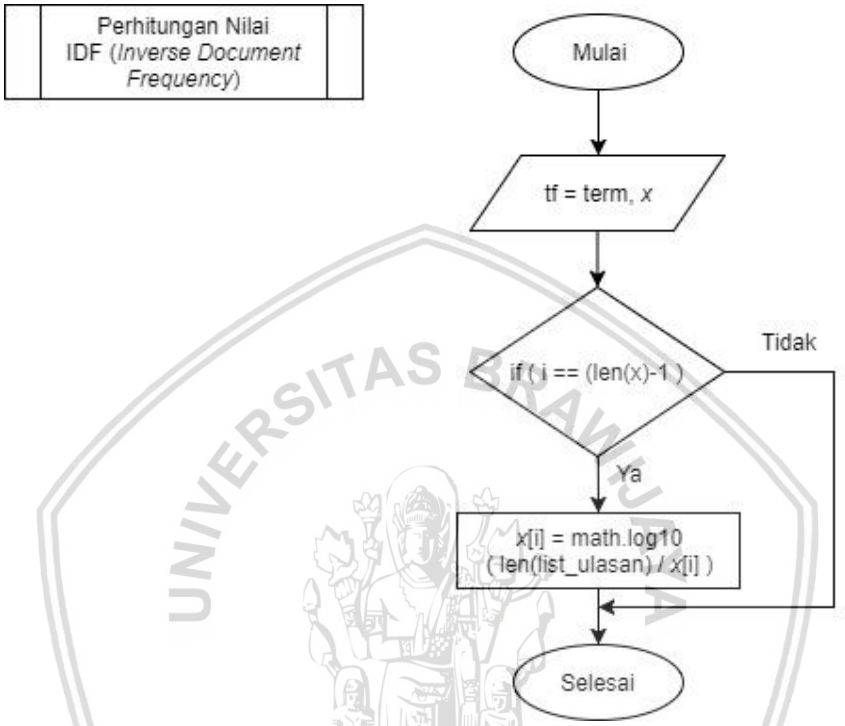
Pada gambar 4.10 menjelaskan tentang proses perhitungan nilai TF (*Term Frequency*) berdasarkan *term* yang didapatkan dari proses *prepocessing*. Perhitungan manual nilai TF dijelaskan pada tabel 4.1 .

Tabel 4.1 Perhitungan nilai TF (*Term Frequency*)

TF	U1	U2	U3	U4	U5	DF
murah	1	1	0	0	0	2
kualitas	1	0	0	0	0	1
bagus	1	0	0	0	0	1
terima	0	1	0	0	1	2
baik	0	1	0	0	1	1

cantik	0	1	0	0	0	1

Setelah proses perhitungan TF (*Term Frequency*) selesai dilakukan, maka tahapan selanjutnya adalah perhitungan jumlah ulasan yang mengandung *term* tertentu yang dinamakan dengan proses IDF (*Inverse Document Frequency*). Penjelasan mengenai alur tahapan perhitungan IDF yang akan dijelaskan pada Gambar 4.11 .



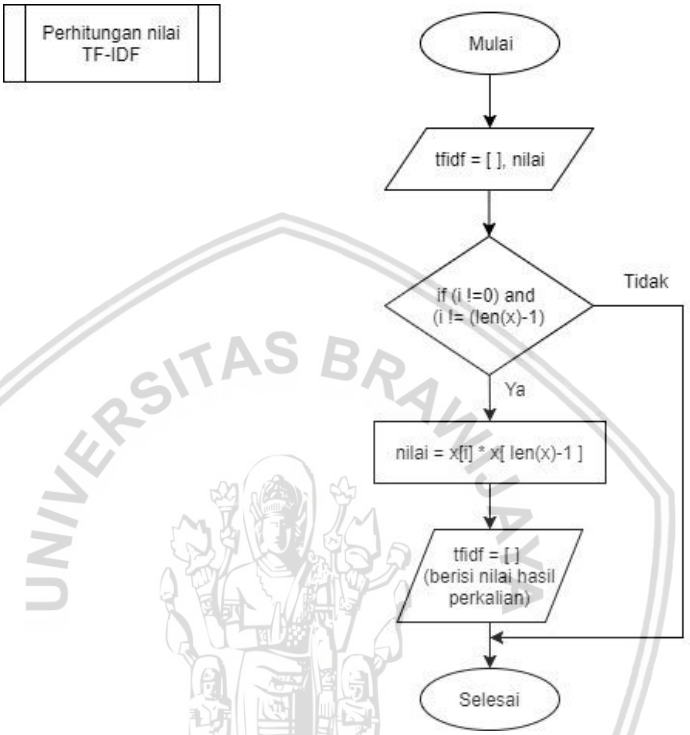
Gambar 4.11 Alur Perhitungan IDF (*Inverse Document Frequency*)

Pada gambar 4.11 setelah dilakukan proses perhitungan nilai IDF (*Inverse Document Frequency*) pada ulasan dan sudah didapatkan hasilnya. Perhitungan manual nilai IDF dijelaskan pada tabel 4.2 .

Tabel 4.2 Perhitungan Nilai IDF (*Inverse Document Frequency*)

TF	U1	U2	U3	U4	U5	IDF
murah	1	1	0	0	0	0,39794
kualitas	1	0	0	0	0	0,69897
bagus	1	0	0	0	0	0,69897
terima	0	1	0	0	1	0,39794
baik	0	1	0	0	0	0,69897
cantik	0	1	0	0	0	0,69897

Setelah kedua proses perhitungan yakni TF (*Term Frequency*) dan IDF (*Inverse Document Frequency*) selesai dilakukan, maka proses akan dilanjutkan untuk menghitung nilai perkalian antara TF (*Term Frequency*) dan IDF (*Inverse Document Frequency*). Penjelasan mengenai alur tahapan perhitungan perkalian antara TF dan IDF akan dijelaskan pada Gambar 4.12 .



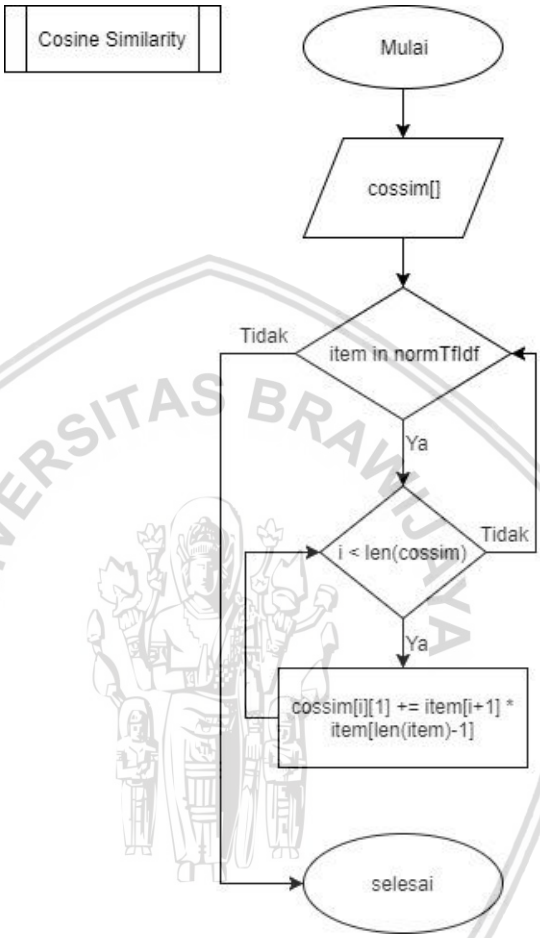
Gambar 4.12 Alur Perhitungan Nilai Perkalian TF dan IDF

Pada gambar 4.12 setelah dilakukan proses perhitungan nilai perkalian antara TF (*Term Frequency*) dan IDF (*Inverse Document Frequency*) pada *term*. Perhitungan manual nilai IDF (*Inverse Document Frequency*) yang dijelaskan tabel 4.3 .

Tabel 4.3 Perhitungan Nilai TF-IDF

TF-IDF	U1	U2	U3	U4	U5
murah	0,39794	0,39794	0	0	0
kualitas	0,69897	0	0	0	0
bagus	0,69897	0	0	0	0
terima	0	0,39794	0	0	0,39794
baik	0	0,69897	0	0	0
cantik	0	0,69897	0	0	0

Setelah proses perhitungan TF-IDF selesai dilakukan, maka proses akan dilanjutkan untuk menghitung nilai *cosine similarity* yang digunakan untuk mencari kesamaan antar ulasan. Penjelasan mengenai alur tahapan perhitungan *cosine similarity* yang akan dijelaskan pada Gambar 4.13 .



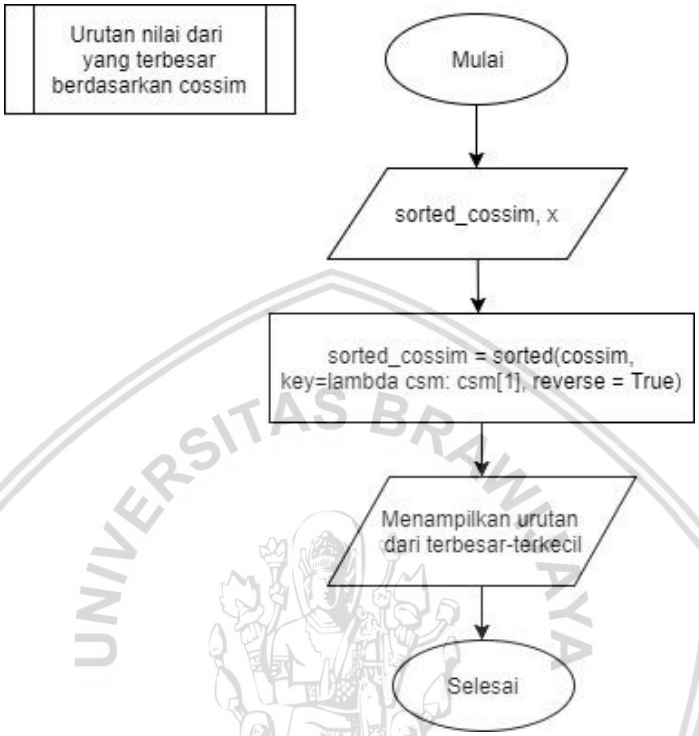
Gambar 4.13 Alur Perhitungan Cosine Similarity

Pada gambar 4.13 setelah dilakukan proses perhitungan nilai *cosine similarity* dengan deklarasi cossim[] sebagai tempat menyimpan nilai cossim yang telah melewati proses perhitungan, dan *x* sebagai variabel. Perhitungan manual nilai hasil perhitungan *cosine similarity* dijelaskan pada tabel 4.4 .

Tabel 4.4 Perhitungan Nilai *Cosine Similarity*

CoSim	U1	U2	U3	U4
U5	0	0,093949	0	0,029389

Maka proses selanjutnya adalah mengurutkan nilai tersebut berdasarkan hasil kemiripan yang terbesar. Penjelasan mengenai alur tahapan pengurutan nilai *cosine similarity* yang akan dijelaskan pada Gambar 4.14 .



Gambar 4.14 Alur Pengurutan Nilai *Cosine Similarity*

Pada gambar 4.14 dilakukan pengurutan dengan cara bertukar nilai tepat di samping posisi angka yang ditukar, langkah tersebut dilakukan secara terus-menerus untuk membandingkan tiap-tiap angka dalam elemen array dan menukar posisi apabila urutannya salah, hingga dapat dipastikan bahwa dalam satu kali iterasi tidak ada lagi perubahan posisi. . Perhitungan manual nilai *cosine similarity* dijelaskan pada tabel 4.5 .

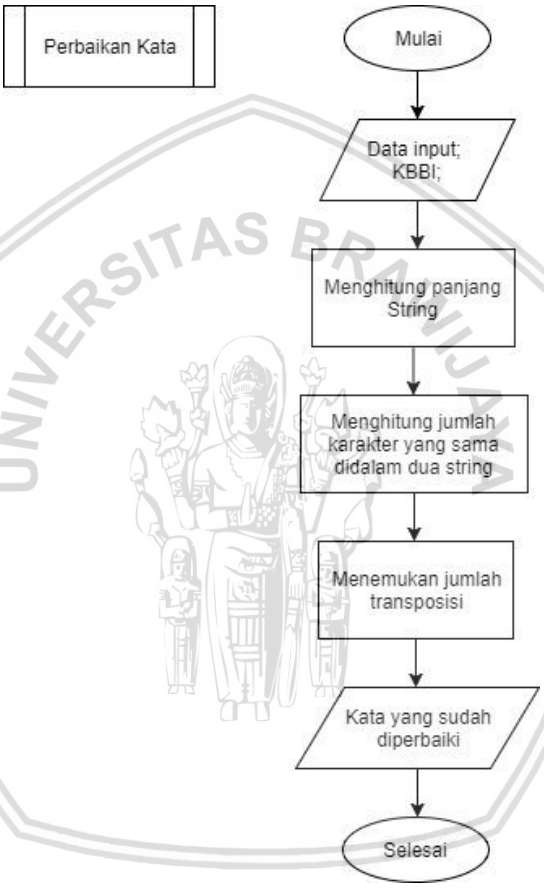
Tabel 4.5 Pengurutan Nilai *Cosine Similarity*

CoSim	U2	U4	U1	U3
	0,092383	0,029389	0	0

4.3.3 Perbaikan Kata

Pada tahapan perbaikan kata ini digunakan untuk memperbaiki kata yang tidak baku setelah dilakukan pencocokan dengan kamus singkatan yang berisi daftar kata yang sering disingkat pada penulisannya, seperti contoh “bagus ditulis bgs, bgus”, “murah ditulis mrah murmer”, dan masih banyak lagi.

Perbaikan kata yang dimaksud adalah mengukur kesamaan pada dua string yaitu antara string target dan string acuhan, jika nilai *distance s*emakin tinggi atau mendekati dengan 1 maka string tersebut semakin mirip dengan string acuhan yang ada pada kamus. Proses ini terdapat beberapa tahapan didalamnya yakni yang pertama menghitung panjang string, menghitung jumlah karakter yang sama didalam dua string, dan menemukan jumlah transposisi. Hasil keluaran dari tahapan ini adalah perubahan dari kata tidak baku menjadi kata baku yang sesuai dengan Kamus Besar Bahasa Indonesia. Gambaran dari alur tahapan perbaikan kata ditunjukkan pada Gambar 4.15 .



Gambar 4.15 Perbaikan Kata

Berdasarkan pada Gambar 4.15 tentang gambaran alur perbaikan terhadap kata yang tidak baku, terdapat beberapa proses kerja sistem. Proses diawali dengan perhitungan panjang string pada masing-masing kata *input* dengan kata acuan yang ada pada KBBI. Kemudian proses dilanjutkan dengan menghitung jumlah karakter yang sama secara berurutan antara 2 kata tersebut, setelah itu dihitung pula jumlah transposisi atau ada tidaknya string yang tertukar.

Perhitungan manual dilakukan berdasarkan kata yang terdeteksi tidak baku yaitu “brg”, “udah”, “pesen”, “dtg”, “y”, “pdhl”, “byr”, “tmpt, “byar”, “lma”, “bnget”, “ya”, dan beberapa kandidat kata benar yaitu “barang”, “sudah”, “pesan”, “datang”,

"iya", "padahal", "bayar", "tempat", "lama", "banget". Berikut merupakan contoh perhitungan manual perbaikan kata menggunakan metode *Jaro Winkler Distance*:

1. Kata Pertama

   String 1 = barang $s_1 = 6$ m = 3

   String 2 = brg $s_2 = 3$ t = 0

   $$dj = \frac{1}{3} \times \left(\frac{3}{|6|} + \frac{3}{|3|} + \frac{3-0}{3}\right) = 0{,}833$$

   $$dw = 0{,}833 + \left(1 * 0{,}1(1 - 0{,}833)\right) = 0{,}85$$

   String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari brg adalah barang.

2. Kata Kedua

   String 1 = sudah $s_1 = 5$ m = 4

   String 2 = udah $s_2 = 4$ t = 0

   $$dj = \frac{1}{3} \times \left(\frac{4}{|6|} + \frac{4}{|4|} + \frac{4-0}{4}\right) = 0{,}933$$

   $$dw = 0{,}933 + \left(1 * 0{,}1(1 - 0{,}933)\right) = 0{,}866$$

   String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari udah adalah sudah.

3. Kata Ketiga

   String 1 = pesan $s_1 = 5$ m = 4

   String 2 = pesen $s_2 = 5$ t = 0

   $$dj = \frac{1}{3} \times \left(\frac{4}{|5|} + \frac{4}{|5|} + \frac{4-0}{4}\right) = 0{,}867$$

   $$dw = 0{,}867 + \left(1 * 0{,}1(1 - 0{,}867)\right) = 0{,}906$$

   String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari pesen adalah pesan.

4. Kata Keempat

   String 1 = datang $s_1 = 6$ m = 3

String 2 = dtg $s_2 = 3$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{3}{|6|} + \frac{3}{|3|} + \frac{3-0}{3}\right) = 0{,}833$$
$$dw = 0{,}833 + \left(1 * 0{,}1(1 - 0{,}833)\right) = 0{,}85$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari dtg adalah datang.

5. Kata Kelima

String 1 = iya $s_1 = 3$ m = 1

String 2 = y $s_2 = 1$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{1}{|3|} + \frac{1}{|1|} + \frac{1-0}{1}\right) = 0{,}777$$

$$dw = 0{,}777 + \left(0 * 0{,}1(1 - 0{,}777)\right) = 0{,}777$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari y adalah iya.

6. Kata Keenam

String 1 = padahal $s_1 = 7$ m = 4

String 2 = pdhl $s_2 = 4$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{4}{|7|} + \frac{4}{|4|} + \frac{4-0}{4}\right) = 0{,}857$$

$$dw = 0{,}857 + \left(1 * 0{,}1(1 - 0{,}857)\right) = 0{,}871$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari pdhl adalah padahal.

7. Kata Ketujuh

String 1 = bayar $s_1 = 5$ m = 3

String 2 = byr $s_2 = 3$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{3}{|5|} + \frac{3}{|3|} + \frac{3-0}{3}\right) = 0{,}867$$

$$dw = 0{,}867 + \left(1 * 0{,}1(1 - 0{,}867)\right) = 0{,}88$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari byr adalah bayar.

8. Kata Kedelapan

String 1 = tempat $s_1 = 6$ m = 4

String 2 = tmpt $s_2 = 4$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{4}{|6|} + \frac{4}{|4|} + \frac{4-0}{4}\right) = 0{,}889$$

$$dw = 0{,}889 + \left(1 * 0{,}1(1 - 0{,}889)\right) = 0{,}9$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari tmpt adalah tempat.

9. Kata Kesembilan

String 1 = bayar $s_1 = 5$ m = 4

String 2 = byar $s_2 = 4$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{4}{|5|} + \frac{4}{|4|} + \frac{4-0}{4}\right) = 0{,}933$$

$$dw = 0{,}933 + \left(1 * 0{,}1(1 - 0{,}933)\right) = 0{,}94$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari byar adalah bayar.

10. Kata Kesepuluh

String 1 = lama $s_1 = 5$ m = 3

String 2 = lma $s_2 = 3$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{3}{|5|} + \frac{3}{|3|} + \frac{3-0}{3}\right) = 0{,}917$$

$$dw = 0{,}917 + \left(1 * 0{,}1(1 - 0{,}917)\right) = 0{,}925$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari lma adalah lama.

11. Kata Kesebelas

String 1 = banget $s_1 = 6$ m = 5

String 2 = bnget $s_2 = 5$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{5}{|6|} + \frac{5}{|5|} + \frac{5-0}{5}\right) = 0{,}944$$

$$dw = 0{,}944 + \left(1 * 0{,}1(1 - 0{,}944)\right) = 0{,}95$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari bnget adalah banget.

12. Kata Keduabelas

String 1 = iya $s_1 = 3$ m = 2

String 2 = ya $s_2 = 2$ t = 0

$$dj = \frac{1}{3} \times \left(\frac{2}{|3|} + \frac{2}{|2|} + \frac{2-0}{2}\right) = 0{,}889$$

$$dw = 0{,}889 + \left(0 * 0{,}1(1 - 0{,}889)\right) = 0{,}889$$

String uji mendapat nilai Jaro mendekati 1, maka kedua string dianggap memiliki kesamaan sehingga dapat dipastikan jika kata baku dari ya adalah iya.

4.4 Perancangan Pengujian Sistem

Layaknya sistem pada umumnya, setelah melakukan proses perancangan sistem, pembuatan sistem, maka tahapan selajutnya adalah melakukan pengujian pada sistem. Tahap pengujian bertujuan untuk mengetahui apakah sistem yang dibuat memiliki fungsi yang sesuai atau belum, selain itu pengujian juga digunakan untuk menguji besarnya nilai kesesuaian yang dihasilkan antara perhitungan manual dengan yang sudah diterapkan pada sistem. Pengujian yang

akan dilakukan terdiri atas pengujian pengaruh nilai *k*, pengujian pengaruh banyaknya data latih, dan pengujian pengaruh penggunaan perbaikan kata. Dari ketiga pengujian tersebut masing-masing akan menguji analisis sentimen tanpa perbaikan kata, dan analisis sentimen dengan menggunakan perbaikan kata. Keduanya akan diuji secara terpisah agar mendapatkan hasil perbandingan *accuracy* antara kedua analisis sentimen tersebut.

4.4.1 Perancangan Pengujian Pengaruh Nilai *k*

Pada pengujian ini dilakukan untuk mengetahui pada nilai *k* yang mana yang menjadi nilai ketetanggan paling baik atau optimal yang digunakan dalam proses klasifikasi K-NN yang nantinya akan mempengaruhi hasil *accuracy* dari system, dan nilai *k* yang digunakan pada pengujian ini juga bervariatif tidak memperdulikan ganjil dan genap. Adapun rancangan detail untuk perancangan pengujian pengaruh nilai *k* ditunjukkan pada Tabel 4.6 .

Tabel 4.6 Perancangan Pengujian Pengaruh Nilai *k*

No.	Nilai *k*-	*Precision*	*Recall*	*Accuracy (%)*
1	3			
2	5			
3	...			
4	...			
5	...			

4.4.2 Perancangan Pengujian Pengaruh Banyaknya Data Latih

Pada pengujian ini dilakukan untuk mencari nilai data latih ke berapa yang memberikan pengaruh yang paling optimal terhadap sistem yang nantinya akan memberikan nilai *accuracy* yang baik. Adapun rancangan detail untuk perancangan pengujian pengaruh banyaknya data latih ditunjukkan pada Tabel 4.7 .

Tabel 4.7 Perancangan Pengujian Pengaruh Banyaknya Data Latih

No.	Jumlah Data Latih	*Precision*	*Recall*	*Accuracy*
1	250			
2	300			
3	350			
4	...			
5	...			

4.4.3 Perancangan Pengujian Pengaruh Penggunaan Perbaikan Kata

Pada pengujian ini dilakukan berdasarkan perbaikan kata dalam setiap *term* yang dianggap tidak baku untuk mengetahui seberapa besar pengaruh perbaikan kata tidak baku terhadap nilai *precision*, *recall*, dan *accuracy* pada

sistem. Adapun rancangan detail untuk perancangan pengujian pengaruh penggunaan perbaikan kata ditunjukkan pada Tabel 4.8 .

Tabel 4.8 Perancangan Pengujian Pengaruh Penggunaan Perbaikan Kata

Uji	*Precision*	*Recall*	*Accuracy*
Tanpa Perbaikan Kata			
Dengan Perbaikan Kata			

4.4.4 Perancangan Hasil Dari Seluruh Pengujian

Pada pengujian ini dilakukan untuk mengetahui hasil dari setiap proses pengujian yang telah dilakukan, hasil yang akan ditampilkan berupa grafik yang terdiri dari nilai *precision*, *recall*, dan *accuracy* dari seluruh proses pengujian. Pada setiap tahap pengujian dilakukan dengan 2 cara yakni yang pertama dengan menggunakan proses *prepocessing* tanpa perbaikan kata, dan yang kedua dengan menggunakan proses *prepocessing* dengan menggunakan perbaikan kata (Antinasari, Perdana, & Fauzi, 2017). Adapun rancangan detail untuk halaman proses data *input* ditunjukkan pada Tabel 4.9 .

Tabel 4.9 Perancangan Hasil Dari Seluruh Pengujian

	Tanpa Perbaikan Kata			Dengan Perbaikan Kata		
	Precision	*Recall*	*Accuracy*	*Precision*	*Recall*	*Accuracy*
Pengujian 1						
Pengujian 2						
Pengujian 3						

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi sistem berdasarkan metodologi dan perancangan yang telah dijelaskan pada bab sebelumnya. Bab ini terdapat pembahasan mengenai batasan implementasi dan implementasi sistem.

5.1 Deskripsi Lingkungan Implementasi

Pada deskripsi lingkungan implementasi ini bertujuan untuk merancang sistem dalam ruang lingkup yang jelas sesuai dengan tujuan sistem, serta perancangan yang sudah dibuat pada bab sebelumnya. Berikut merupakan deksripsi lingkungan implementasi yang ada pada penelitian yang akan dibangun ini:

1. Penggunaan perangkat keras
   - Laptop dengan processor Intel® Core i5-5200U CPU @2.7GHz
   - RAM 4.00 GB
   - *Memory* 500 GB
2. Penggunaan perangkat lunak
   - Sistem Operasi *Windows* 8 64-bit
   - *Phyton*
   - *Microsoft Word 2010*
   - *Microsoft Excel 2010*
   - *Notepad*
3. Metode yang ada dan yang akan digunakan dalam menyelesaikan masalah pada penelitian yang akan dibangun adalah metode *K-Nearest Neighbor* (K-NN), dan *Jaro Winkler Distance*.
4. Data yang digunakan adalah data ulasan dari situs belanja *online "Lazada"* dengan total data 500 ulasan yang akan dibagi menjadi 50 ulasan sebagai data uji, dan 450 ulasan sebagai data latih. Dengan produk yang dipilih yakni produk tas dengan rentang harga Rp 25.000 – Rp. 150.000.
5. Kamus yang digunakan sebagai acuan dalam perbaikan kata menggunakan kamus yang dibuat sendiri berdasarkan kata yang biasanya disingkat.
6. Proses perbaikan kata dengan metode *Jaro Winkler Distance* dilakukan melalui proses pencocokan terhadap kata yang ada pada kamus singkatan yang dibuat berisi daftar kata yang biasanya dalam penggunaan nya disingkat terutama pada penulisan suatu ulasan.
7. Perbaikan kata hanya dilakukan terhadap huruf, tidak untuk angka maupun tanda baca.
8. Keluaran (*output*) yang dihasilkan sistem berupa analisis sentimen dari setiap data uji yang dimasukkan ke sistem.

5.2 Implementasi Sistem

Implementasi sistem menjelaskan tentang tahapan-tahapan yang harus dilakukan pada aplikasi analisis sentimen pada ulasan *"Lazada"* berbahasa Indonesia menggunakan metode *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance*. Pada proses implementasi sistem tidak sembarangan namun sudah didasari pada tahapan dan alur proses yang sudah dijelaskan pada bab sebelumnya. Pada penjelasan tahap implementasi ini akan dicantumkan sub bab yang masing-masing akan mewakili tahapan prosesnya.

5.2.1 Implementasi Pemberian Data *Input*

Yang paling utama tentunya tahap implementasi data *input* pada system, dimana data *input* tersebut berupa ulasan atau data teks yang sebelumnya sudah tersimpan dalam format .txt . Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.1 .

Baris Ke-	Kode Program
1	`berkas_stopword = open('stopword indonesia.txt')`
2	`kamus_singkatan = open('kamus singkatan.txt')`
3	`berkas_kbbi = open('kbbi.dic')`
4	`stopword_indo = berkas_stopword.read().lower()`
5	`singkatan = kamus_singkatan.read().lower()`
6	`kbbi = berkas_kbbi.read().lower()`
7	`berkas_stopword.close()`
8	`kamus_singkatan.close()`
9	`berkas_kbbi.close()`

Gambar 5.1 Implementasi Pemberian Data *Input*

Keterangan Gambar 5.1:

Baris 1-3 :Digunakan untuk membuka file yang akan diproses.

Baris 4-6 :Deklarasi variabel sekaligus untuk membaca file agar bisa diproses.

Baris 7-9 :Digunakan untuk menutup file yang telah dibaca.

5.2.2 Implementasi Proses *Prepocessing* Data *Input*

Tahapan implementasi ini merupakan tahapan yang digunakan dalam proses perhitungan selanjutnya. Dimana setiap hasil dari proses *prepocessing* akan disimpan pada bentuk array yang sebelumnya sudah dijelaskan pada Gambar 4.2 . Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.2 .

Baris Ke-	Kode Program
1	`regex_kata = r"[A-Za-z]+"`
2	`regex_kamus = r"[a-z]+\s[a-z]+"`
3	
4	`stopword = re.findall(regex_kata, stopword_indo)`
5	`list_stopword = [kata for kata in`
6	`re.findall(regex_kamus, singkatan)]`

```
 7  list_kbbi = [kata for kata in re.findall(regex_kata,
 8  kbbi)]
 9  singkatan = [item for item in re.findall(regex_kamus,
10  singkatan)]
11  list_singkatan = []
12  for item in singkatan:
13       temp = re.findall(regex_kata, item)
14       list_singkatan.append(temp)
15
16  ## Tokenisasi
17  term = []
18  for index in range(len(list_kata)):
19       list_term = [list_kata[index]]
20       df = 0
21       for ulasan in list_ulasan:
22            jumlah = 0
23            ketemu = False
24            hasil_regex = re.findall(regex_kata,
25  ulasan.lower())
26            for kata in hasil_regex:
27                 if kata == list_kata[index]:
28                      jumlah += 1
29                      ketemu = True
30            if ketemu == True:
31                 df += 1
32            list_term.append(jumlah)
33       list_term.append(df)
34       term.append(list_term)
35
36  print ('\n## Term ##')
37  for index in term:
38       print (index)
```

Gambar 5.2 Implementasi Proses *Prepocessing* Data *Input*

Keterangan Gambar 5.2:

Baris 1-2 :Deklarasi variabel yang berisi perintah untuk menghilangkan angka, tanda baca, dll.

Baris 4-14 :Deklarasi variabel dengan nama *stopword*, *list_stopword*, *list*_kbbi, singkatan, *list*_singkatan yang digunakan untuk memastikan agar pada setiap dokumen yang akan diproses tidak ada lagi *stopword*.

Baris 16-34 :Merupakan proses tokenisasi atau pemisahan kalimat menjadi beberapa *term*. Perulangan yang pertama digunakan untuk menemukan setiap term yang ada pada daftar ulasan. Perulangan yang kedua digunakan untuk menambahkan daftar kata dan nilai DF jika *term* yang ditemukan bernilai *true*.

Baris 36-38 :Digunakan untuk mencetak daftar *term* yang telah ditemukan.

5.2.3 Implementasi Perhitungan Nilai TF-IDF

Tahapan perhitungan nilai TF-IDF dilakukan setelah proses *prepocessing* selesai dilakukan dan mendapatkan nilai pada masing-masing nilai *Term Frequency* (TF), dan *Inverse Document Frequency* (IDF) yang sebelumnya sudah dijelaskan pada Gambar 4.12 . Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.3 .

Baris Ke-	Kode Program
1	`## Tf dan Idf`
2	`tf = term`
3	`for x in tf:`
4	`    for i in range(len(x)):`
5	`        if (i != 0) and (i != (len(x)-1)):`
6	`            if x[i] != 0:`
7	`                x[i] = 1 + math.log10(x[i])`
8	`        elif (i == (len(x)-1)):`
9	`            x[i] = math.log10(len(list_ulasan)/x[i])`
10	
11	`print ('\n## IDF ##')`
12	`for x in tf:`
13	`    print (x)`
14	
15	`## Tf * Idf`
16	`tfIdf = []`
17	`for x in tf:`
18	`    temp = [x[0]]`
19	`    for i in range(len(x)):`
20	`        if (i != 0) and (i != (len(x)-1)):`
21	`            nilai = x[i] * x[len(x)-1]`
22	`            temp.append(nilai)`
23	`    tfIdf.append(temp)`
24	
25	`print ('\n## TF-IDF ##')`
26	`for x in tfIdf:`
27	`    print (x)`

Gambar 5.3 Implementasi Perhitungan Nilai TF-IDF

Katerangan Gambar 5.3:

Baris 1-7 :Deklarasi variabel tf, pada perulangan yang pertama digunakan untuk menyeleksi nilai TF, jika nilai !=0 maka akan masuk ke perhitungan `1 + math.log10(x[i])`.

Baris 8-13 :Perhitungan nilai IDF dengan rumus `math.log10(len(list_ulasan)/x[i])`, kemudian akan dicetak hasilnya.

Baris 15-27 :perhitungan nilai TF-IDF dengan mengalikan hasil dari masing-masing TF dan IDF yang hasilnya akan disimpan pada *array* tfidf[], kemudian sistem akan mencetak hasil.

5.2.4 Implementasi Perhitungan *Cosine Similarity*

Tahapan perhitungan nilai *cosine similarity* ini dilakukan setelah proses *prepocessing* dan perhitungan TF-IDF selesai dilakukan. Pada tahapan ini bertujuan untuk mendapatkan hasil dari analisis sentimen dengan mengurutkan hasil Cosim dari nilai tertinggi hingga terendah yang sebelumnya sudah dijelaskan pada Gambar 4.13 dan Gambar 4.14 . Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.4 .

Baris Ke-	Kode Program
1	`## Total per row`
2	`total = []`
3	`for x in range(len(list_ulasan)):`
4	`    total.append(0)`
5	
6	`for x in tfIdf:`
7	`    for i in range(len(total)):`
8	`        total[i] += x[i+1]**2`
9	`print ('\n## Total ##')`
10	`for x in range(len(total)):`
11	`    print ('U' + str(x+1) + ' = ' + str(total[x]))`
12	`## Norm Tf-Idf`
13	`normTfIdf = tfIdf`
14	`for x in normTfIdf:`
15	`    for i in range(len(total)):`
16	`        if (total[i] != 0):`
17	`            x[i+1] = x[i+1]/total[i]`
18	`print ('\n## Norm TF-IDF ##')`
19	`for x in normTfIdf:`
20	`    print (x)`
21	`## cos sim`
22	`cossim = []`
23	`for x in range(len(list_ulasan)-1):`
24	`    cossim.append([x, 0])`
25	`for x in normTfIdf:`
26	`    for i in range(len(cossim)):`
27	`        cossim[i][1] += x[i+1] * x[len(x)-1]`
28	`print ('\n## Cossim ##')`
29	`for x in range(len(cossim)):`
30	`    print ('U' + str(cossim[x][0] + 1) + ' = ' +`
31	`str(cossim[x][1]))`
32	`sorted_cossim = sorted(cossim, key=lambda csm: csm[1],`
33	`reverse = True)`
34	`print ('\n## Sorted Cossim ##')`
35	`for x in range(len(sorted_cossim)):`
36	`    print ('U' + str(sorted_cossim[x][0] +1 ) + ' = '`
37	`+ str(sorted_cossim[x][1]))`

Gambar 5.4 Implementasi Perhitungan Nilai *Cosine Similarity*

Keterangan Gambar 5.4:

Baris 1-12 :Deklarasi variabel total [] untuk menyimpan nilai total dari setiap *list* ulasan. Perulangan pertama digunakan untuk mengitung nilai total, perulangan kedua untuk

menempatkan nilai total pada variabel total [] yang sesuai dengan panjang *list* ulasan, kemudian nilai akan dicetak.

Baris 14-23 :Deklarasi variabel normTfidf = nilai tfidf yang digunakan untuk menyimpan nilai perkalian antara nilai TF dan IDF yang sudah dihitung sebelumnya dengan rumus `x[i+1] = x[i+1]/total[i]`, kemudian hasil akan dicetak.

Baris 25-37 :Deklarasi variabel cossim [] untuk menyimpan nilai perhitungan cossim. Perulangan pertama digunakan untuk menemukan panjang array yang akan diisi nilai cossim,perulangan kedua untuk menghitung nilai cossim dengan rumus `cossim[i][1] += x[i+1] * x[len(x) 1]`, berdasarkan nilai normTfidf yang sudah dihitung sebelumnya, kemudian hasil akan dicetak.

Baris 39-45 :Digunakan untuk mengurutkan nilai cossim yang sudah ditemukan dengan perintah `sorted_cossim = sorted(cossim, key=lambda csm: csm[1], reverse = True)`, kemudian hasil pengurutan akan dicetak.

5.2.5 Implementasi Proses Perbaikan Kata

Tahapan perbaikan kata digunakan untuk mencocokkan *term* yang sudah didapatkan dengan kamus yang sebelumnya sudah dijelaskan pada Gambar 4.15. Setelah itu hasil dari perbaikan akan ditampilkan pada sistem, dan dilakukan perhitungan proses *prepocessing* hingga *Cosine Similarity* kembali untuk mendapatkan hasil analisis sentimen. Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.5 .

Baris Ke-	Kode Program
1	`print("\n## Jaro ##")`
2	`for i in range(len(list_kata_ulasan)):`
3	`  for j in range(len(list_kata_ulasan[i])):`
4	`  for index in range(len(list_singkatan)):`
5	`  if list_kata_ulasan[i][j] ==`
6	`list_singkatan[index][0]:`
7	`  m = 0`
8	`  s1 = 0`
9	`  s2 = 0`
10	`  t = 0`
11	`  for huruf in`
12	`list_kata_ulasan[i][j]:`
13	`  s2 += 1`
14	`  if huruf in`
15	`list_singkatan[index][0]:`
16	`  m += 1`
17	`  for huruf in`
18	`list_singkatan[index][1]:`
19	`  s1 += 1`
20	`  dj = (1/3)* ((m/s1) + (m/s2) + ((m-`
21	`t)/m))`

```
22                          if dj > -1:
23                               dw = dj + (1 * 0.1 *(1 - dj))
24                               print(list_kata_ulasan[i][j],
25  list_singkatan[index][1])
26                               print(m, s1, s2, t)
27                               print(dj, dw)
28                               if dw > 0.75:
29                                    list_kata_ulasan[i][j] =
30  list_singkatan[index][1]
31
32  print("\n## List kata ulasan sesudah diperbaiki ##")
33  for item in list_kata_ulasan:
34       print(item)
35  print("\n## List Kata ##")
36  list_kata = []
37  for item in list_kata_ulasan:
38       for kata in item:
39            if (kata not in list_kata) and (kata not in
40  stopword):
41                 list_kata.append(kata)
42
43  for item in list_kata:
44       print(item)
```

Gambar 5.5 Implementasi Proses Perbaikan Kata

Keterangan Gambar 5.5:

Baris 1-4 :Digunakan untuk mencari *term* kata ulasan yang masuk dalam daftar kata tidak baku.

Baris 5-19 :Digunakan untuk pencocokan antara *term* yang akan diperbaiki dengan kamus singkatan dengan mencari nilai yang akan disimpan pada variabel *m, s1, s2, t* yang nantinya akan digunakan dalam proses perhitungan selanjutnya.

Baris 20-30 :Digunakan untuk menghitung jarak antar string dengan rumus `dj = (1/3)* ((m/s1) + (m/s2) + ((m-t)/m))`, dimana jarak yang dihasilkan tidak boleh kurang dari -1, dan nilai transposisinya dengan rumus `dw = dj + (1 * 0.1 *(1 - dj)`.

Baris 32-46 :Menampilkan hasil kata yang sudah diperbaiki.

5.2.6 Implementasi Proses Analisis Sentimen

Pada tahapan implementasi ini yang nantinya akan menjadi keluaran sistem yakni hasil analisis sentimen berupa analisis sentimen positif atau negatif dari masing-masing data uji yang dimasukkan baik dengan menggunakan perbaikan kata atau tanpa menggunakan perbaiakn kata. Implementasi sistem ditunjukan dengan potongan kode program pada Gambar 5.6 .

Baris Ke-	Kode Program
1	`total_sentimen = 0`
2	`for x in range(k):`
3	`    total_sentimen +=`
4	`list_sentimen_ulasan[sorted_cossim[x][0]]`
5	`    print(sorted_cossim[x][0]+1,`
6	`list_sentimen_ulasan[sorted_cossim[x][0]],`
7	`total_sentimen)`
8	`print("\n## Sentimen ##")`
9	`if total_sentimen >= int (k*threshold):`
10	`    print("Positif")`
11	`else:`
12	`    print("negatif")`
13	`list_kata_ulasan = []`
14	`for item in list_ulasan:`
15	`    temp = [x.lower() for x in re.findall(regex_kata,`
16	`item)]`
17	`    list_kata_ulasan.append(temp)`

Gambar 5.6 Implementasi Proses Analisis Sentimen

Keterangan Gambar 5.6:

Baris 1-7 :Digunakan untuk menghitung total sentimen berdasarkan nilai cossim yang sudah terurut.

Baris 8-17 :Mencetak hasil analisis sentimen berupa positif atau negatif dengan *input*an user berupa nilai *k*.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai pengujian dan analisis sistem berdasarkan metodologi, perancangan, dan implementasi yang telah dijelaskan pada bab sebelumnya.

6.1 Pengujian Pengaruh Nilai *k*

Pada proses pengujian yang pertama ini adalah pengujian pengaruh nilai *k*. Proses pengujian dilakukan untuk menemukan pada nilai *k* yang paling baik atau optimal yang digunakan dalam proses klasifikasi K-NN yang nantinya akan mempengarui hasil *accuracy* dari sistem. Nilai *k* yang digunakan pada pengujian ini bervariatif yang mana setiap nilai *k* akan menjadi parameter dari pengujian. Pengujian ini dilakukan dengan dua tahapan yakni tahap 1 untuk proses *prepocessing* tanpa perbaikan kata, dan tahap 2 untuk *prepocessing* dengan perbaikan kata.

6.1.1 Pengujian Pengaruh Nilai *k* Tanpa Perbaikan Kata

Pada pengujian pertama tanpa perbaikan kata dilakukan untuk mengetahui nilai *k* mana yang merupakan solusi paling baik atau optimal yang digunakan dalam proses klasifikasi, pengujian ini menggunakan 450 data latih dengan 50 data uji. Pengujian pengaruh nilai *k* dengan proses *prepocessing* tanpa perbaikan kata atau tahap 1 ditunjukkan pada tabel 6.1 .

Tabel 6.1 Pengujian Pengaruh Nilai K Tanpa Perbaiakan Kata

No	Nilai *k*-	*Precision*	*Recall*	*Accuracy* (%)
1	3	0,72	1	72
2	5	0,72	1	72
3	10	0,72	1	72
4	15	0,72	1	72
5	20	0,74	1	74
6	25	0,72	1	72
7	30	0,72	1	72
8	40	0,7	1	70
9	50	0,68	1	68

Berdasarkan Tabel 6.1, bisa kita lihat jika nilai *recall* untuk setiap nilai *k* untuk seluruh data yang diujikan memperoleh nilai yang optimal yaitu 1. Sedangkan untuk perolehan nilai *precision* pada seluruh data mendapatkan hasil nilai yang berbeda. Perubahan nilai *precision* dipengaruhi oleh jumlah kesesuaian antara analisis yang dilakukan pakar dan sistem. Pada tabel diatas diketahui jika nilai *k* yang optimal adalah pada *k*=20 dengan *accuracy* sebesar 74 %.

6.1.2 Pengujian Pengaruh Nilai *k* Dengan Perbaikan Kata

Pada pengujian pertama dengan perbaikan kata dilakukan untuk mengetahui nilai *k* mana yang merupakan solusi paling baik atau optimal yang digunakan dalam proses klasifikasi, pengujian ini menggunakan 450 data latih dengan 50 data uji. Pengujian pengaruh nilai *k* dengan proses *prepocessing* dengan perbaikan kata atau tahap 2 ditunjukkan pada tabel 6.2 .

Tabel 6.2 Pengujian Pengaruh Nilai K Dengan Perbaikan Kata

No	Nilai *k*-	*Precision*	*Recall*	*Accuracy* (%)
1	3	0,74	1	74
2	5	0,74	1	74
3	10	0,74	1	74
4	15	0,74	1	74
5	20	0,76	1	76
6	25	0,7	1	70
7	30	0,7	1	70
8	40	0,7	1	70
9	50	0,68	1	68

Berdasarkan Tabel 6.2, bisa kita lihat jika nilai *recall* untuk setiap nilai *k* untuk seluruh data yang diujikan memperoleh nilai yang optimal yaitu 1. Sedangkan untuk perolehan nilai *precision* pada seluruh data mendapatakan hasil nilai yang berbeda. Perubahan nilai *precision* dipengaruhi oleh jumlah kesesuaian antara analisis yang dilakukan pakar dan sistem. Pada tabel diatas diketahui jika nilai *k* yang optimal adalah pada *k*=20 dengan nilai *accuracy* sebesar 76 %.

6.2 Pengujian Pengaruh Banyaknya Data Latih

Proses pengujian kedua dilakukan berdasarkan nilai *k* terbaik yang sudah didapatkan dari proses pengujian yang telah dilakukan sebelumnya. Dari nilai *k* tersebut akan dilakukan pengujian dengan memberikan data latih secara bertahap dari 250 data, 300 data, 350 data, 400 data, hingga 450 data dengan setiap data latih yang diberikan diujikan dengan 50 data uji. Pada pengujian ini bertujuan untuk mendapatkan jumlah data latih yang memberikan hasil *accuracy* optimal terhadap sistem. Pengujian ini dilakukan dengan dua tahapan yakni tahap 1 untuk proses *prepocessing* tanpa perbaikan kata, dan tahap 2 untuk *prepocessing* dengan perbaikan kata.

6.2.1 Pengujian Pengaruh Banyaknya Data Latih Tanpa Perbaikan Kata

Pada pengujian ini dilakukan berdasarkan nilai *k* terbaik yang sudah didapatkan dari pengujian sebelumnya, pengujian ini menggunakan data latih secara bertahap dari 250 data, 300 data, 350 data, 400 data, hingga 450 data dengan setiap data latih yang diberikan diujikan dengan 50 data uji. Pengujian pengaruh banyaknya data latih dengan proses *prepocessing* tanpa perbaikan kata atau tahap 1 ditunjukkan pada tabel 6.3 .

Tabel 6.3 Pengujian Pengaruh Banyaknya Data Latih Tanpa Perbaikan Kata

No	Jumlah Data	*Precision*	*Recall*	*Accuracy* (%)
1	250	0,72	1	72
2	300	0,7	1	70
3	350	0,7	1	70
4	400	0,7	1	70
5	450	0,74	1	74

Berdasarkan Tabel 6.3, bisa kita lihat jika nilai *recall* untuk setiap nilai *k* untuk seluruh data yang diujikan memperoleh nilai yang optimal yaitu 1. Sedangkan untuk perolehan nilai *precision* pada seluruh data mendapatakan hasil nilai yang berbeda. Perubahan nilai *precision* dipengaruhi oleh jumlah kesesuaian antara analisis yang dilakukan pakar dan sistem. Pada tabel diatas diketahui jika nilai *accuracy* yang optimal adalah pada data ke 450 dengan *accuracy* sebesar 74 % .

6.2.1 Pengujian Pengaruh Banyaknya Data Latih Dengan Perbaikan Kata

Pada pengujian ini dilakukan berdasarkan nilai *k* terbaik yang sudah didapatkan dari pengujian sebelumnya, pengujian ini menggunakan data latih secara bertahap dari 250 data, 300 data, 350 data, 400 data, hingga 450 data dengan setiap data latih yang diberikan diujikan dengan 50 data uji. Pengujian pengaruh banyaknya data latih dengan proses *prepocessing* dengan perbaikan kata atau tahap 1 ditunjukkan pada tabel 6.4 .

Tabel 6.4 Pengujian Pengaruh Banyaknya Data Latih Dengan Perbaikan Kata

No	Jumlah Data	*Precision*	*Recall*	*Accuracy* (%)
1	250	0,72	1	72
2	300	0,7	1	70
3	350	0,66	1	66
4	400	0,64	1	64
5	450	0,76	1	76

Berdasarkan Tabel 6.4, bisa kita lihat jika nilai *recall* untuk setiap nilai *k* untuk seluruh data yang diujikan memperoleh nilai yang optimal yaitu 1. Sedangkan untuk perolehan nilai *precision* pada seluruh data mendapatkan hasil nilai yang berbeda. Perubahan nilai *precision* dipengaruhi oleh jumlah kesesuaian antara analisis yang dilakukan pakar dan sistem. Pada tabel diatas diketahui jika nilai *accuracy* yang optimal adalah pada data ke 450 dengan *accuracy* sebesar 76 % .

6.3 Pengujian Pengaruh Penggunaan Perbaikan Kata

Pada pengujian ini dilakukan untuk mengetahui nilai dari *precision*, *recall*, dan *accuracy* pada masing-masing pengujian dengan menggunakan nilai *k* dan data latih optimal yang didapatkan. Pengujian ini dilakukan dengan dua tahapan yakni tahap 1 untuk proses *prepocessing* tanpa perbaikan kata, dan tahap 2 untuk *prepocessing* dengan perbaikan kata. Hasil pengujian ditunjukkan pada Tabel 6.5 .

Tabel 6.5 Pengujian Pengaruh Penggunaan Perbaikan Kata

Pengujian	*Precision*	*Recall*	*Accuracy* (%)
Tanpa Perbaikan Kata (Tahap 1)	0,74	1	74
Dengan Perbaikan Kata (Tahap 2)	0,76	1	76

Berdasarkan Tabel 6.5, dengan didapatkan hasil nilai *accuracy* terbaik terdapat pada 450 data latih, maka untuk analisis sentimen tanpa perbaikan kata nilai *precision* sebesar 0,74, nilai *recall* sebesar 1, dan nilai *accuracy* sebesar 74 %. Sedangkan untuk analisis sentimen dengan perbaikan kata nilai *precision* sebesar 0,76, nilai *recall* sebesar 1, dan nilai *accuracy* sebesar 76 %. Sehingga analisis sentimen akan lebih baik jika dilakukan perbaikan kata terlebih dahulu.

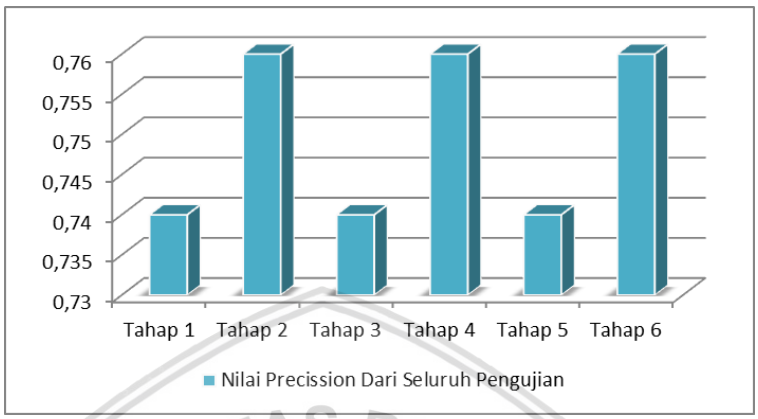
6.4 Hasil Dari Seluruh Pengujian

Pada pengujian ini dilakukan untuk mengetahui hasil dari setiap proses pengujian yang telah dilakukan, hasil dari setiap pengujian akan ditampilkan dalam bentuk grafik yang terdiri dari nilai *precision*, *recall*, dan *accuracy*. Pada grafik hasil seluruh pengujian akan menampilkan 6 tahapan yang ada pada setiap proses pengujian dengan urutan sebagai berikut :

- Tahap 1 : Pengaruh nilai *k* tanpa perbaikan kata.
- Tahap 2 : Pengaruh nilai *k* dengan perbaikan kata.
- Tahap 3 : Pengaruh banyaknya data latih tanpa perbaikan kata.
- Tahap 4 : Pengaruh banyaknya data latih dengan perbaikan kata.
- Tahap 5 : Pengaruh penggunaan perbaikan kata tanpa perbaikan kata di proses *prepocessing*.
- Tahap 6 : Pengaruh penggunaan perbaikan kata dengan perbaikan kata di proses *prepocessing*.

6.4.1 Hasil Seluruh Pengujian *Precision*

Hasil pengujian akan ditampilkan dalam bentuk grafik nilai *precision* pada masing-masing pengujian yang telah dilakukan. Grafik hasil pengujian untuk seluruh nilai *precision* akan ditunjukkan pada Gambar 6.1 .

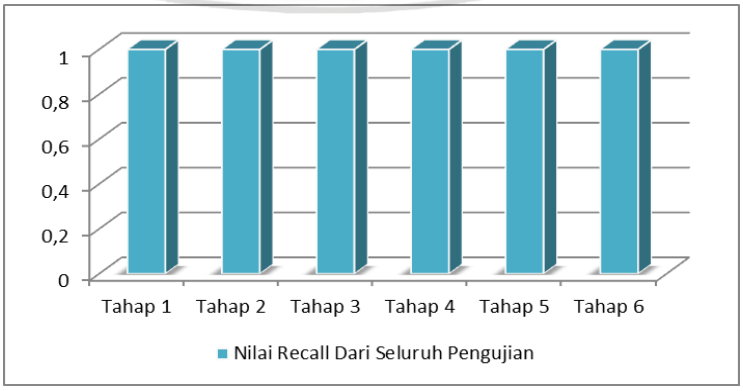


Gambar 6.1 Nilai *Precision* Dari Seluruh Pengujian

Berdasarkan Gambar 6.1, menunjukkan hasil *precision* dari seluruh pengujian maka dapat disimpulkan jika nilai *precision* akan cenderung lebih tinggi ketika terdapat perbaikan kata terlebih dahulu karena ketika dalam *prepocessing* kata yang tidak baku akan diperbaiki menggunakan kamus singkatan. Nilai *precision* ditentukan oleh tingkat ketepatan prediksi sistem sesuai dengan perhitungan prediksi benar dari seluruh total data latih, maupun prediksi salah dari seluruh total data latih yang diberikan.

6.4.2 Hasil Seluruh Pengujian *Recall*

Hasil pengujian akan ditampilkan dalam bentuk grafik nilai *recall* pada masing-masing pengujian yang telah dilakukan. Grafik hasil pengujian untuk seluruh nilai *recall* akan ditunjukkan pada Gambar 6.2 .

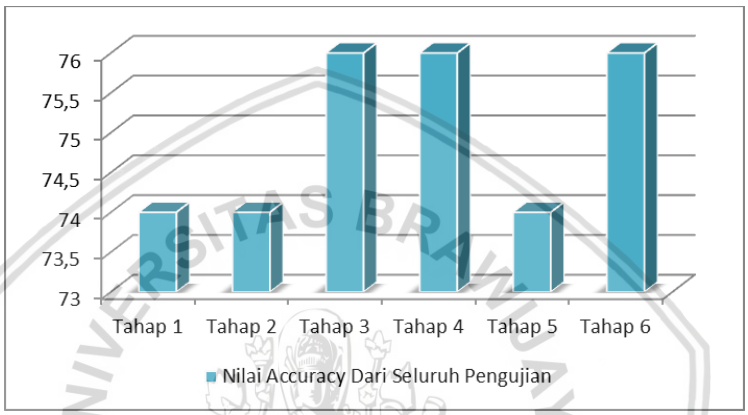


Gambar 6.2 Nilai Recall Dari Seluruh Pengujian

Berdasarkan Gambar 6.2, menunjukkan hasil *recall* dari seluruh pengujian maka dapat disimpulkan jika nilai *recall* menunjukkan hasil yang optimal yaitu 1 yang berarti tingkat keberhasilan sistem dalam mengenali suatu kelas sangat baik, tidak hanya pada pengujian dengan perbaikan kata, namun juga dalam pengujian tanpa perbaikan.

6.4.3 Hasil Seluruh Pengujian *Accuracy*

Hasil pengujian akan ditampilkan dalam bentuk grafik nilai *recall* pada masing-masing pengujian yang telah dilakukan. Grafik hasil pengujian untuk seluruh nilai *recall* akan ditunjukkan pada Gambar 6.3 .



Gambar 6.3 Nilai Accuracy Dari Seluruh Pengujian

Berdasarkan Gambar 6.3, menunjukkan hasil *accuracy* dari seluruh pengujian maka dapat disimpulkan jika nilai *accuracy* menunjukkan hasil yang tinggi ketika pada pengujian dengan perbaikan kata, karena ketika dalam *prepocessing* kata yang tidak baku akan diperbaiki menggunakan kamus singkatan. Namun, dalam pengujian tahap 3 yakni pengujian tentang pengaruh banyaknya data latih tanpa perbaikan kata juga mendapatkan *accuracy* yang tinggi karena pada pengujian tersebut sebelumnya sudah dilakukan pengujian untuk mengetahui pengaruh nilai *k*, dan didapatkan nilai *k*=20 yang paling optimal. Sehingga dari nilai *k* tersebut mempengarui *accuracy* dari kedua pengujian pengaruh banyaknya data latih, baik tanpa perbaikan kata maupun dengan perbaikan kata keduanya memperoleh hasil *accuracy* yang optimal.

BAB 7 PENUTUP

Bab ini akan membahas tentang kesimpulan dari penelitian yang telah dilakukan tentang Analisis Sentimen Pada Ulasan *"Lazada"* Berbahasa Indonesia Menggunakan *K-Nerarest Neighbor* (K-NN) Dan Perbaikan Kata Menggunakan *Jaro Winkler Distance*, serta saran yang diberikan oleh penulis terkait penelitian yang serupa untuk penelitian selanjutnya.

1.1 Kesimpulan

Setelah seluruh rangkaian proses mulai dari studi pustaka hingga proses pengujian dan analisis telah selesai dilakukan untuk permasalahan analisis sentimen dan perbaikan kata menggunakan *K-Nearest Neighbor* (K-NN) dan *Jaro Winkler Distance*, sehingga didapatkan kesimpulan sebagai berikut:

1. Untuk metode *K-Nearest Neighbor* (K-NN) dapat diterapkan dan dilakukan proses implementasi dengan baik untuk menentukan analisis sentimen pada ulasan *"Lazada"* berbahasa indonesia.
2. Dalam proses perbaikan kata menggunakan metode *Jaro Winkler Distance* dapat menghasilkan perbaikan kata yang baik dan sesuai dengan perhitungan manual yang dilakukan oleh user.
3. Hasil *precision* dan *accuracy* pada setiap skenario pengujian baik dalam analisis sentimen tanpa menggunakan perbaikan kata serta dengan perbaikan kata mendapatkan nilai yang beragam, namun untuk nilai *recall* tetap sama dengan nilai optimal yakni 1.
   - Pengujian pertama dilakukan dengan pengujian pengaruh nilai *k*, data uji yang diberikan sebanyak 50 data, dan untuk seluruh data latih sebanyak 450 data. Pengujian diambil nilai *precision*, *recall*, dan *accuracy* berdasarkan nilai *k* yang diambil secara acak yakni *k*=3, *k*=5, *k*=10, *k*=15, *k*=20, *k*=25, *k*=30, *k*=40, *k*=50, hasil *accuracy* terbaik ditemukan pada nilai *k*=20 dengan nilai *accuracy* untuk analisis sentimen tanpa perbaikan kata sebesar 74 %, dan analisis sentimen dengan perbaikan kata sebesar 76 %. Namun nilai *precision* akan cenderung menurun seiring bertambah besarnya nilai *k* yang diberikan.
   - Pengujian kedua dilakukan dengan pengujian pengaruh banyaknya data latih dengan nilai *k* optimum yang telah didapatkan untuk mengetahui pada data latih berapakah yang memberikan nilai *accuracy* terbaik. Data latih yang diujikan diberikan secara bertahap yakni 250 data, 300 data, 350 data, 400 data, dan 500 data. Setelah proses pengujian dilakukan, hasil *precision*, *recall*, dan *accuracy* terbaik terdapat pada 450 data latih dengan nilai *precision* sebesar 0,74, nilai *recall* sebesar 1, dan nilai *accuracy* sebesar 74 % untuk analisis sentimen tanpa perbaikan kata, serta nilai *precision* sebesar 0,76, nilai *recall* sebesar 1, dan nilai *accuracy* sebesar 76 % untuk analisis sentimen dengan perbaikan kata.
   - Pengujian ketiga dilakukan dengan pengujian pengaruh penggunaan perbaikan kata. Proses pengujian dilakukan dengan menggunakan nilai *k*

optimum, dan jumlah data latih optimum yang telah didapatkan. Nilai *k* dan jumlah data latih tersebut akan diujikan pada masing-masing proses analisis sentimen tanpa perbaikan kata dengan hasil akurasi 74 %, dan proses analisis sentimen menggunakan perbaikan kata dengan hasil akurasi sebesar 76 %.

4. Dari ketiga hasil pengujian yang telah dilakukan, maka dapat disimpulkan bahwa analisis sentimen menggunakan perbaikan kata lebih baik daripada analisis sentimen tanpa perbaikan kata yang dapat dilihat pada nilai *precision*, dan *accuracy*.

1.2 Saran

Sistem Analisis Sentimen Pada Ulasan *"Lazada"* Berbahasa Indonesia Menggunakan *K-Nerarest Neighbor* (K-NN) Dan Perbaikan Kata Menggunakan *Jaro Winkler Distance* diharapkan mampu berkembang lebih baik lagi dengan berusaha memperbaiki kekurangan pada sistem ini.

1. Saran dari penulis yang semoga bermanfaat bagi penelitian selanjutnya yang serupa dengan sistem ini, khususnya pada perbaikan kata pada kamus singkatan berbahasa indonesia yang digunakan sebagai acuan ditambahkan lagi yang lebih lengkap dengan istilah-istilah singkatan lain yang sering digunakan dalam penulisan ulasan.
2. Kata yang *termasuk* dalam *stopword* bahasa indonesia lebih diperhatikan kembali karena pada sebuah ulasan banyak digunakan kata yang *termasuk* dalam *stopword*.

DAFTAR PUSTAKA

Abbott, D. (2013). *Introduction to Text Mining Virtual Data Intensive Summer School*.

Adikara, P. P. (2012, November 17). Dipetik November 05, 2017, dari Kamus Kata Dasar Dan Stopword List Bahasa Indonesia: http://hikaruyuuki.lecture.ub.ac.id/kamus-kata-dasar-dan-stopword-list-bahasa-indonesia/

Antinasari, P., Perdana, R. S., & Fauzi, M. A. (2017). *Analisis Sentimen Tentang Opini Film Pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naive Bayes Dengan Perbaikan Kata Baku*, 1-9.

Baskoro, S., Ahmad, R., & Furqon, M. (2015). *Pencarian Pasal Pada Kitab Undang-Undang Hukum Pidana (KUHP) Berdasarkan Kasus Menggunakan Metode Cosine Similarity dan Latent Semantic Indexing (LSI)*, 1-6.

Christian, H., Agus, M. P., & Suhartono, D. (2016). *SINGLE DOCUMENT AUTOMATIC TEXT SUMMARIZATION USING TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)*, 1-10.

Dao, T., & Simpson, T. (2006). *Measuring Similarity Between Sentences*.

Dzikrulloh, N. N., Indriati, & Setiawan, B. D. (2017). Metode K-Nearest Neighbor (KNN). *Penerapan Metode K–Nearest Neighbor (KNN) dan Metode Weighted Product (WP) dalam penerimaan Calon Guru Dan Karyawan Tata Usaha Baru Berwawasan Teknologi (Studi Kasus : Sekolah Menengah Kejuruan Muhammadiyah 2 KEdiri)*, 2.

Edward H. Porter, a. W. (t.thn.). Approximate String Comparison and its Effect on an Advance Record Linkage System. 1-11.

Estu Nurjanah, W., Setya Pradana, R., & Ali Fauzi, M. (2017). *Analisis Sentimen Terhadap Tayangan Televisi Berdasarkan Opini Masyarakat pada Media Sosial Twitter menggunakan Metode K-Nearest Neighbor dan Pembobotan Jumlah Retweet*, 1750-1757.

Fahma, A. I., Cholissodin, I., & Perdana, R. S. (2018). *Identifikasi Kesalahan Penulisan Kata (Typographical Error) pada Dokumen Berbahasa Indonesia Menggunakan Metode N-gram dan Levenshtein Distance*, 1-10.

Hasugian, J. (2006). *Penelusuran Informasi Ilmiah Secara Online: Perlakuan terhadap Seorang Pencari Informasi sebagai Real User* , 1-13.

Kurniawati, A., Pupitodjati, S., & Rahman, S. (2016). *Implementasi Algoritme Jaro Winkler Distance Untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia*, 1-4.

Lestari, N. P. (2015). *UJI RECALL AND PRECISION SISTEM TEMU KEMBALI INFORMASI OPAC PERPUSTAKAAN ITS SURABAYA*, 1-18.

Na'firul, H. A., Sutardi, & Rahmat, R. (2016). *Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance*, 1-8.

Nanda, R. (2015, October 8). Dipetik November 24, 2017, dari TEKS ULASAN : PENGERTIAN, STRUKTUR DAN CONTOH TEKS ULASAN: http://www.materikelas.com/teks-ulasan-pengertian-struktur-dan-contoh-teks-ulasan/

Pustejovsky, J., & Stubbs, A. (2012). *Natural language Annotation for machine Learning.* Cambrige: O'relly.

Robertson, S. (2004). *Understanding Inverse Document Frequency:On theoretical arguments for IDF*, 1-19.

Rocmawati, R., & Kusumaningrum, R. (2015). *Studi Perbandingan Algoritme Pencarian String Dalam Metode Approximate String Matching Untuk Identifikasi Kesalahan Pengetikan Teks*, 1-11.

Samuel, Y., Delima, R., & Rachmat, A. (2014). *Implementasi Metode K-Nearest Neighbor dengan Decision Rule untuk Klasifikasi Subtopik Berita*, 1-15.

Sasongko, W. J., & Hartati, S. (2011). *Text Document Retrieval In English Using Keywords of Indonesian Dictionary Based*, 1-7.

Xia, T., & Chai, Y. (2011). *An Improvement to TF-IDF: Term Distribution based Term Weight Algorithm*, 1-8.