ANALISIS PERBANDINGAN SENTIMEN PADA DATA *REVIEW* PENGGUNA APLIKASI NOVEL *ONLINE* (WATTPAD DAN DREAME) DI GOOGLE PLAY STORE MENGGUNAKAN METODE SUPPORT VECTOR MACHINE DAN ASOSIASI

Tugas Akhir

Disusun sebagai salah satu syarat

untuk memperoleh gelar Ahli Madya

Program Studi Statistika Terapan dan Komputasi

oleh

Septi Hayuning Tyas

4112317018

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2020

ANALISIS PERBANDINGAN SENTIMEN PADA DATA *REVIEW* PENGGUNA APLIKASI NOVEL *ONLINE* (WATTPAD DAN DREAME) DI GOOGLE PLAY STORE MENGGUNAKAN METODE SUPPORT VECTOR MACHINE DAN ASOSIASI

Tugas Akhir

Disusun sebagai salah satu syarat

untuk memperoleh gelar Ahli Madya

Program Studi Statistika Terapan dan Komputasi

oleh

Septi Hayuning Tyas

4112317018

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2020

PERNYATAAN

Dengan ini, saya

nama : Septi Hayuning Tyas

NIM : 4112317018

program studi : Statistika Terapan dan Komputasi

menyatakan bahwa tugas akhir berjudul Analisis Perbandingan Sentimen pada Data *Review* Pengguna Aplikasi Novel *Online* (Wattpad dan Dreame) di Google Play Store Menggunakan Metode *Support Vector Machine* dan Asosiasi ini benar- benar karya saya dan bebas plagiat. Atas pernyataan ini, saya pribadi siap menanggung resiko apabila di kemudian hari terbukti terdapat plagiat dalam tugas akhir ini, maka saya bersedian menerima sanksi ketentuan peraturan perundang-undangan.

Semarang, 24 September 2020

Peneliti

Septi Hayuning Tyas

4112317018

PENGESAHAN

Tugas Akhir berjudul *Analisis Perbandingan Sentimen pada Data Review Pengguna Aplikasi Novel Online (Wattpad dan Dreame) di Google Play Store Menggunakan Metode Support Vector Machine dan Asosiasi* karya Septi Hayuning Tyas NIM 4112317018 ini telah dipertahankan dalam Ujian Tugas Akhir FMIPA Universitas Negeri Semarang pada tanggal 7 Oktober 2020 dan disahkan oleh Panitia Ujian.

Semarang, 1 November 2020

Panitia

Ketua,

Dr. Suginto, M.Si.
NIP 196102191993031001

Sekretaris,

Dr. Mulyono, M.Si.
NIP 197009021997021001

Penguji I,

Prof. Y.L. Sukestiyarno, M.S., Ph.D.
NIP 195904201984031002

Penguji II/Pembimbing,

Amidi, S.Si., M.Pd.
NIP 198703012014041001

MOTTO

- *Dream it, wish it, do it!*
- Jangan pernah menunggu waktu, karena waktu tak akan pernah mau menunggu.
- *Ask yourself if what you're doing today will get you closer to where you want to be tomorrow* (Anonymous).

PERSEMBAHAN

Dengan rasa syukur kepada Allah SWT, atas segala kuasa-Nya tugas akhir ini kupersembahkan kepada:

1. Bapak dan Ibuku tercinta yang senantiasa memberikan dukugan moril dan materiil.
2. Bapak Amidi, S.Si., M.Pd. yang senantiasa membimbing saya hingga akhir
3. Kakak-kakak dan adikku yang selalu memberi motivasi dan *support* selama mengerjakan tugas akhir ini.
4. Teman-temanku yang membantu dan mendukung saya dalam menyelesaikan tugas akhir ini.
5. Almamaterku.

PRAKATA

Segala puji dan syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul **"Analisis Perbandingan Sentimen pada Data *Review* Pengguna Aplikasi Novel *Online* (Wattpad dan Dreame) di Google Play Store Menggunakan Metode *Support Vector Machine* dan Asosiasi"** ini sebagai salah satu syarat untuk memperoleh gelar Diploma pada Program Studi Statistika Terapan dan Komputasi Fakultas Matematika dan Ilmu Pengetahuan Alam.

Penyelesaian Tugas Akhir ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang,
2. Dr. Sugianto, M.Si, Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang,
3. Dr. Mulyono, M.Si., Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang,
4. Dr. Iqbal Kharisudin, M.Sc., Koordinator Program Studi Statistika Terapan dan Komputasi,
5. Amidi, S.Si., M.Pd., Dosen pembimbing, yang telah memberikan bimbingan dan pengarahan dalam penyusunan Tugas Akhir ini,
6. Bapak dan Ibu tercinta yang selalu memberikan dukungan moril dan materiil dalam menyelesaikan Tugas Akhir,
7. Kakak-kakak dan adikku yang selalu memberkan semangat, motivasi dan doa dalam menyelesaikan Tugas Akhir,
8. Teman-teman terbaik Anggi, Alifia, Reza, Alisha dan Linda yang selalu mau direpoti, mendukung, mendoakan dan menyemangati dalam pengerjaan Tugas Akhir,
9. Ayu dan Elisa, Sahabat-sahabat terbaik, yang selalu menemani, menyemangati dan memotivasi selama mengerjakan Tugas Akhir,
10. Teman-teman yang tak bisa disebutkan satu-satu, yang memberikan *support* dan doa, dan

11. Semua pihak yang telah membantu dalam penyusunan Tugas Akhir baik secara langsung maupun tidak langsung.

Semoga segala bantuan dan doa yang telah diberikan, akan mendapat balasan dari Allah SWT. Penulis menyadari, Tugas Akhir ini masih banyak kekurangan karena keterbatasan kemampuan dan pengetahuan penulis. Penulis berharap kritik dan saran yang membangun guna untuk melengkapi dan memperbaiki dalam penelitian. Semoga Tugas Akhir ini dapat berguna serta bermanfaat bagi pembaca.

Semarang, Agustus 2020

Penulis

ABSTRAK

Septi Hayuning Tyas. 2020. *Analisis Perbandingan Sentimen pada Data Review Pengguna Aplikasi Novel Online (Wattpad dan Dreame) di Google Play Store Menggunakan Metode Support Vector Machine dan Asosiasi*. Tugas Akhir, Statistika Terapan dan Komputasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Amidi S.Si., M.Pd.

Kata Kunci: Sentimen, *Review*, Wattpad, Dreame, Asosiasi, SVM, Klasifikasi

Di era digital sekarang ini, semua kebutuhan akan kegiatan sudah tersedia dalam satu genggaman. Kemajuan teknologi mendorong manusia untuk dapat melakukan segala aktivitas secara efisien. Pada era digital seperti ini, hobi kini dapat dilakukan di manapun dan kapanpun secara efisien, seperti hobi membaca dan menulis, saat ini terdapat aplikasi yang dapat digunakan untuk menyalurkan hobi membaca dan menulis seperti Wattpad dan Dreame. Ada kelebihan dan kelemahan dari aplikasi Wattpad dan Dreame, ada berbagai komentar dalam situs Google Play Store. *Review* aplikasi pada Google Play Store dapat dijadikan indikasi baik atau buruknya suatu aplikasi dan sebagai pencarian masalah yang dikeluhkan pengguna terhadap aplikasi. Analisis deskriptif yang diperoleh dengan *rating* dari total 4137 *review* pengguna Wattpad diperoleh sebanyak 33,12% pengguna sangat tidak menyukai, 17,04% tidak suka, 15,71% netral, 11,31% menyukai dan 22,82%, sedangkan untuk aplikasi Dreame dari 3090 *review* yang diperoleh, sebanyak 10,94% pengguna sangat tidak menyukai, 13,24% tidak menyukai, 27,86% netral, 21,85% menyukai dan 26,12% pengguna sangat menyukai. Maka, berdasakan *rating*, Dreame lebih unggul dari Wattpad. Dengan menggunakan *machine learning Support Vector Machine*, akan dilakukan prediksi yang divisualisasika dalam bentuk *wordcloud* dan kemudian dicari asosiasi kata untuk memecahkan masalah. Dalam pelabelan data berdasarkan Vader Lexicon diperoleh hasil sentimen negatif sebanyak 1058 dan positif 2356 untuk aplikasi Wattpad, sedangkan untuk aplikasi Dreame diperoleh sebanyak 761 data negatif dan 1865 data positif. Dengan metode SVM diprediksi dengan akurasi untuk aplikasi Wattpad sebesar 88,60% dan aplikasi Dreame sebesar 87,45%. Pada kelas negatif, asosiasi kata diekstraksi menjadi informasi untuk menemukan faktor-faktor keluhan pengguna menggunakan diagram *fishbone*, dapat diketahui, ada 15 masalah yang dikeluhkan pengguna Wattpad dan sebanyak 17 masalah yang dikelhkan oleh penguna Dreame yang kemudian diklasifikasikan menjadi faktor-faktor yang meliputi, proses, produk, pelaku/orang, tempat, harga serta promosi yang ada pada kedua aplikasi, dapat dicari pemecahan dari masalah-masalah yang ditemukan. Dari banyaknya keluhan dapat diketahui, masalah pada Dreame lebih banyak dari Wattpad, sehingga perlu lebih banyak perbaikan untuk aplikasi Dreame.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi semakin berkembang dan memudahkan masyarakat dalam berbagai hal, termasuk gaya hidup dan hobi yang kini bisa dipermudah dengan adanya teknologi. Ada berbagai macam aplikasi yang disediakan yang dapat memudahkan masyarakat untuk menjalankan kegiatan sehari – hari. Segala kebutuhan masyarakat kini bisa didapat dalam satu genggaman, termasuk hobi membaca atau menulis. Untuk itu ada beberapa apllikasi novel daring yang lebih efiesien dan murah bagi yang memiliki hobi membaca, selain itu aplikasi-aplikasi tersebut juga menyediakan fasilitas untuk menulis yang bisa dipublikasikan dan dibaca oleh pengguna di aplikasi tersebut. Performa aplikasi terkadang membuat pengguna merasa kecewa karena tidak sesuai dengan apa yang diharapkan. Hal ini membuat pengguna merasa menyesal telah men*download* aplikasi tersebut, sehingga pada Google Play menyediakan ulasan yang dapat digunakan pengguna aplikasi untuk menyampaikan keluhan baik positif maupun negatif.

Data *review* pada Google Play begitu banyak dan beragam, akan sulit untuk mengetahui apakah aplikasi ini baik atau tak layak karena sistemnya yang rumit. Dengan klasifikasi data review menjadi positif dan negatif akan mempermudah bagi perusahaan melakukan pengembangan aplikasi sesuai dengan harapan masyarakat. Selain itu dapat dibandingkan pula antar dua aplikasi serupa sehingga masyarakat akan lebih bisa memilih yang sesuai dengan keinginannya. Untuk aplikasi novel daring bisa digunakan contoh Wattpad dan Dreame yang memiliki pengguna cukup banyak.

Melalui *review* dari pengguna diharapkan akan memudahkan masyarakat yang akan menggunakan aplikasi dapat terbantu dengan adanya *review* ini. Akan tetapi, Masyarakat yang ingin mengunduh aplikasi tersebut sulit untuk memilih mana yang lebih baik karena begitu banyaknya *review*

pengguna yang masuk, maka diperlukan klasifikasi data *review* yang memberikan perbandingan positif dan negatif. Klasifikasi data *review* pengguna berdasarkan jenis sentimen akan mempermudah perusahaan aplikasi dan masyarakat untuk menilai mana yang lebih baik dari kedua aplikasi berdasarkan *review* dari pengguna aplikasi tersebut.

Data *review* yang berisi begitu banyak kalimat dan kata akan terasa sulit dilakukan analisis, maka dengan berbagai proses data melalui program yang dibuat sedemikian rupa akan mempermudah menganalisis dan melakukan klasifikasi kata. Sehingga dengan pelabelan dapat dicari analisis sentimennya. Dibutuhkan *machine learning* untuk melakukan klasifikasi berdasarkan sentiment pengguna Dreame maupun Wattpad. Untuk proses pengklasifikasian dilakukan dengan metode *Support Vector Machine* (SVM). Selain itu dari data *review* akan menggunakan asosiasi.

Dengan analisis sentimen ini, akan didapatkan hasil akhir presentase ulasan potif dan negatif pada aplikasi Wattpad dan Dreame, serta didapatkan kata apa yang sering muncul dalam ulasan positif dan negatif, sehingga peruhasaan akan mempertahankan apa yang disukai dan tidak disukai oleh pengguna, selain itu, masyarakat akan tahu apa saja fitur yang baik dan tidak baik dalam aplikasi tersebut, dan apa saja yang menjadi masalah dalam aplikasi tersebut. Sehingga masyarakat bisa memilih untuk lebih memilih aplikasi mana yang lebih nyaman digunakan nantinya dengan masalah-masalah tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana gambaran umum data ulasan tentang aplikasi Wattpad dan Dreame berdasarkan situs Google Play?

2. Bagimana klasifikasi dan akurasi data ulasan pengguna aplikasi Wattpad dan Dreame dengan menggunakan metode *Support Vector Machine* (SVM) dan Asosiasi?
3. Informasi apa yang didapatkan dalam setiap klasifikasi yang telah dilakukan?
4. Faktor–faktor apa saja yang harus dilakukan untuk memperbaiki dari hasil ulasan negatif yang didapat?

1.3 Pembatasan Masalah

Batasan masalah yang ditentukan untuk menghindari perluasan pembahasan dalam penelitian adalah sebagai berikut:

1. Data yang akan diklasifikasi hanya data *review* pengguna Wattpad dan Dreame pada Google Play.
2. *Review* pada aplikasi yang akan diklasifikasi adalah yang berbahasa Inggris.
3. Pelabelan data menggunakan Vader Lexicon di Python.
4. Analisis yang digunakan untuk analisis sentimen adalah SVM dan Asosiasi.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui gambaran umum data ulasan tentang aplikasi Wattpad dan Dreame berdasarkan situs Google Play.
2. Mengetahui klasifikasi dan akurasi data ulasan pengguna aplikasi Wattpad dan aplikasi Dreame dengan menggunakan metode *Support Vector Machine* (SVM) dan Asosiasi.
3. Mendapatkan informasi penting dalam setiap klasifikasi.
4. Mendapatkan informasi faktor–faktor apa saja yang harus dilakukan untuk memperbaiki dari hasil ulasan negatif yang didapat.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Mengetahui gambaran umum data *review* tentang aplikasi Wattpad dan Dreame berdasarkan situs Google Play.
2. Klasifikasi *review* dari pengguna aplikasi dapat mempermudah pihak yang berkepentingan untuk dapat lebih cepat dan efisien mencari apa yang diperlukan.
3. Hasil klasifikasi sentimen dapat dijadikan sebagai bahan evaluasi pihak managemen untuk pengembangan pelayanan.
4. Hasil klasifikasi sentimen dapat dijadikan acuan bagi masyarakat yang akan mengunduh aplikasi untuk mempertimbangkan sebelum mengunduh aplikasi.

1.6 Sistematika Penulisan

Sistematika penulisan yang dipergunakan dalam penulisan tugas akhir ini dapat diuraikan sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini akan dibahas tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Berisi tentang uraian teoritis atau teori – teori yang mendasari pemecahan tentang masalah – masalah yang berhubungan dengan judul tugas akhir.

BAB III METODE PENELITIAN

Metode pengumpulan data, variabel penelitian, analisis data dan alur penelitian.

BAB IV HASIL DAN PEMBAHASAN

Berisi tentang hasil penelitian dan pembahasan dari permasalahan.

BAB V PENUTUP

Berisi tentang simpulan dan saran dari permasalahan.

BAB II

TINJAUAN PUSTAKA

Pada tinjauan pustaka ini dijelaskan mengenai teori-teori dasar dan literatur yang menjadi dasar dalam penyelesaian masalah pada penelitian ini. Berbagai sumber yang digunakan, baik berupa buku, artikel, jurnal digunakan untuk mendukung teori penyelesaian tugas akhir ini.

2.1 Wattpad

Wattpad pertama kali diluncurkan pada Desember 2006, hasil kolaborasi antara Allen Lau dan Ivan Yuen. Basis Wattpad sendiri berada di Toronto, Kanada. Wattpad sudah memiliki 15 juta pengguna dengan lebih dari 400 juta cerita. Visi dari Wattpad yaitu untuk menghibur dan menghubungkan dunia dengan sebuah cerita. Wattpad adalah "rumah" bagi lebih dari 65 juta orangorang yang menghabiskan lebih dari 15 miliar menit per bulan untuk membaca cerita di Wattpad. Wattpad menklaim bahwa 90% aktivitas penggunanya diakses melaui *mobile* serta mendukung lebih dari 50 bahasa. Hal ini membuktikan bahwa sebenarnya banyak orang memiliki ketertarikan tersendiri dalam hal menulis dan membaca. Sehingga dengan adanya Wattpad, memudahkan mereka untuk menyalurkan rasa kertetarikan tersebut. Terdapat berbagai *genre* bacaan yang bisa dinikmati seperti *triller*, *romance*, *teenfiction*, *fanfiction*, dan masih banyak lagi (Ulfa, 2018).

2.2 Dreame

Dreame merupakan aplikasi novel *online* semacam Wattpad.aplikasi ini diluncurkan pada Agustus 2018. Aplikasi Dreame sudah diunduh oleh 5 juta pengguna. Aplikasi ini hampir serupa dengan Wattpad hanya ada beberapa fitur tambahan seperti koin untuk membaca yang membuat cerita menjadi berbayar dan diberikan royalti bagi penulis, sehingga penulis di Dreame akan mendapatkan keuntungan dari hasil menulisnya.

2.3 Google Play

Google Play Store (dulunya bernama Android Market) merupakan toko aplikasi resmi milik Google untuk perangkat yang menggunakan sistem operasi Android. Saat ini dalam toko aplikasinya, Google Play Store memiliki aplikasi (*update* per tanggal 10 Desember 2017), sejumlah 3.532.448 aplikasi. Banyaknya aplikasi yang terdapat pada Google Play Store membuat toko aplikasi ini sangat menarik untuk dijadikan sebagai objek penelitian, khususnya dalam bidang *data mining*. Salah satu contohnya yaitu analisis sentimen yang merupakan cabang ilmu dari *data mining*, sangat cocok jika diaplikasikan pada Google Play Store. Analisis sentimen digunakan untuk menentukan sentimen para pengguna dari setiap aplikasi yang terdapat pada toko aplikasi tersebut dengan mengklasifikasikan ratusan, bahkan ribuan *text review* dari pengguna secara otomatis. Namun kendalanya saat ini adalah pihak Google sendiri tidak menyediakan API (*Application Programming Interface*) agar data pada Google Play Store dapat diintegrasikan dengan perangkat lunak yang sedang dikembangkan oleh *software developer* atau untuk keperluan sebagai data penelitian. Saat ini, pihak Google menyediakan API hanya untuk *developer* android yang aplikasinya terdaftar pada Google Play Store, API tersebut juga sangat terbatas, developer hanya bisa memanipulasi data-data tertentu dari aplikasi miliknya sendiri (Ilmawan, 2018).

2.4 *Online Review*

Review memiliki arti ulasan yang menurut Kamus Besar Bahasa Indonesia merupakan tanggapan, dan *online* atau daring (dalam jaringan) menurut KBBI merupakan terhubung melalui jejaring komputer, internet dan sebagainya.

Review mempunyai arti sebagai tinjauan atau ringkasan dari beberapa sumber baik buku, film, berita, dan lain-lain. *Review* difungsikan sebagai salah satu alat untuk meninjau kualitas suatu karya. Pardiyono (2007) menyatakan bahwa teks ulasan adalah teks yang berisi pemberian kritik, evaluasi, atau melakukan ulasan terhadap karya cipta intelektual. Teks ini bertujuan untuk memberikan kritikan,

hasil evaluasi, atas suatu karya ilmiah, buku, atau karya seni. Teks *review* adalah tulisan yang isinya menimbang atau menilai sebuah karya yang dikarang atau dicipta orang lain (Fikria, 2018).

Online review dapat diartikan sebagai komentar yang berisi kritikan, pujian dan masukan kepada platform, lembaga atau karya yang dilakukan melalui jaringan komputer atau internet.

2.5 Analisis Deskriptif

Analisis deskriptif adalah cara penggambaran suatu masalah berdasarkan kumpulan data dengan cara menyusun data tersebut sedemikian rupa sehingga dapat dipahami dengan mudah karakteristik datanya yang berguna untuk keperluan selanjutnya, dengan kata lain terdapat suatu proses pengumpulan data dan pengolahan data berdasarkan tujuan. Pengolahan data pada analisis deskriptif hanya berhubungan dengan menguraikan atau memberikan keterangan pada suatu data secara umum atau menggambarkan secara umum dari data yang didapatkan (Fikria, 2018).

Hasan (2004:185) menjelaskan: Analisis deskriptif adalah merupakan bentuk analisis data penelitian untuk menguji generalisasi hasil penelitian berdasarkan satu sample. Analisa deskriptif ini dilakukan dengan pengujian hipotesis deskriptif. Hasil analisisnya adalah apakah hipotesis penelitian dapat digeneralisasikan atau tidak. Jika hipotesis nol (H0) diterima, berarti hasil penelitian dapat digeneralisasikan. Analisis deskriptif ini menggunakan satu variabel atau lebih tapi bersifat mandiri, oleh karena itu analisis ini tidak berbentuk perbandingan atau hubungan (Nasution, 2017).

2.6 Data Mining dan Klasifikasi

Data Mining merupakan ilmu yang mempelajari metode untuk mengekstrak pengetahuan atau menemukan pola dari himpunan data yang banyak. Data mining adalah proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik-teknik, metode-metode, atau

algoritma dalam data mining sangat bervariasi. Pemilihan metode atau algoritma yang tepat sangat bergantung pada tujuan dan proses *Knowledge Discovery in Database (KDD)* secara keseluruhan (Mardi, 2017).

Menurut Han dan Kamber (2006) proses KDD terdiri dari langkah – langkah yang dinyatakan berikut:

1. *Data cleaning*

   Proses membersihkan data untuk menghilangkan *noise* dan data yang tidak konsisten.
2. *Data integration*

   Proses mengombinasikan data apabila memilki sumber data yang masih terpecah dalam sistem data mining.
3. *Data selection*

   Pengambilan data yang sesuai untuk digunakan dalam proses analisis *data mining*.
4. *Data transformation*

   Proses dimana data ditransformasikan menjadi bentuk-bentuk yang sesuai untuk proses dalam *data mining*.
5. *Data mining*

   Proses untuk mengekstrak kumpulan/ himpunan data agar menjadi ilmu pengetahuan.
6. *Pattern evaluation*

   Proses untuk mengidentifikasi kebenaran dari pola data yang mewakili pengetahuan berdasarkan beberapa tindakan.
7. *Knowledge representation*

   Proses untuk menyajikan presentasi teknik visual digunakan untuk menampilkan hasil pengetahuan kepada pengguna.

Machine learning adalah pembelajaran mesin yang sangat membantu dalam menyelesaikan masalah, membuat mudah dalam mengerjakan sesuatu (Telaumbanua, Hulu, Nadeak, Lumbantong, & Dharma, 2019). *Machine learning* merupakan ilmu yang mempelajari tentang algoritma yang digunakan oleh program komputer untuk memperoleh kesimpulan dari data yang sudah ada.

Macam – macam algoritma dalam *machine learning* sebagai berikut:

a. *Supervised learnig*
b. *Unsupervised learning*
c. *Semi – supervised learning*
d. *Active learning*

2.7 Analisis Sentimen

Menurut KBBI, sentimen adalah pendapat atau pandangan yang dibesarkan pada perasaan yang berlebih –lebihan pada sesuatu.

Analisis sentimen atau bisa di sebebut juga opinion mining merupakan proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam suatu kalimat apakah beopini positif atau negatif (Sudiantoro & Zuliarso, 2018).

Analisis sentimen merupakan analisis dengan menggunakan pendapat atau pandangan dari masyarakat untuk mengklasifikasikannya berdasarkan emosi, berada dikelas positif atau negatif.

2.8 *Text Mining*

Text mining adalah proses intensif pengetahuan yang berupaya mengekstaksi informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi pola yang menarik (Feldman & Sanger, 2007). Tugas khusus *text mining* antara lain, mengategorikan teks dan mengelompokkan teks (Putri & Setiadi, 2014).

Dalam melakukan *text mining*, teks perlu diolah terlebih dahulu untuk mempersiapkan data agar bisa dilakukan analisis yang lebih lanjut, tahapan ini disebut sebagai *text preprocessing.* Selain *text preprocessing*, ada beberapa tahapan dalam *text mining* menurut Feldman dan Sanger (2007):

2.8.1 Text Preprocessing

Proses awal dalam *text mining* untuk mengubah data mentah menjadi data yang terstruktur. Ada beberapa tahap dalam *preprocessing text mining* yang secara umum tahapanya sebagai berikut:

a. *Spelling Normalization*

Proses perbaikan dan mengganti kata yang salah eja atau singkatan menjadi kata yang sesuai dengan ejaan. Perbaikan ini bertujuan untuk menghindari jumlah perhitungan kata yang melebar. Jika kata tak sesuai dengan ejaan, maka pada proses selanjutnya kata yang sekiranya penting terbuang dan dianggap spam karena tak sesuai dengan ejaan yang sesuai.

b. *Case Folding*

Proses untuk mengubah huruf kapital menjadi huruf kecil dan menghapus karakter – karakter yang tidak diperlukan seperti angka, tanda baca dan hal lain yang bukan karakter huruf a – z.

c. *Tokenizing*

Proses memecah kalimat menjadi kata – kata yang bermakna untuk dilakukan analisis.

d. *Filtering*

Proses mengambil kata yang penting hasil dari *tokenizing* dan membuang kata – kata tak penting yang berada dalam *stopword*, seperti kata hubung dan imbuhan.

2.8.2 Feature Selection

Tahapan untuk mengurangi dimensi dari sebuah data tekstual sehingga hasil dari proses *Text Mining* memiliki kualitas yang lebih baik. Proses yang dilakukan pada tahapan ini adalah *stopword removal* yaitu menghilangkan kata-kata yang dianggap tidak penting atau tidak menggambarkan isi dari sebuah kalmiat (Indraloka & Santosa, 2017).

2.8.3 Text Representation

Proses merubah data tekstual menjadi representasi yang lebih mudah untuk diproses. Pada tahapan ini, sebuah kalimat direpresentasikan sebagai objek dan katakata yang menyusunnya direpresentasikan sebagai fitur. Data tekstual akan membentuk sebuah ruang dengan jumlah objek sebanyak

jumlah kalimat yang ada dan jumlah fitur sebanyak jumlah kata yang berbeda (Indraloka & Santosa, 2017).

2.9 *Term Frequency-Inverse Ducument Frequency (TF-IDF)*

Term Frequency (TF) adalah bobot dari suatu kata, t, dalam suatu dokumen, d dan dilambangkan dengan $TF_{t,d}$. Pendekatan paling sederhana dari konsep ini adalah dengan menyatakan bobot suatu kata t sebagai jumlah kemunculannya pada dokumen d (Andayani & Ryansyah, 2017).

TF dirumuskan sebagai berikut:

$$\boldsymbol{TF}_{\mathbf{t,d}} = \begin{cases} \mathbf{log\ f_{t,d}+1}, & \mathbf{f_{t,d}>0} \\ \mathbf{0}, & \mathbf{f_{t,d}=0} \end{cases} \qquad (2.1)$$

Di mana,

$TF_{t,d}$ = *term frequency*

$f_{t,d}$ = jumlah kemunculan kata t di dalam dokumen d

Konsep *inverse document frequency* (IDF) dibuat untuk mengurangi efek dari kata yang frekuensinya terlalu tinggi dalam kumpulan dokumen. Ide dasarnya adalah untuk menurunkan bobot dari kata dengan frekuensi kolektif (frekuensi total kemunculan kata di semua dokumen) yang tinggi (Andayani & Ryansyah, 2017).

IDF dirumuskan sebagai berikut:

$$\mathbf{IDF} = \mathbf{log}\ \frac{\boldsymbol{td}}{\boldsymbol{DF}} \qquad (2.2)$$

Di mana,

DF = *document frequency*, jumlah dokumen di mana kata tersebut mucul.

Term Frequency – Inverse Ducument Frequency (TF – IDF) adalah statistik numerik yang dimaksudkan untuk mencerminkan betapa pentingnya sebuah kata dalam dokumen koleksi atau *corpus* (Purbo, 2019).

Rumus TF – IDF dirumuskan sebagai berikut:

$$\mathbf{TF - IDF = TF.IDF} \qquad (2.3)$$

2.10 Algoritma Prediksi *Support Vector Machine*

Metode SVM merupakan metode yang menggunakan *hyperplane* untuk digunakan sebagai pemisah antara data secara linier, sehingga untuk mengatasi permasalahan data yang berupa data nonlinier, maka dapat digunakan teknik *kernel trick*. Metode *Support Vector Machine* (SVM) juga cukup baik dalam menyelesaikan permasalahan klasifikasi *multiclass*. Konsep *One Against All* adalah salah satu cara dalam mengatasi permasalahan *multiclass* (Perdana & Furqon, 2018).

2.10.1 SVM Linearly Separable Data

Linearly separable data merupakan data yang dapat dipisahkan secara linier. Misalkan $\{x_1,..., x_n\}$ adalah dataset dan $y_i \in \{+1,-1\}$ adalah label kelas dari data x_i. Anggap ada beberapa hyperplane yang memisahkan sampel positif dan negatif, maka x yang berada pada hyperplane akan memenuhi persamaan w.x + b=0. Untuk permasalahan data linier, algoritma *support vector* hanya mencari hyperplane dengan margin yang terbesar (jarak antara dua kelas pola) (Utami & Susyanto, 2012).

2.10.2 SVM Nonlinearly Separable Data

Untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier formula SVM harus dimodifikasi karena tidak akan ada solusi yang ditemukan. Oleh karena itu, kedua bidang pembatas harus diubah sehingga lebih fleksibel (untuk kondisi tertentu) dengan penambahan variabel ξ_i ($\xi_i \geq 0$, $\forall_i$: $\xi_i = 0$ jika x_i diklasifikasikan dengan benar) menjadi $x_i.w + b \geq 1 - \xi_i$ untuk kelas 1 dan $x_i.w + b \leq -1+\xi_i$ untuk kelas 2. Pencarian bidang pemisah terbaik dengan dengan penambahan variabel ξ_i sering juga disebut soft margin hyperplane (Sembiring, 2007).

2.11 Evaluasi Sistem Klasifikasi

Dalam klasifikasi, performa sistem perlu dilakukan evaluasi untuk mengukur akurasi dalam melakukan klasifikasi. Untuk melakukan evaluasi dalam klasifikasi, perlu dilakukan beberapa langkah sepeti berikut:

2.11.1 K-Fold Cross Validation

Cross validation adalah salah satu teknik untuk menvalidasi keakuratan sebuah model baru yang dibangun berdasarkan dataset baru. Data yang digunakan dalam pembuatan model baru disebut data latih dan data yang digunakan untuk menvalidasi adalah data tes. Model klasifikasi baru dibuat untuk memprediksi suatu dataset baru agar bisa diterapkan dimasa mendatang, sehingga diperlukan suatu validasi agar model yang dihasilkan mempunyai performa yang baik (Fikria, 2018).

K-Fold Cross Validation adalah metode validasi yang membagi data ke dalam subset, kemudian dilakukan perulangan (iterasi) pengujian sebanyak K. Pada setiap pengulangan, digunakan satu subset sebagai data uji dan subset lainnya sebagai data pembelajaran (Masripah, 2016).

2.11.2 Precision

Precision disebut juga nilai prediktif positif adalah potongan instance yang relevan untuk diambil (Purbo, 2019). Untuk mendapatkan nilai *Precision* dapat dihitung dengan persamaan berikut:

$$\boldsymbol{Precision} = \frac{\boldsymbol{Relevant\ item\ retrived}}{\boldsymbol{retrieved\ item}} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \qquad (2.4)$$

Di mana,

TP = *True Positive*

FP = *False Positive*

2.11.3 Recall

Recall yang dikenal juga sebagai sensitivitas merupakan potongan contoh revelan untuk diambil (Purbo, 2019). Persamaan untuk menghitung *Recall* adalah sebagai berikut:

$$\boldsymbol{Recall} = \frac{\boldsymbol{Relevant\ item\ retrived}}{\boldsymbol{retrieved\ item}} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \qquad (2.5)$$

Di mana,

FN = *False Negative*

2.11.4 F – Measure

F-measure adalah pengaruh *relativ*e hasil kombinasi nilai *precision* dengan nilai recall. F-measure dapat digunakan untuk mengukur kinerja dari sistem klasifikasi yang merupakan rata-rata harmonis dari *precision* dan *recall*. F-measure dapat memberikan penilaian yang lebih seimbang, F-measure dapat dihitung dengan persamaan berikut (Fikria, 2018):

$$\boldsymbol{F - Measure} = \frac{\mathbf{2}.(\boldsymbol{Recall}.\boldsymbol{Precision})}{(\boldsymbol{Recall+Precision})} \qquad (2.6)$$

2.11.5 Accuracy

Accuracy merupakan rata – rata aritmatika dari Presisi dan Inverse Presisi (oleh Bias) serta rata –rata aritmatika dari Recall dan Inverse Recall (Purbo, 2019). *Accuracy* dapat dihitung dengan rumus berikut:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Prediksi\ data\ benar}}{\boldsymbol{Total\ data}} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \qquad (2.7)$$

Di mana,

TN = *True Negative*

2.12 *Word Cloud*

Word cloud adalah visualisasi dari data teks, digunakan untuk menggambarkan metadata kata kunci (tag), yang biasanya merupakan berbentuk

kata tunggal, dengan warna dan ukuran setiap tag yang menunjukan kadar pentingnya suatau tag (Setyobudi, Alwi, & Astuti, 2018). *Word cloud* berisi kumpulan kata – kata yang membentuk seperti awan kata dengan tampilan visualisasi semakin besar kata di awan, maka kata tersebut frekuensi munculnya semakin banyak.

2.13 Asosiasi Kata

Association rules bertujuan untuk menentukan hubungan antar item dalam suatu dataset (sekumpulan data) yang telah ditentukan. Dengan menggunakan metode *Association rules* dapat mencari kombinasi yang paling sering terjadi dari suatu dataset. Terdapat tiga kriteria ukuran yaitu *support, confidence,* dan *lift* (Lazuardi, 2014).

Asiosiasi kata adalah mencari sebuah nilai dari hubungan suatu kata. Hasil asosiasi teks menunjukkan besarnya korelasi antar kata dan seberapa sering kata-kata tersebut muncul bersamaan dalam satu kalimat. Semakin sering kata tersebut muncul bersamaan dalam satu kalimat maka semakin besar pula nilai korelasi (Ihsan, Roza, & Widodo, 2019).

2.14 Diagram Sebab Akibat

Fungsi dasar diagram Fishbone (Tulang Ikan)/ *Cause and Effect* (Sebab dan Akibat)/ Ishikawa adalah untuk mengidentifikasi dan mengorganisasi penyebab – penyebab yang mungkin timbul dari suatu efek spesifik dan kemudian memisahkan akar penyebabnya (Heri Murnawan, 2014).

Diagram Fishbone merupakan diagram yang berbentuk seperti tulang ikan yang berisikan sebab dan akibat dari suatu masalah, diagram ini digunakan untuk mengidentifikasi faktor dari masalah – masalah yang ada, di mana setiap duri menggambarkan dari masalah.

2.15 Bahasa Python

Python adalah bahasa pemrograman dengan *general – purpose, high – level programming* atau bahasa pemrograman tingkat tinggi yang menitikberatkan pada

code readability, dan *syntax* yang memungkinkan *programmer* untuk mengekspresikan konsepnya dengan *lines of code* yang lebih sedikit daripada bahasa lainnya (Purbo, 2019).

Python menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman lain dan alat-alat bantu lainnya. Python hadir dengan pustaka-pustaka standar yang dapat diperluas serta dapat dipelajari hanya dalam beberapa hari (R, Rahmansyah, & Darwin, 2017).

Menurut Perkasa, Widyantara, & Susanto (2014), fitur yang dimiliki Python sebagai berikut:

a. Memiliki kepustakaan yang luas; dalam distribusi Python telah disediakan modul-modul.
b. Memiliki tata bahasa yang jernih dan mudah dipelajari.
c. Memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.
d. Berorientasi obyek.
e. Dapat dibangun dengan bahasa Python maupun C/C++.

BAB III

METODE PENELITIAN

4.1 Jenis dan Sumber Data Penelitian

Jenis data yang digunakan dalam penelitian ini adalah data primer. Data yang digunakan diperoleh dengan cara *web scraping* dari ulasan pengguna di situs web Google Play menggunakan *software* Python. Data yang digunakan, data ulasan dari pengguna aplikasi Wattpad dan Dreame di Google Play.

4.2 Metode Pengambilan Data

Metode yang digunakan untuk pengambilan data dari penelitian ini adalah dengan *web scraping* dari ulasan aplikasi Dreame dan Wattpad di Google Play yang berbahasa Inggris dan diperoleh data tanggal, pengguna dan ulasan dari kedua aplikasi tersebut.

4.3 Instrumen Pengambilan Data

Instrumen atau alat untuk pengambilan data ulasan tersebut adalah dengan menggunakan *software* Anaconda Navigator dengan bahasan pemrograman Python 3 dengan bantuan Chrome dan Web Driver serta didapatkan data berupa CSV yang dibaca dengan Microsoft Excel.

4.4 Variabel Penelitian

Variabel yang digunakan dalam penelitian ini adalah tanggal ulasan, nama pengguna (*user*), rating dan ulasan (*review*) dari pengunduh aplikasi Wattpad dan Dreame di Google Play.

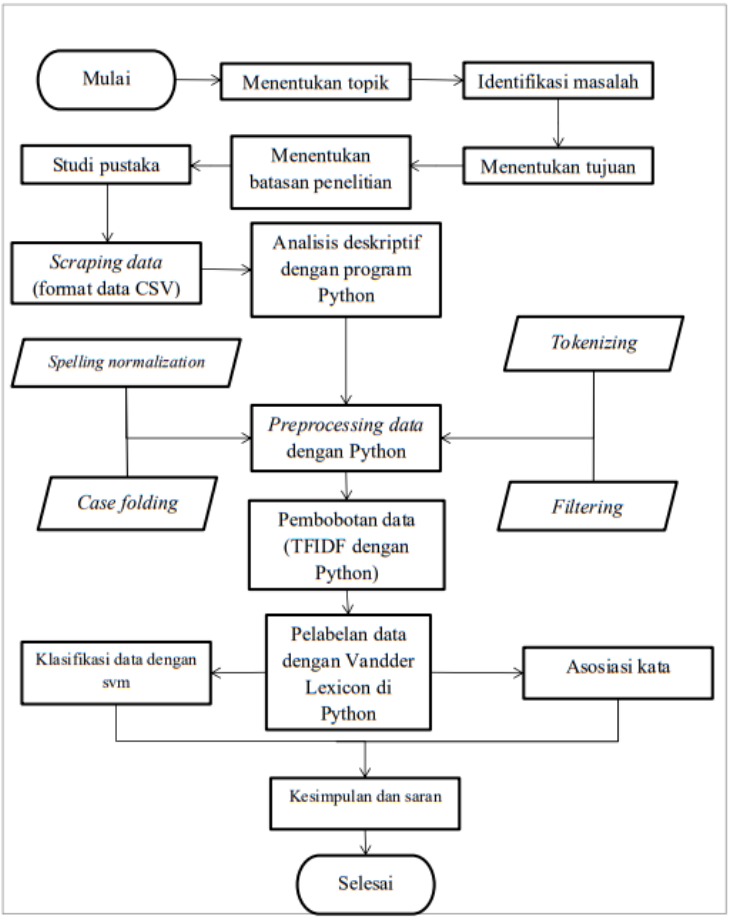
4.5 Teknik Analisis Data

Software yang digunakan dalam penelitian ini adalah Microsoft Excel 2010 dan Python dengan Anaconda Navigator. Metode analisis yang digunakan dalam penelitian ini yaitu:

a. Analisis deskriptif, digunakan untuk menggambarkan secara umum presepsi pengguna aplikasi Wattpad dan Dreame melalui *rating* yang diberikan oleh pengguna.
b. Metode *machine learning* yaitu *Support Vector Machine (SVM)* untuk melakukan klasifikasi *review* dalam bentuk kelompok positif dan negatif.
c. Asosiasi digunakan untuk mengidentifikasi dan membentuk pola kata yang berhubungan dengan kata lain yang berguna untuk mendapatkan informasi yang penting.

4.6 Alur Penelitian

Diagram alur tahapan penelitian ini divisualisasikan sebagai berikut:



Gambar 3. 1 Diagram Penelitian

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

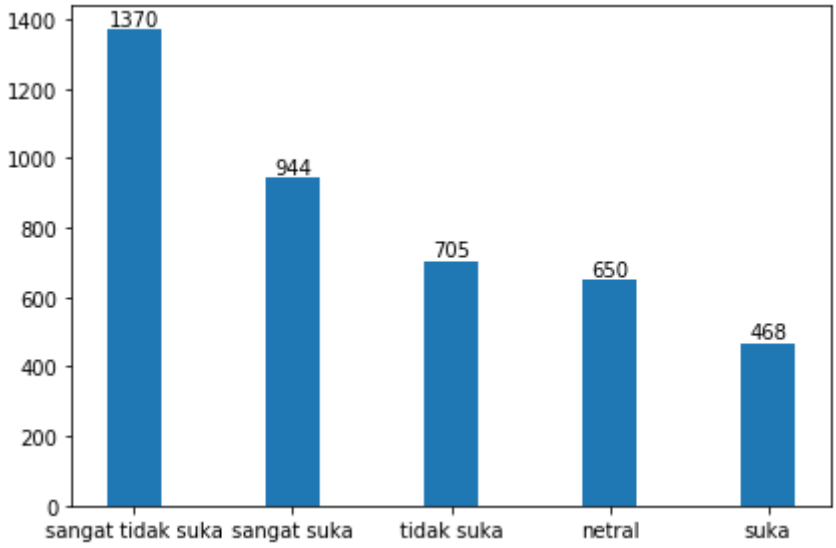
Hasil penelitian berdasarkan data ulasan di Google Play untuk dilakukan analisis sentimen dengan bantuan *software* Anaconda Navigator dengan bahasa pemrograman Python 3.

4.1.1 Scraping Data

Data *review* aplikasi Wattpad dan Dreame di Google Play diperoleh dengan menggunakan metode *scraping* menggunakan *Software* Anaconda Navigator menggunakan bahasa Python 3. Metode *scraping* ini merupakan metode untuk menangkap data pada suatu web dan dilakukan ekstraksi menjadi bentuk data dalam format CSV. Data yang diperoleh dari *scraping* web pada Google Play terdiri dari 4 varibel, yaitu tanggal ulasan, nama pengguna (*user*), *rating* dan *review*. Data yang diperoleh dari kedua aplikasi tersebut selanjutnya akan dilakukan analisis deskriptif atau gambaran secara umum berdasarkan *rating*.

4.1.2 Analisis Deskriptif

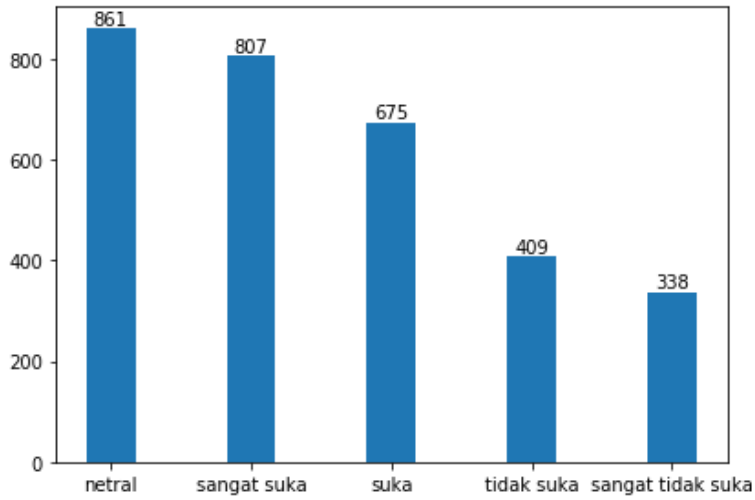
Analisis deskriptif merupakan gambaran umum dengan melihat *rating* dari ulasan data di Google Play. Pengelompokan data untuk analisis deskriptif ini menggunakan nilai *rating* 1 dianggap sangat tidak suka, 2 dianggap tidak suka, 3 dianggap netral, 4 dianggap suka, dan 5 dianggap sangat suka. Dalam data *review* aplikasi Wattpad didapatkan hasil analisis deskriptif ditampilkan dalam diagram.



Gambar 4.1 Diagram *Rating* Wattpad

Berdasarkan gambar di atas gambaran secara umum dengan menggunakan nilai *rating* didapatkan hasil dari 4137 data *review* terdapat 1370 pengguna yang sangat tidak suka, 705 tidak suka, 650 netral, 468 suka dan 944 sangat suka dengan aplikasi Wattpad. Jadi gambaran secara umum berdasarkan *rating* untuk aplikasi Wattpad lebih banyak yang tidak menyukai.

Dalam data *review* pada aplikasi Dreame didapatkan hasil analisis deskriptif yang divisualisasikan dengan diagram berikut.



Gambar 4.2 Diagram *Rating* Dreame

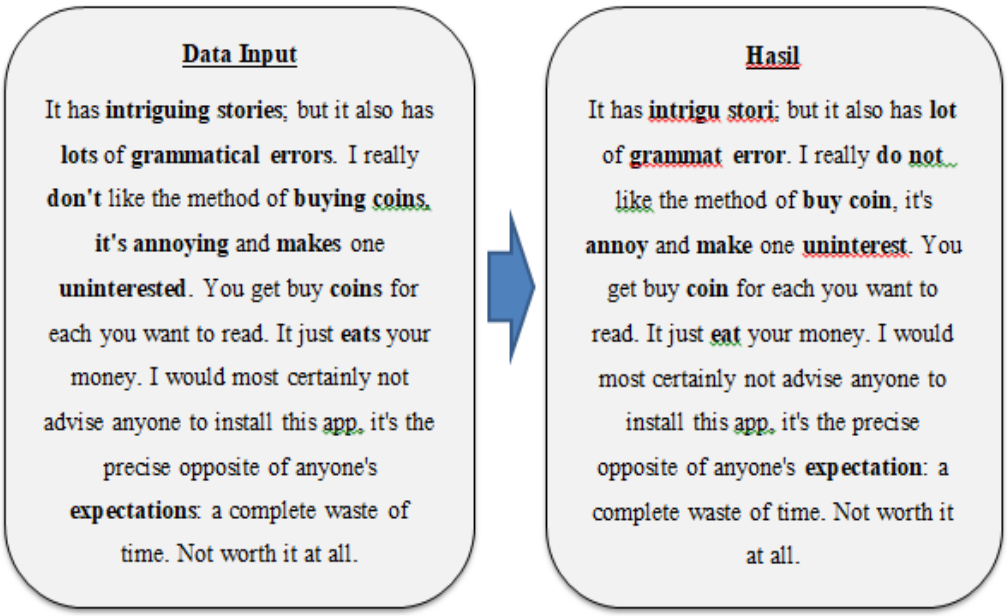
Berdasarkan gambar di atas gambaran secara umum dengan menggunakan nilai *rating* didapatkan hasil dari 3090 data *review* terdapat 338 pengguna yang sangat tidak suka, 409 tidak suka, 861 netral, 675 suka dan 807 sangat suka dengan aplikasi Dreame. Berdasarkan gambaran umum yang diperoleh dari *rating*, pengguna Dreame lebih banyak yang menyukai daripada tidak menyukai.

4.1.3 *Text Preprocessing*

Data *review* aplikasi Wattpad dan Dreame yang diperoleh dari Google Play didapatkan data teks yang strukturnya masih belum beraturan jika langsung dilakukan analisis, masih banyak tanda baca dan kata – kata yang tidak perlu, maka data tersebut harus dilakukan *preprocessing* menggunakan *text mining*. Untuk membersihkan data dengan *text mining* dilakukan tahapan proses diantaranya adalah *tokenization*, *case folding*, *spelling normalization* dan *filtering.*

4.1.3.1 *Spelling Normalization*

Dalam data *review* masih terdapat beberapa kata singkatan serta kata yang tidak baku, maka dengan ini dilakukan *spelling normalization* untuk memperbaiki atau subtitusi pada kata – kata yang kurang tepat menjadi kata yang sesuai.

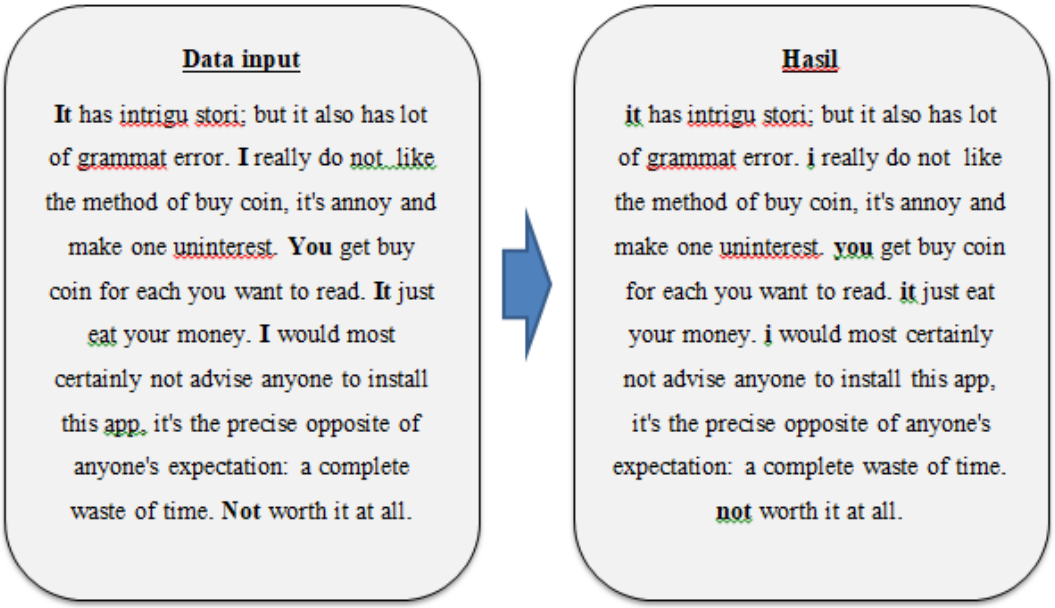


Gambar 4.3 Proses *Spelling Normalization*

Berdasarkan gambar 4.3 proses *spelling normalization* terlihat dari kata yang tercetak tebal dirubah menjadi kata dasar dan dibenarkan pengejaannya sesuai yang ada dalam program Python 3.

4.1.3.2 *Case Folding*

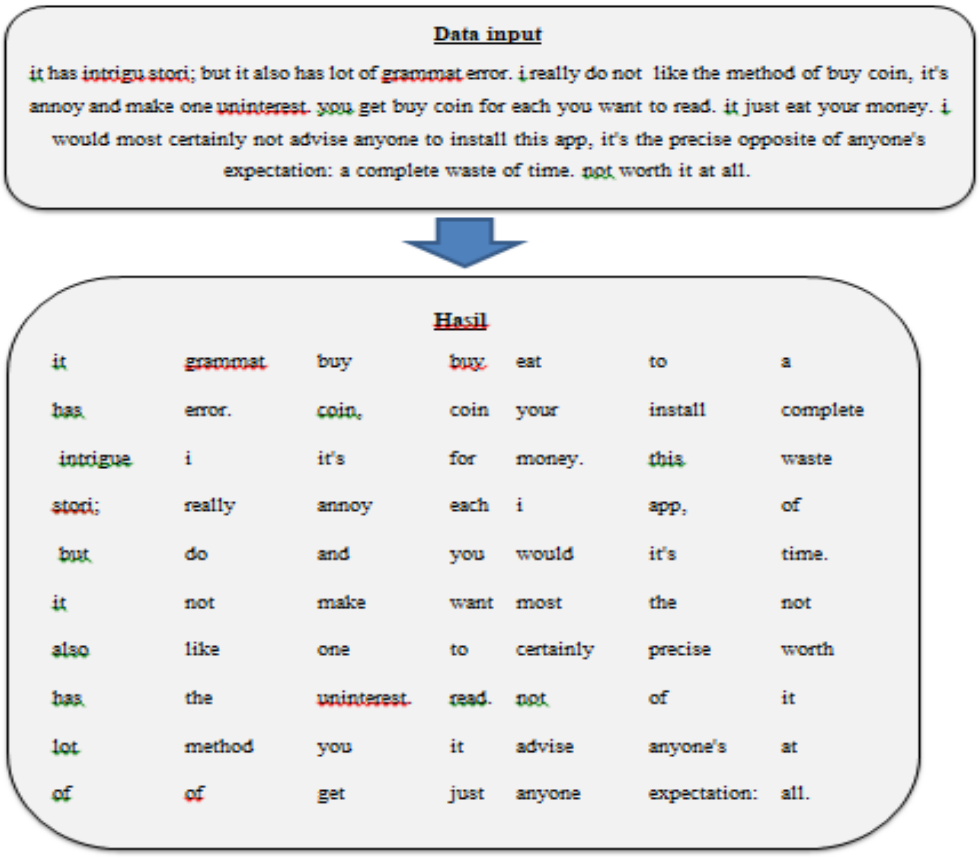
Proses selanjutnya dalam menyiapkan data adalah *case folding* yang berguna untuk merubah huruf kapital menjadi huruf kecil. Dalam pemprosesan ini agar dalam analisis sentimen agar semua huruf menjadi sama. Contohnya merubah huruf 'I' menjadi 'i' dalam data tekstual.



Gambar 4.4 Proses *Case Folding*

4.1.3.3 *Tokenizing*

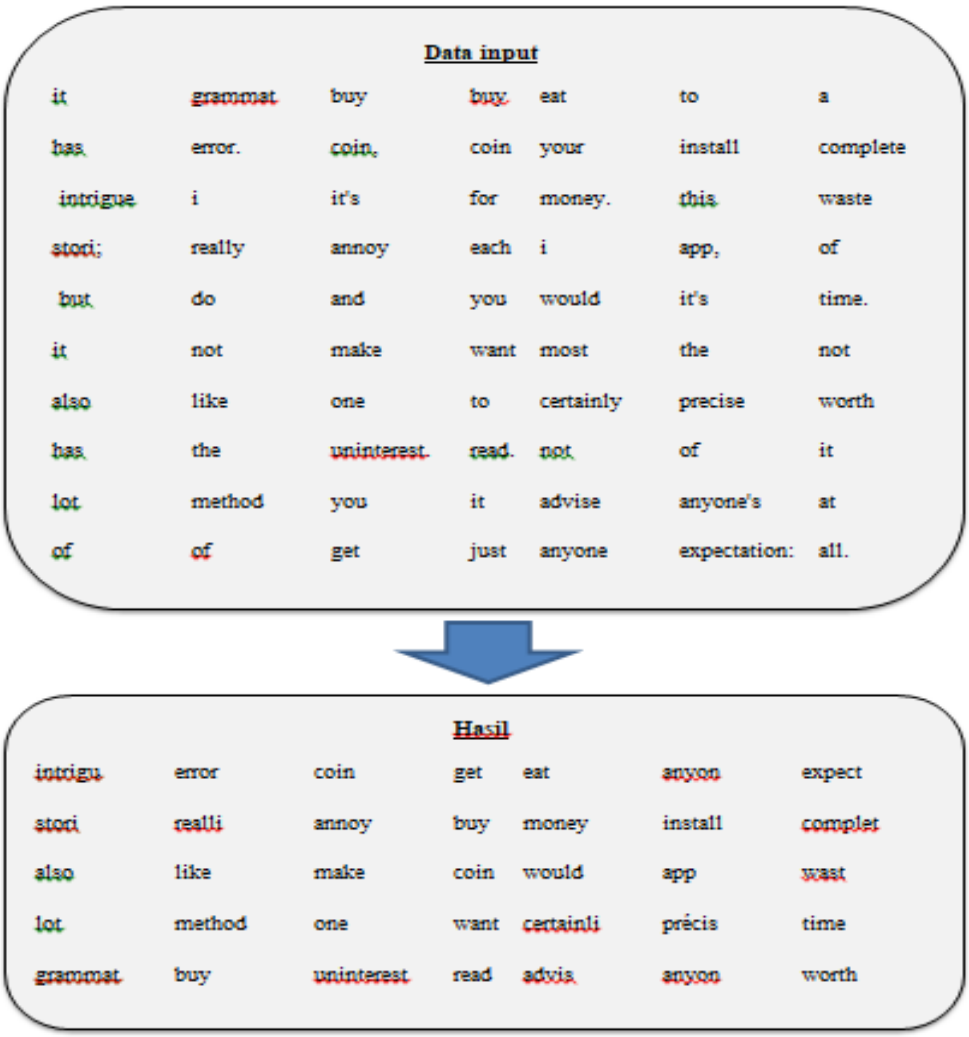
Proses *tokenizing* ini berfungsi untuk memecah kalimat menjadi daftar kata – kata agar menjadi lebih mudah dalam menentukan entitas dalam melakukan analisis. Data yang telah menjadi potongan kata – kata akan lebih mudah dilihat frekuensi kata terbanyak untuk divisualisasikan nantinya.



Gambar 4.5 Proses *Tokenizing*

4.1.3.4 *Filtering*

Proses *filtering* merupakan proses menghilangkan kata tak perlu, tanda baca dan angka – angka. Data setelah difilter akan menjadi data yang hanya berisi kata – kata yang bermakna saja. Dalam proses ini ada menghilangkan kata yang ada dalam *stopword* seperti kata hubung yang tidak penting untuk dilakukan analisis.



Gambar 4.6 Proses *Filtering*

4.1.4 Pelabelan Kelas Sentimen

Data yang telah dilakukan *preprocessing* selanjutnya akan dilakukan pelabelan menjadi kelas positif dan negatif sebelum dilakukan klasifikasi. Pelabelan untuk kelas sentimen dilakukan otomatis dengan menggunakan Vader Lexicon di Python. Setiap kata tersebut akan dihitung berapa nilai kata positif dan berapa nilai negatif. Dalam pelabelan Vader Lexicon ini menggunakan sistem *compound* atau menghitung nilai yang berkonotasi positif maupun negatif. *Compound* merupakan penggabungan dua kata yang memiliki arti sendiri, sehingga bobot nilai jika ada dua kata yang berkonotasi positif dan negatif, jika digabungkan akan menjadi kalimat negatif, maka *compound* tersebut akan memasukkan ke dalam kata negatif,

begitupun sebaliknya. Nilai dalam *compound* setiap kalimat dan kata memiliki nilai sendiri, tergantung bagaimana kalimatnya, jika lebih condong ke negatif, maka akan masuk dalam kelas negatif dan sebaliknya. Ulasan negatif diberi nilai minus dan kata positif diberi nilai plus. Jika kata tak terdeteksi dalam konotasi positif atau negatif, kata tersebut kan masuk dalam *compound* yang berkonotasi netral. Dari satu kalimat dihitung berapa kata positif dan berapa kata negatif lalu dilihat hasilnya, jika pada satu ulasan tersebut lebih banyak kata negatif maka akan masuk dalam kelas negatif dan bila lebih banyak kata positif maka akan masuk dalam kelas sentimen positif.

Pelabelan data untuk analisis sentimen menggunakan *package* NLTK di Python dengan modul Vader Lexicon yang berisi kamus kumpulan kata dengan nilainya untuk diklasifikasikan. Dengan Vader Lexicon setiap kata di data akan masuk menjadi tiga kelas, kelas positif, negatif dan netral sehingga data dapat diberi label menjadi positif atau negatif.

Tabel 4. 1 Perhitungan Skor Sentimen Vader Lexicon

Review	Skor Negatif	Skor Netral	Skor Positif	Total Skor	Hasil
intriguing stories also lots grammatical errors really dont like method buying coins annoying makes one uninterested get buy coins want read eats money would certainly advise anyone install precise opposite anyone expectations complete waste time worth	0,229	0,646	0,125	-0,6896	Negatif

Berdasarkan tabel 4.1 dapat diketahui, dengan perhitunag Vander Lexicon didapatkan kelas sentimen negatif pada data *review*, karena lebih tinggi nilai negatif daripada nilai positf. Untuk nilai negatif sebesar 0,229 sedangkan positif 0,125, sehingga kata negatif lebih dominan dan dapat disimpulkan bahwa *review* tersebut masuk dalam kelas negatif.

Berdasarkan perhitungan skor sentimen, maka dilakukan *array* pada data yang telah dilakukan *preprocessing* baik data *review* Dreame maupun Wattpad untuk mendapatkan label dengan pengaplikasian Vader Lexicon.

Tabel 4. 2 Hasil Pelabelan Kelas Sentimen Vader Lexicon

Sentimen	Wattpad	Dreame
Positif	2356	1865
Negatif	1058	761

Berdasarkan tabel 4.2 dapat diketahui bahwa kedua aplikasi memiliki lebih banyak ulasan positif daripada ulasan negatif. Untuk aplikasi Wattpad terdapat 2356 ulasan positif dan 1058 ulasan negatif, sedangkan untuk aplikasi Dreame terdapat 1865 ulasan positif dan 761 ulasan negatif.

4.1.5 Klasifikasi Sentimen

Data yang telah dilakukan pelabelan selanjutnya akan dilakukan prediksi klaifikasi dengan algoritma SVM. Sebelum dilakukan prediksi, data terlebih dahulu dipartisi menjadi data latih dan data uji dengan Python. Pembagian data latih dan data uji menjadi 90% data latih dan 10% data yang akan diujikan untuk diprediksi dengan algoritma SVM untuk masing-masing data *review* aplikasi Wattpad dan Dreame.

Tabel 4. 3 Partisi Data Latih dan Data Uji

Aplikasi	Data Latih (90%)	Data Uji (10%)	Jumlah
Wattpad	3072	342	3414
Dreame	2363	263	2626

Berdasarkan tabel 4.3, partisi data *review* aplikasi Wattpad untuk data latih di dapatkan 3072 dan data yang akan diuji sebanyak 342. Data *review* aplikasi Dreame didapatkan partisi data latih sebanyak 2363 dan data yang akan diuji sebanyak 263.

Setelah data dilakukan partisi, selanjutnya dilakukan prediksi klasifikasi dengan algoritma SVM. Algoritma SVM terdapat kernel Linear, RBF, Sigmoid dan

Polynomial. Dari keempat kernel tersebut akan dilihat akurasi yang tertinggi yang akan digunakan untuk mmemprediksi klasifikasi data.

Tabel 4. 4 Perbandingan Akurasi Kernel

Kernel	Wattpad	Dreame
Linear	88,30%	87,45%
RBF	88,59%	81,37%
Polynomial	74,27%	73,00%
Sigmoid	88,60%	86,69%

Berdasarkan tabel 4.4 dapat diketahui bahwa pada data *review* aplikasi Wattpad didapatkan nilai akurasi tertinggi menggunakan kernel Sigmoid dengan nilai akurasi sebesar 88,60%. Pada data *review* aplikasi Dreame didapatkan nilai akurasi tertinggi menggunakan metode kernel Linear dengan nilai akurasi sebesar 87,45%. Berdasarkan hasil perbandingan kernel tersebut, maka untuk data Wattpad menggunakan kernel Sigmoid dan data Dreame menggunakan kernel Linear.

Setelah didapatkan nilai akurasi terbaik, akan dicari nilai prediksi dengan *Cofusion Matrix* berdasarkan nilai akurasi terbaik dari model kernel. Untuk mencari *Cofusion Matrix* data Wattpad dengan menggunakan SVM model kernel Sigmoid dan pada aplikasi Dreame menggunaka *Cofusion Matrix* dengan SVM model kerel Linear.

Tabel 4. 5 *Cofusion Matrix*

Aplikasi	Aktual	Prediksi	
		Negatif	Positif
Wattpad	**Negatif**	73	23
	Positif	16	230
	Akurasi	**88,60%**	
Dreame	**Negatif**	49	24
	Positif	9	181
	Akurasi	**87,45%**	

Berdasarkan tabel 4.5 dapat diketahui bahwa, dari sebanyak 96 data ulasan negatif pada aplikasi Wattpad, 73 data terprediksi dengan benar masuk dalam klasifikasi negatif, sedangkan sebanyak 23 data salah klasifikasi menjadi kelas positif. Sebanyak 246 data ulasan positif pada aplikasi Wattpad, 230 data

terprediksi dengan benar terklasifikasi positif, sedangkan 16 data salah prediksi sehingga terklasifikasi menjadi negatif.

Data ulasan aplikasi Dreame, sebanyak 73 data ulasan negatif, 49 data terprediksi benar masuk dalam kelas negatif, sedangkan 24 data salah prediksi sehingga terprediksi menjadi kelas positif yang seharusnya masuk dalam kelas negatif. Sebanyak 190 data ulasan positif aplikasi Dreame, 181 terprediksi dengan benar masuk dalam kelas positif, sedangkan 9 data salah prediksi masuk dalam kelas negatif yang seharurnya masuk dalam kelas positif.

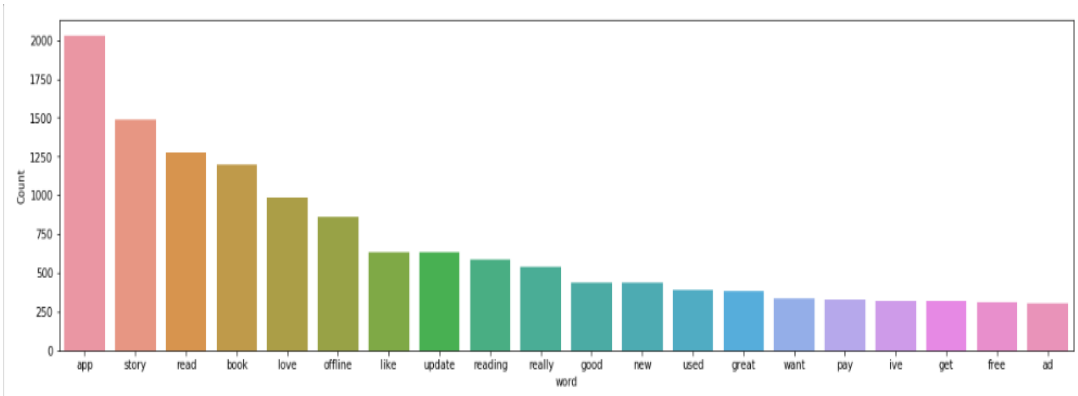
4.1.6 Visualisasi dan Asosiasi

Data *review* yang telah diklasifikasi selanjutnya akan divisualisasikan dan dicari asosiasi kata yang berhubungan untuk mengetahui bagian yang disukai dan tidak disukai pengguna dari aplikasi tersebut dan mencari sebab dari aplikasi tersebut tidak disukai agar mempermudah perusahaan untuk memperbaiki bagian tersebut. Visualisasi dengan mnggunakan *wordcloud* bertujuan untuk membandingan frekuensi kata yang berkomentar positif dan negatif. Dengan *wordcloud* dapat dilihat apa saja kata yang sering muncul dalam data positif dan negatif.

Asosiasi kata dari kata yang sering muncul tersebut, berfungsi untuk menangkap informasi secara lengkap dengan menangkap arti dari kata tersebut berdasarkan kata yang berasosiasi, sehingga mempermudah untuk mengidentifikasi masalah yang disampaikan pengguna melalui *review*.

4.1.6.1 Wattpad

Data *review* positif yang didapatkan dari data pelabelan kelas sentimen akan dicari frukuensi kata terbanyak untuk divisualisasikan dalam bentuk *wordcloud*. Dari 2356 *review* positif aplikasi Wattpad akan dicari kata – kata yang sering dibicarakan dari *review* tersebut, apa saja yang menjadi keunggulan dari aplikasi tersebut agar menjadi acuan masyarakat sebelum menguduh aplikasi Wattpad dan menjadi acuan platform agar mempertahankan performa dari aplikasi tersebut agar tidak menghilangkan hal-hal yang mendapat respon positif pengguna saat pembaruan aplikasi.



Gambar 4. 7 Frekuensi Kata Positif Wattpad

Berdasarkan gambar 4.7 dapat diketahui bahwa kata *app* merupakan kata yang paling sering dibicarakan dengan frekuensi 2000, diikuti kata *story* dengan frekuensi kata sebanyak 1500, diikuti kata *read, book, love, offline* dan seterusnya yang berhubungan dengan *review* positif. Selanjutnya akan ditampilkan visualisasi dalam bentuk *wordcloud* untuk melihat kumpulan kata-kata positif yang sering dibicarakan.

Gambar 4. 8 *Wordcloud Review* Positif Wattpad

Berdasarkan gambar 4.8 dapat diketahui bahwa dalam *wordcloud* kata *app* dan *story* tercetak paling besar, maka kata tersebut yang sering dibicarakan dalam *review* positif pengguna. Kata-kata yang terdapat dalam *wordcloud* tersebut

merupakan kata yang dibicarakan pengguna dalam *review* positif, pengguna tampak menyukai cerita yang ada dalam aplikasi Wattpad.

Tabel 4. 6 Asosiasi Kata Positif Wattpad

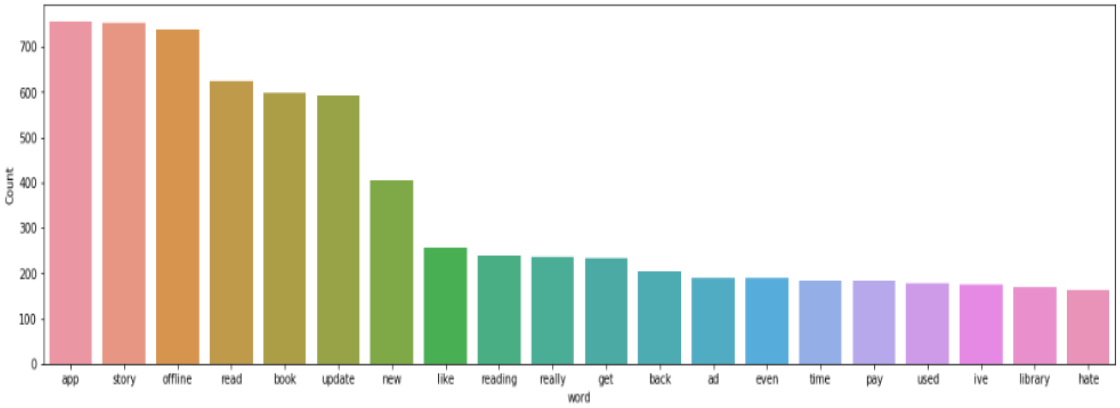
Offline		Read		Pay		Offline	
Customer	0,45	Offline	0,41	Read	0,24	Read	0,41
Dissatisfied	0,45	Books	0,28	Free	0,22	Stories	0,28
Hurt	0,26	Want	0,25	Used	0,22	Books	0,27
Afraid	0,26	Pay	0,24	Money	0,22	Update	0,27
Excited	0,26	Able	0,22			Two	0,27
Describe	0,22	Stories	0,22			New	0,25
Accept	0,22					Premium	0,24
Planning	0,22					Unlimited	0,21
Restored	0,22					Able	0,20
Upgrade	0,22					Wattpad	0,20
Band	0,22						
Continuosly	0,22						
Incomparable	0,22						
Assume	0,22						
Reed	0,22						
Stated	0,22						
Choosed	0,22						

Berdasarkan tabel 4.6 dapat diketahui, kata *application* berasosiasi dengan beberapa kata yang diperoleh informasi bahwa aplikasi Wattpad tidak memuaskan tapi pengguna merasa senang dengan adanya aplikasi tersebut bisa pulih dengan adanya *upgrade* sehingga aplikasi wattpad menjadi pilihan bagi pengguna. Kata *read* berasosiasi dengan kata *offline, books, want, pay, able* dan *stories*, di mana didapatkan informasi bahwa, di aplikasi Wattpad dapat membaca buku cerita secara *offline* yang diinginkan tanpa membayar, hal ini membuat aplikasi Wattpad disukai oleh penggunanya.

Kata *pay* berdasarkan tabel asosiasi diperoleh informasi bahwa pembayaran untuk membaca di aplikasi Wattpad bisa menggunakan uang dan bisa juga diakses gratis, sehingga ini menjadi kelebihan informasi, ada cerita berbayar dan ada pula yang gratis. Kata *offline* berdasarkan tabel asosiasi, diperoleh informasi bahwa dengan *offline* pengguna tetap dapat membaca cerita dan dengan adanya pembaruan

aplikasi menjadi dua cerita yang dapat dibaca secara *offline* tapi dengan pembaruan ke premium bisa o*ffline* tanpa ada batasan.

Data *review* negatif dari aplikasi Wattpad juga akan dilakukan visualisasi dan asosiasi seperti halnya data *review* positif dan diberikan tambahan untuk diagram sebab – akibat untuk mencari solusi dari masalah yang dikeluhkan oleh pengguna.



Gambar 4. 9 Frekuensi Kata Negatif Wattpad

Berdasarkan gambar 4.9 dapat diketahui bahwa kata *app* dan *story* merupakan kata yang paling sering dibicarakan dengan frekuensi lebih dari 700, diikuti kata *offline* dengan frekuensi kata sebanyak 700, diikuti kata *read*, *book*, *update* dan seterusnya yang berhubungan dengan *review* negatif. Selanjutnya akan ditampilkan visualisasi dalam bentuk *wordcloud* untuk melihat kumpulan kata – kata negatif yang sering dibicarakan.

Gambar 4. 10 *Wordcloud Review* Negatif Wattpad

Berdasarkan gambar 4.10 dapat diketahui kata yang sering dibicarakan yang divisualisasikan melalui *wordcloud* adalah kata *app* dan *story* yang dapat dilihat tulisan kata tersebut meiliki ukuran terbesar, lalu diikuti kata *book, reading, new update* yang mewakili komentar – komentar negatif.

Tabel 4. 7 Asosiasi Kata Negatif Wattpad

Story	New	Pay	Time
Kick 0,25 App 0,25 Connections 0,25 Offline 0,25 Pay 0,25 Update 0,25 Entirely 0,25 Shuts 0,25 Apart 0,25 Reconnected 0,25 Shuts 0,25 Usually 0,21	Update 0,42 Memory 0,23 Relogin 0,23	Unless 0,26 Books 0,23 Read 0,22 Stupid 0,22 Basic 0,22 Money 0,21 Premium 0,20	Long 0,33 Failed 0,22 Asks 0,22 Offline 0,22 Ended 0,20
Library	**Update**	**Application**	**Book**
Instantly 0,26 Load 0,26	New 0,44 Offline 0,21	Announced 0,41 Written 0,41	Resume 0,27

Whereas 0,26 Fulfilled 0,26 Books 0,22		Accommodate 0,41 Afraid 0,41 Anytime 0,41 Clash 0,41 Corrupted 0,41 Payable 0,41 Comments 0,41 Strangely 0,41 Via 0,41 Earth 0,41 Like 0,41 Follower 0,29 Includes 0,29 Posted 0,29 Register 0,29 Specific 0,29 Error 0,29 Android 0,29 Laptop 0,26 Choice 0,23 Content 0,23 Unlike 0,23 Votes 0,23 Web 0,23 Failed 0,20 Comes 0,20 Usually 0,20	Awkward 0,23 Uneven 0,23 Broken 0,23 Library 0,23 Sort 0,21 Minute 0,20

Berdasarkan tabel 4.7 dapat diketahui bahwa kata *story* berasosiasi dengan kata *kick, app, connection, offline, pay* dan lainnya. Dari tabel asosiasi kata diperoleh informasi bahwa cerita dalam aplikasi Wattpad banyak dikeluhkan pengguna karena butuh koneksi internet dan tidak bisa dibaca secara *offline*. Untuk membaca cerita secara *offline* harus membayar. Untuk cerita juga butuh koneksi ulang dan pengguna merasa tak nyaman dengan cerita yang selalu tertutup banyak iklan.

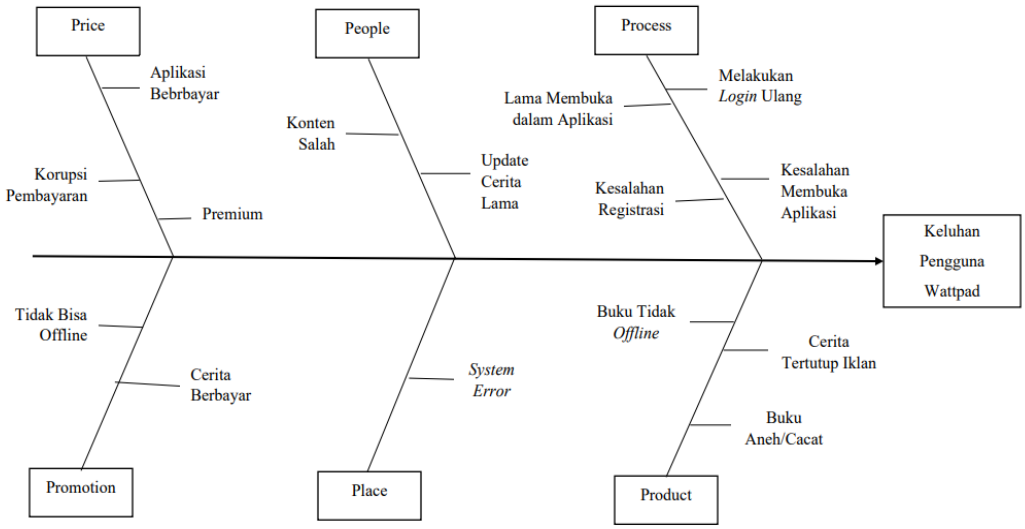
Kata *new* berdasarkan tabel asosiasi diperoleh informasi bahwa, pembaruan aplikasi versi terbaru membutuhkan banyak memori dan harus melakukan *login* ulang setelah adanya pembaruan aplikasi. Berdasarkan tabel asosiasi, kata *pay* yang berasosiasi dengan beberapa kata, dapat diketahui informasi, bahwa pengguna tidak suka aplikasi Wattpad karena pembayaran yang diperlukan untuk membaca buku

dan uang diperlukan untuk *upgrade* dari versi dasar menjadi versi premium. Dengan ini, adanya pembayaran tidak disukai oleh pengguna.

Berdasarkan asosiasi kata yang diperoleh dari kata *time* ada beberapa informasi yang diperoleh, waktu yang diperlukan pengguna untuk membuka aplikasi terlalu lama karena sering terjadi kesalahan untuk membuka buku secara *offline* dan aplikasi meminta berhenti setelah membutuhkan waktu yang lama. Kata *librabry* yang diperoleh asosiasi kata, dapat diketahui informasi bahwa perpustakaan tidak bisa digunakan untuk membuka buku secara instan, harus memuat beberapa buku sehingga sulit untuk mengingat buku mana saja yang telah dibaca dan belum dibaca dan kapasitas perpustakaan tidak mencukupi.

Berdasarkan kata *update* yang ada dalam tabel asosiasi, didapatkan informasi bahwa, pembaruan yang terbaru tidak bisa digunakan secara *offline*, sehingga pengguna merasa tak suka menggunakan aplikasi Wattpad. Kata *application* berdasarkan asosiasi kata, diperoleh informasi aplikasi Wattpad tidak disukai pengguna karena pemberitahuan tertulis tidak muncul juga akomodasi yang diberikan membuat pengguna khawatir dan dapat menimbulkan komentar yang berisi pertikaian kapan saja. Pengguna juga tak suka aplikasi ini karena menjadi aplikasi berbayar dan menyebut aplikasi melakukan korupsi karena dengan adanya sistem berbayar pada aplikasi yang dinilai cukup aneh. Pada aplikasi juga terjadi kesalahan saat registrasi baik menggunakan android maupun laptop. Pengguna mengeluhkan aplikasi ini selalu gagal saat melakukan *vote* konten melalui web. Aplikasi dinilai selalu terjadi kesalahan dalam penyeleksi konten sehingga dinilai buruk oleh pengguna.

Kata *book* dalam asosiasi didapatkan beberapa informasi bahwa ringkasan buku begitu aneh dan ganjil. Selain itu, buku pada perpustakaan banyak jenis buku yang dianggap cacat, sehingga membutuhkan waktu beberapa menit untuk memahami isi buku.



Gambar 4. 11 Diagram *Fishbone* Keluhan Pengguna Wattpad

Berdasarkan gambar 4.11 dapat diketahui faktor – faktor penyebab keluhan pengguna Wattpad berdasarkan *review* negatif pengguna. Setelah diketahui faktor – faktor penyebab pegguna tidak menyukai aplikasi Wattpad, akan dibuat tabel pemecahan masalah dari keluhan pengguna terhadap aplikasi Wattpad.

Tabel 4. 8 Pemecahan Masalah Aplikasi Wattpad

No	Faktor	Masalah	Pemecahan Masalah
1	*Process*	Harus melakukan *login* ulang setelah pembaruan aplikasi	Pihak developer harus memperbaiki sistem agar setelah melakukan pembaruan aplikasi, pengguna tidak perlu melakukan *login* kembali
		Membutuhkan waktu lama dalam membukka aplikasi	Pihak developer harus melakukan *update* sistem agar waktu untuk membuka aplikasi tidak terlalu lama *loading* dan berusaha untuk memperkecil ukuran aplikasi agar tidak terlalu berat
		Terjadi kesalahan/gagal	Developer harus melakukan *upgrade database* agar kapasitas

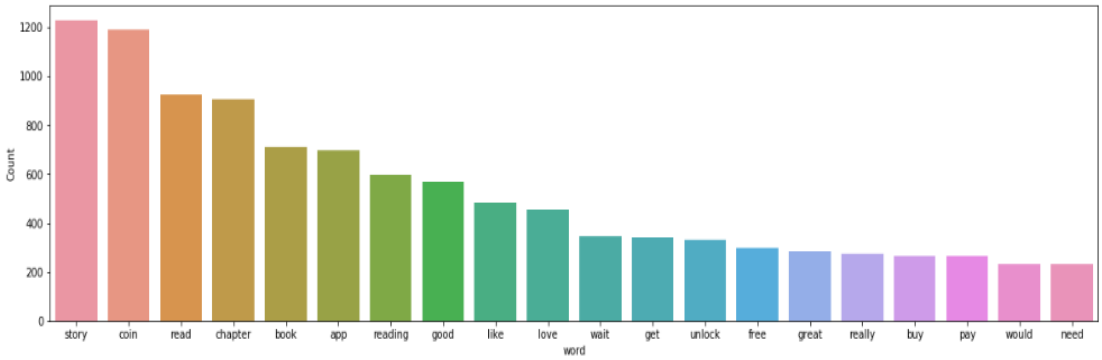
		dalam membuka aplikasi	pengguna dalam melakukan kegiatan di aplikasi tidak terjadi kegagalan karena sistem pada *database* tidak sibuk
		Terjadi kesalahan saat melakukan registrasi	Pihak developer harus memperbaiki sistem *database* bagi pengguna agar mampu memuat penambahan pengguna agar bisa melakukan registrasi untuk pengguna baru
2	*Product*	Buku tidak bisa dibaca secara *offline*	Pihak developer seharusnya mengembalikan sistem versi lama yang bisa membaca buku secara *offline*, karena pengguna lebih nyaman membaca secara *offline*
		Cerita pada aplikasi tertutup iklan	Pihak developer harus memperbaiki sistem pada aplikasi agar iklan tidak menutupi bagian cerita, iklan bisa diberikan tempat tersendiri agar tidak mengganggu kenyamanan dalam membaca
		Buku aneh/cacat untuk dibaca	Pihak developer harus memberikan filter pada aplikasi, buku mana yang layak untuk ditampilkan, sehingga perlu adanya sistem untuk peyeleksian cerita dengan kriteria tertentu
3	*People*	Konten pada cerita salah	Pihak penulis cerita harusnya bisa menyesuaikan konten cerita dengan benar sesuai isi cerita, pihak developer juga harus melakukan penyeleksian secara sistem dengan kesesuaian isi cerita dengan *genre*

		Update cerita lama	Pihak developer bisa membuat sistem dengan batas waktu untuk melakukan *update* cerita agar tidak terlalu lama kecuali cerita memang telah selesai, developer bisa melakukan pembatalan publikasi cerita jika cerita tidak *update* dalam jagka waktu tertentu
4	*Place*	*System error*	Sistem eror terjadi karena aplikasi lama dan aplikasi terlalu berat dan membutuhkan banyak ruang dalam ponsel ataupun laptop, pengguna sebaiknya menyiapkan lebih banyak ruang untuk aplikasi dan berusaha mengecilkan ukuran aplikasi dengan hapus data, pihak developer sebisa mungkin membuat aplikasi menjadi ringan dengan penghapusan data-data yang tidak perlu agar tidak terjadi banyak eror
5	*Price*	Aplikasi berbayar	Pihak developer memberikan peringatan pada pengguna bahwa memang ada aplikasi dengan mode berbayar khusus untuk premium, sedangkan untuk versi dasar aplikasi masih bisa dinikmati secara gratis
		Korupsi pembayaran aplikasi	Pihak developer lebih menjelaskan pembayaran digunakan untuk memberikan suasana berbeda bagi pengguna yang premium dengan tidak adanya iklan, sehingga pihak developer

			juga cerita berbayar akan masuk pada penulis sebagai royalti telah menghibur pembaca
		Perlu *upgrade* aplikasi premium	Developer lebih menekankan bahwa tidak semua harus dilakukan *upgrade* ke dalam versi premium, versi premium digunakan untuk pengguna yang ingin tidak ada iklan saat membuka aplikasi
6	*Promotion*	Tidak bisa membaca cerita secara *offline*	Pihak developer harus mengembalikan sistem seperti semula agar seuai dengan iklan yang dilakukan, membaca secara *offline* dan gratis yang menjadi keunggulan dari aplikasi
		Ada beberapa cerita berbayar	Pihak developer harus konsisten agar pengguna bisa membaca secara gratis yang sudah menjadi keunggulan aplikasi bagi para pengguna

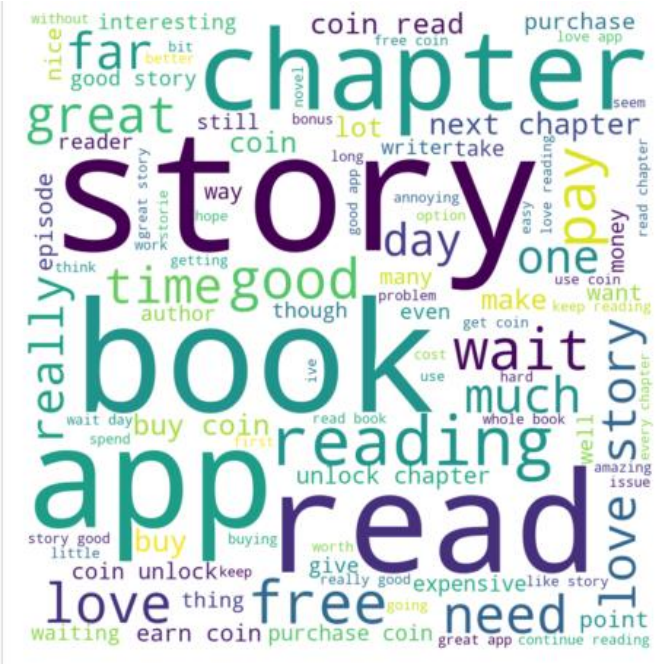
4.1.6.2 Dreame

Hasil pelabelan positif dari data *review* aplikasi Dreame didapatkan sebanyak 1865 data yang akan dilakukan visualisasi dengan melihat frekuensi kata terbanyak yang dibicarakan. Dengan visualisasi, data frekuensi kata akan diperoleh hal – hal yang disukai oleh pengguna.



Gambar 4. 12 Frekuensi Kata Positif Dreame

Berdasarkan tabel 4.12 dapat diketahui bahwa kata *story* memiliki frekuensi terbanyak dari data *review* positif aplikasi Dreame dengan frekuensi lebih dari 1200, diikuti kata *coin* dengan frekuensi sebanyak 1200. Selanjutnya visualisasi akan dibentuk menjadi *wordcloud* yang berisi kumpulan kata yang sering dibicarakan oleh pengguna aplikasi Dreame.



Gambar 4. 13 *Wordcloud Review* Positif Dreame

Berdasarkan gambar 4.13 dapat diketahui bahwa kata *story, book, app* dan *read* merupakan kata yang sering dibicarakan karena pada *wordcloud*, kata – kata tersebut tercetak paling besar dibanding kata – kata lain. Selanjutnya dari kata – kata yang sering mucul tersebut, akan dicari asosiasi kata untuk memperoleh

informasi – informasi apa saja yang membuat aplikasi Dreame dinilai baik oleh penggunanya.

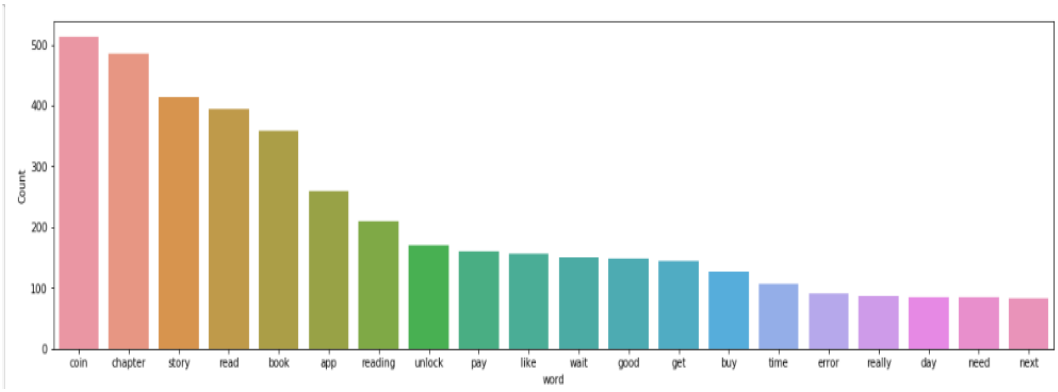
Tabel 4. 9 Asosiasi Kata Positif Dreame

Coin		**Chapter**		**Read**	
System	0,39	Next	0,33	Want	0,22
Complete	0,29	Days	0,23	Book	0,20
		Wait	0,21		
Book		**Buy**		**Wait**	
Finish	0,25	Coin	0,25	Days	0,43
Whole	0,25			Next	0,25
Price	0,20			Long	0,23
Read	0,20			Chapter	0,21

Berdasarkan tabel 4.9, asosiasi kata psitif dari *coin* diperoleh informasi bahwa aplikasi Dreame menggunakan koin dan bisa dibuka secara lengkap dengan koin. Kata *chapter* berdasarkan tabel asosiasi, diperoleh informasi untuk membuka *chapter* terbaru, pengguna akan menunggu hari berikutnya.

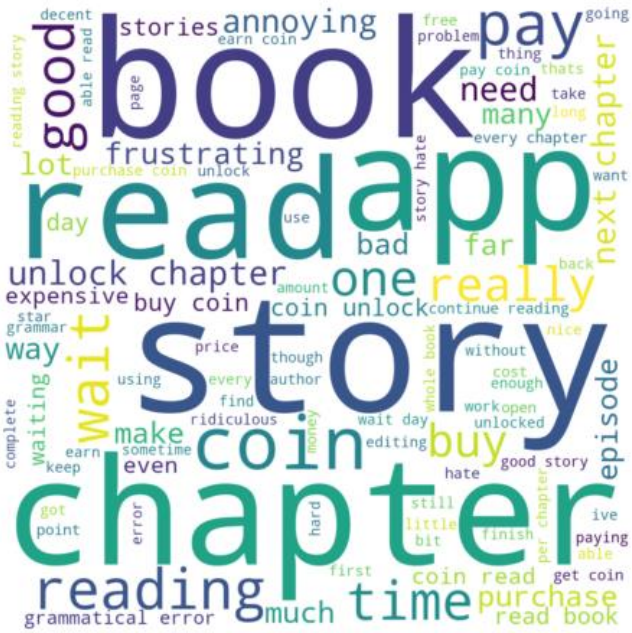
Berdasarkan kata *read* dari tabel asosiasi diperoleh informasi bahwa, pengguna aplikasi menyukai aplikasi Dreame karena ingin membaca buku di aplikasi tersebut. Kata *book* dapat diperoleh informasi berdasarkan asosiasinya yaitu, pengguna menyukai membaca buku di Dreame yang sudah selesai dan dapat membaca isi keseluruhan buku dengan harga yang murah. Kata *buy* yang berasosiasi dengan kata *coins* dapat diketahui informasi yang diperoleh, bahwa aplikasi Dreame dapat digunakan untuk membeli koin untuk membaca buku tanpa menunggu. Kata *wait* dapat diperoleh informasi berdasarkan asosiasi kata positifnya yaitu, pengguna menunggu hari selanjutnya untuk dapat membuka *chapter* baru.

Data *review* negatif dari aplikasi Dreame akan dilakukan visualisasi dengan diagram frekuensi dan *wordcloud* serta akan diasosiasikan seperti halnya data *review* positif dan diberikan tambahan untuk diagram sebab – akibat untuk mencari solusi dari masalah yang dikeluhkan oleh pengguna.



Gambar 4. 14 Frekuensi Kata Negatif Dreame

Berdasarkan gambar 4.14 dapat diketahui dari komentar negatif terhadap aplikasi Dreame kata yang sering dibicarakan adalah kata *coin* dengan frekuensi kata lebih dari 500 diikuti kata *chapter* dengan frekuensi 500 kemudian diikuti kata *story*, *read*, *book* dan kata – kata lain.



Gambar 4. 15 *Wordcloud Review* Negatif Dreame

Berdasarkan gambar 4.15 didapatkan informasi dari kumpulan kata – kata negatif, bahwa kata negatif yang sering dibicarakan adalah *chapter*, *book* dan *story*, kata – kata tersebut tercetak paling besar di *wordcloud*, sehingga ulasan negatif pengguna sering mengeluhkan tentang hal – hal tersebut. Dari kata yang sering

muncul, akan dilakukan asosiasi untuk mengetahui masalah – masalah yang dikeluhkan oleh pengguna Dreame.

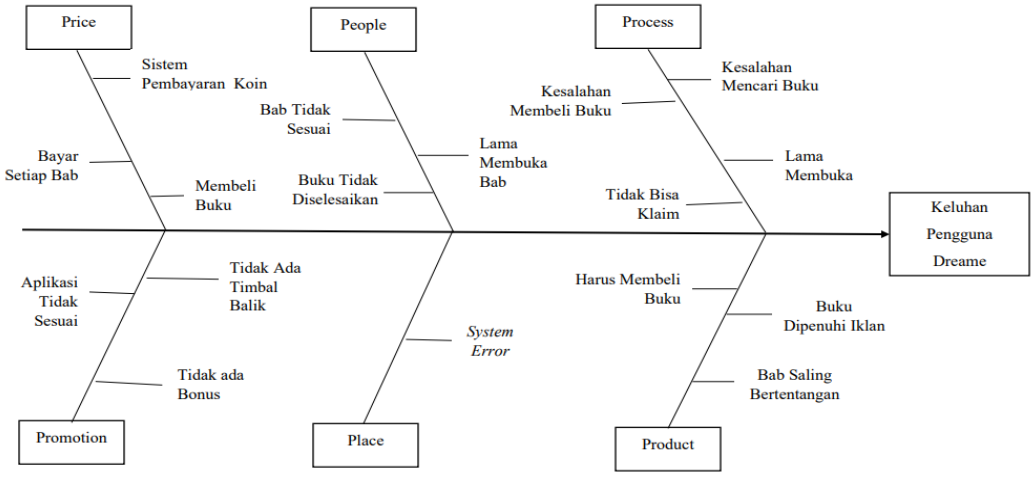
Tabel 4. 10 Asosiasi Kata Negatif Dreame

Coin		Chapter		Book		Buy	
System	0,33	Next	0,32	Buy	0,30	Book	0,30
Reach	0,33	Every	0,27	Amazon	0,29	Coins	0,24
Media	0,25	Per	0,24	Glitch	0,27		
Future	0,24	Pay	0,21	Whole	0,23		
Knowing	0,24			Finish	0,22		
Social	0,24			Advertised	0,22		
Consists	0,24			Bugs	0,22		
Rated	0,24			Loose	0,22		
Ruined	0,24			Periodically	0,22		
Invested	0,24			Search	0,22		
Odd	0,23			Claim	0,22		
Weird	0,21						
Wait		**App**		**Day**		**Next**	
Days	0,44	Giving	0,26	Reach	0,41	Chapter	0.32
Long	0,31	Library	0,24	Accompany	0,38	Ran	0.21
Hours	0,27	Even	0,23	Bundle	0,38	Incompatible	
Forget	0,25	Feedback	0,22	Seven	0,38	0,20	
Correct	0,23	Snippets	0,21	Signing	0,38	Message	
Smaller	0,23	Timeline	0,21	Throw	0,38	0,20	
		Voracious	0,21	Monthly	0,31	Task	
		Bonus	0,21	Subscription	0,31	0,20	
		Closing	0,21	Response	0,27	Awhile	
		Execute	0,21	Basically	0,27	0,20	
		Night	0,21	Receive	0,27	Bools	0,20
		Operation	0,21	Trial	0,27	Close	
		Recently	0,21	Dreame	0,23	0,20	
		Let	0,21	Left	0,22	Freeze	0,20
				Supposed	0,22	Happen	0,20
						Part	0,20
						Reopen	0,20
						Extensively	
						0,20	
						Decide	0,20
						Favorite	0,20
						Loved	0,20
						Worrying	0,20

Berdasarkan tabel 4.10 didapatkan informasi dari asosiasi kata negatif. Dari kata *coin* dapat diperoleh informasi bahwa pengguna tidak menyukai sistem koin

karena untuk mendapatkan koin dirasa aneh dan tidak konsisten hal ini dapat merusak investasi koin yang diperoleh. Kata *chapter* dapat diperoleh informasi dari asosiasi kata negatifnya, bahwa pengguna tidak menyukai karena untuk melanjutkan *chapter* selanjutnya harus membayar per *chapter*. Kata *book* diperoleh beberapa asosiasi kata dan didapatkan informasi, pengguna tidak menyukai buku karena harus membeli terlebih dahulu dan lebih memilih membeli buku di Amazon. Terjadi kesalahan dalam membeli buku dan buku dipenuhi oleh iklan. Terjadi pula kesalahan dalam mencari dan mengklaim buku. Kata *buy* yang berasosiasi dengan kata *book* dan *coin*, diperoleh informasi bahwa pengguna tidak menyukai cara membeli buku dengan koin. Kata *wait* diperoleh informasi dari asosiasi kata yang diperoleh, pengguna tidak menyukai terlalu lama menunggu beberapa jam bahkan beberapa hari, sehingga pengguna akan lupa karena lamanya menunggu.

Kata *app*, berdasarkan asosiasi kata dapat diperoleh informasi bahwa, aplikasi tidak memberikan timbal balik yang sesuai pada para penggunanya, aplikasi dianggap rakus karena tidak memberikan bonus untuk pengguna, aplikasi sering menutup secara berkala dalam pengoperasiannya, dan aplikasi tidak memberikan mode malam untuk pengguna. Kata *day*, berdasarkan tabel asosiasi, diperoleh beberapa informasi, untuk berlangganan bulanan bisa dicoba dengan percobaan tujuh hari dan tidak bisa dibatalkan karena aplikasi tidak sesuai dengan ekspektasi yang diharapkan. Kata *next* dapat diperoleh informasi berdasarkan asosiasi kata. Untuk *chapter* selanjutnya saling bertentangan, *chapter* selanjutnya saat dijalankan akan menutup dan kadang tidak merespon apapun pesan dari pengguna dan hal ini membuat pengguna khawatir.



Gambar 4. 16 Diagram *Fishbone* Keluhan Pengguna Dreame

Berdasarkan gambar 4.16 dapat diketahui faktor – faktor penyebab keluhan pengguna terhadap aplikasi Dreame. Daari faktor – faktor tersebut akan dicari pemecahan masalah dari keluhan pengguna. Keluhan – keluhan tersebut akan dibuat tabel pemecahan masalah sebagai solusi dari keluhan – keluhan pengguna.

Tabel 4. 11 Pemecahan Masalah Aplikasi Dreame

No	Faktor	Masalah	Pemecahan Masalah
1	*Process*	Terjadi kesalahan dalam mencari buku	Pihak developer harus memperbaiki sistem agar saat pengguna mencari buku tidak terjadi eror karena sistem yang sibuk, serta perbaikan untuk daftar *keyword*
		Kesalahan saat membeli buku	Pihak developer harus melakukan perbaikan sistem saat pengguna tidak gagal dalam pembelian buku dan pengguna tidak perlu kehilangan koin
		Proses membuka aplikasi lama	Developer harus melakukan *upgrade database* agar kapasitas dan menurunkan kapasitas memori agar aplikasi tidak berat sehingga proses membuka bisa cepat

		Tidak bisa klaim	Pihak developer harus memperbaiki sistem agar dalam klaim koin tidak terjadi kegagalan dan merugikan pengguna
2	*Product*	Harus membeli buku	Pihak Dreame harus dijelaskan lebih rinci ke pengguna, jika sistem membeli buku di sini menggunakan koin, bisa berbayar atau bisa diperoleh secara gratis dengan menunggu beberapa hari untuk mengumpulkan koin
		Buku dipenuhi iklan	Pihak developer harus memperbaiki sistem pada aplikasi agar iklan tidak menutupi bagian cerita, iklan bisa diberikan tempat tersendiri agar tidak mengganggu kenyamanan dalam membaca
		Bab saling bertentangan	Pihak developer harus memberikan filter pada aplikasi selayaknya editor, agar bab dalam buku bisa saling berkaitan, pihak developer juga bisa menjelaskan pada pengguna, bahwa tanggung jawab cerita ada di tangan penulis, pihak Dreame tidak bisa seleksi semua cerita dari bab per bab
3	*People*	Bab tidak sesuai	Pihak developer menjelaskan bahwa dalam cerita sepenuhnya ditanggung oleh penulis, kecuali jika terjadi penyalahgunaan konten
		Lama membuka bab	Pihak developer melakukan peningkatan sistem agar bisa beralih ke bab selanjutnya dengan

			cepat dan menjelaskan untuk membuka bab diperlukan koin sebagai royalti
		Buku tidak diselesaikan	Pihak developer dapat membuat sistem yang mampu memberikan peringatan pada penulis yang lama tidak menyelasikan bukunya agar segera menyelesaikan dan akan membatalkan publikasi jika batas waktu tidak terpenuhi
4	*Place*	*System error*	Sistem eror terjadi karena aplikasi lama dan aplikasi terlalu berat dan membutuhkan banyak ruang, pengguna sebaiknya menyiapkan lebih banyak ruang untuk aplikasi dan berusaha mengecilkan ukuran aplikasi dengan hapus data, pihak developer sebisa mungkin membuat aplikasi menjadi ringan dengan penghapusan data-data yang tidak perlu agar tidak terjadi banyak eror
5	*Price*	Sistem pembayaran dengan koin	Pihak developer lebih menjelaskan lagi pada pengguna, untuk sistem koin dapat diperoleh secara gratis dengan menunggu beberapa hari dan bisa juga membeli koin jika tidak mau menunggu
		Harus membayar setiap membuka bab	Pihak developer menjelaskan bahwa setiap membuka bab memerlukan koin yang akan diberikan pada penulis agar memperoleh timbal balik dari pembacanya

		Harus melakukan pembelian buku agar bisa membaca	Pihak Dreame menjelaskan bahwa pembelian buku bisa dilakukan dengan koin yang bisa diperoleh secara gratis
6	*Promotion*	Tidak diberikan timbal balik bagi pengguna	Pihak developer harus memberikan timbal balik sesuai yang dilakukan saat iklan aplikasi
		Aplikasi tidak sesuai dengan ekspektasi pengguna	Pihak developer harus lebih jujur dalam melakukan iklan, agar dapat memenuhi ekspektasi pengguna aplikasi Dreame
		Tidak diberikan bonus dari aplikasi premium	Pihak developer harus memberikan keistimewaan pada pengguna premium, karena telah membayar untuk berlangganan aplikasi, sehingga ada perbedaan rasa saat mengguakan aplikasi premium dan aplikasi dasar

4.2 Pembahasan

Gambaran umum berdasarkan *rating*, diperoleh hasil, sebanyak 33,12% pengguna sangat tidak menyukai, 17,04% tidak suka, 15,71% netral, 11,31% menyukai dan 22,82% sangat menyukai aplikasi Wattpad dari total 4137 *review* pengguna. Pada aplikasi Dreame dari 3090 *review* yang diperoleh, sebanyak 10,94% pengguna sangat tidak menyukai, 13,24% tidak menyukai, 27,86% netral, 21,85 menyukai dan 26,12% pengguna sangat menyukai. Maka, berdasakan *rating*, aplikasi Dreame lebih unggul dari aplikasi Wattpad.

Pembahasan hasil dari analisis sentimen dua aplikasi membaca dan menulis berdasarkan *review* pengguna di Google Play, didapatkan sentimen positif dan negatif dari kedua aplikasi. Untuk aplikasi Wattpad terdapat sebanyak 2356 *review* positif dan 1058 *review* negatif.

Aplikasi Dreame setelah melakukan pembersihan data dengan *text mining* dan dilakukan pelabelan, diperoleh sebanyak 1865 masuk dalam klasifikasi positif dan 761 masuk dalam klasifikasi negatif. Berdasarkan hasil dari pelabelan dua aplikasi, keduanya sama-sama lebih banyak disukai oleh penggunanya. Sehingga, kedua aplikasi ini memiliki keunggulan yang lebih banyak dari pada kekurangan masing – masing.

Setelah didapatkan pelabelan, data dibagi menjadi data uji dan data latih untuk dilakukan prediksi dengan *machine learning* SVM. Dari hasil uji akurasi kernel dalam SVM, akurasi terbaik untuk prediksi data *review* Wattpad menggunakan kernel Sigmoid dengan akurasi sebesar 88,60%, dengan prediksi tepat pada kelas positif sebanyak 230 data dan kelas negatif sebanyak 73 data, sedangkat terjadi salah prediksi positif sebanyak 23 data dan salah prediksi negatif sebanyak 16 data.

Aplikasi Dreame memiliki akurasi prediksi terbaik menggunakan kernel Linear dengan akurasi sebesar 87,45%, dengan prediksi data secara tepat untuk kelas positif sebanyak 181 data dan kelas negatif sebanyak 49 data, sedangkan kesalahan prediksi positif sebanyak 24 data dan kesalahan prediksi negatif sebanyak 9 data.

Berdasarkan data – data yang telah terlabeli, akan dilakukan asosiasi pada masing – masing kelas, aplikasi Wattpad kelas sentimen positif berasosiasi dengan membaca cerita secara *offline* dan gratis, sedangkan untuk aplikasi Dreame berasosiasi dengan membaca buku dengan sistem koin yang bisa diperoleh gratis dengan menunggu. Kelas sentimen negatif pada aplikasi Wattpad berasosiasi dengan *update* terbaru aplikasi tak bisa digunakan secara *offline*, ada sistem berbayar untuk membaca buku, dan terjadi kesalahan dalam registrasi serta aplikasi gagal dibuka, sedangkan pada aplikasi Dreame berkaitan dengan lamanya menunggu untuk membuka bab selanjutnya, harus membayar dengan koin untuk membuka bab, dan adanya sistem koin untuk membuka cerita.

Untuk setiap aplikasi dan akan ditemukan masalah yang telah dibuat dalam diagram *fishbone* yang kemudian dicari pemecahan masalah dari faktor – faktor yang ditemukan. Faktor – faktor yang ditemukan pada kedua aplikasi tersebut

meliputi, proses, produk, pelaku/orang, tempat, harga serta promosi yang ada pada kedua aplikasi.

Aplikasi Wattpad pada bagian proses mengalami masalah seperti, harus melakukan *login* ulang setelah pembaruan aplikasi, membutuhkan waktu lama dalam membukka aplikasi, terjadi kesalahan/gagal dalam membuka aplikasi dan terjadi kesalahan saat melakukan registrasi. Pada bagian produk terdapat masalah, buku tidak bisa dibaca secara *offline*, cerita pada aplikasi tertutup iklan dan buku aneh/cacat untuk dibaca. Masalah pada pelaku dalam aplikasi Wattpad adalah konten pada cerita salah, *Update* cerita lama. Pada tempat didapatkan masalah, *system error*. Masalah dalam harga adalah aplikasi berbayar, korupsi pembayaran aplikasi dan perlu *upgrade* aplikasi premium. Masalah pada bagian promosi adalah tidak bisa membaca cerita secara *offline* dan ada beberapa cerita berbayar.

Pemecahan dari masalah pada aplikasi Wattpad diperoleh sebagai berikut. Pihak developer harus memperbaiki sistem agar setelah melakukan pembaruan aplikasi, pengguna tidak perlu melakukan *login* kembali, pihak developer harus melakukan *update* sistem agar waktu untuk membuka aplikasi tidak terlalu lama *loading* dan berusaha untuk memperkecil ukuran aplikasi agar tidak terlalu berat, developer harus melakukan *upgrade database* agar kapasitas pengguna dalam melakukan kegiatan di aplikasi tidak terjadi kegagalan karena sistem pada *database* tidak sibuk, Pihak developer harus memperbaiki sistem *database* bagi pengguna agar mampu memuat penambahan pengguna agar bisa melakukan registrasi untuk pengguna baru, pihak developer seharusnya mengembalikan sistem versi lama yang bisa membaca buku secara *offline*, karena pengguna lebih nyaman membaca secara *offline*, pihak developer harus memperbaiki sistem pada aplikasi agar iklan tidak menutupi bagian cerita, iklan bisa diberikan tempat tersendiri agar tidak mengganggu kenyamanan dalam membaca, pihak developer harus memberikan filter pada aplikasi, buku mana yang layak untuk ditampilkan, sehingga perlu adanya sistem untuk peyeleksian cerita dengan kriteria tertentu, pihak penulis cerita harusnya bisa menyesuaikan konten cerita dengan benar sesuai isi cerita, pihak developer juga harus melakukan penyeleksian secara sistem dengan kesesuaian isi cerita dengan *genre*, pihak developer bisa membuat sistem dengan batas waktu

untuk melakukan *update* cerita agar tidak terlalu lama kecuali cerita memang telah selesai, developer bisa melakukan pembatalan publikasi cerita jika cerita tidak *update* dalam jagka waktu tertentu, sistem eror terjadi karena aplikasi lama dan aplikasi terlalu berat dan membutuhkan banyak ruang dalam ponsel ataupun laptop, pengguna sebaiknya menyiapkan lebih banyak ruang untuk aplikasi dan berusaha mengecilkan ukuran aplikasi dengan hapus data, pihak developer sebisa mungkin membuat aplikasi menjadi ringan dengan penghapusan data-data yang tidak perlu agar tidak terjadi banyak eror, pihak developer memberikan peringatan pada pengguna bahwa memang ada aplikasi dengan mode berbayar khusus untuk premium, sedangkan untuk versi dasar aplikasi masih bisa dinikmati secara gratis, pihak developer lebih menjelaskan pembayaran digunakan untuk memberikan suasana berbeda bagi pengguna yang premium dengan tidak adanya iklan, sehingga pihak developer juga cerita berbayar akan masuk pada penulis sebagai royalti telah menghibur pembaca, developer lebih menekankan bahwa tidak semua harus dilakukan *upgrade* ke dalam versi premium, versi premium digunakan untuk pengguna yang ingin tidak ada iklan saat membuka aplikasi, pihak developer harus mengembalikan sistem seperti semula agar seuai dengan iklan yang dilakukan, membaca secara *offline* dan gratis yang menjadi keunggulan dari aplikasi dan pihak developer harus konsisten agar pengguna bisa membaca secara gratis yang sudah menjadi keunggulan aplikasi bagi para pengguna.

Masalah-masalah yang dikeluhkan pengguna Dreame dirangkum sebagai berikut. Proses memunjukan beberapa masalah yang dikeluhakan seperti, terjadi kesalahan dalam mencari buku, kesalahan saat membeli buku, proses membuka aplikasi lama dan tidak bisa klaim. Untuk permasalahan pada produk adalah harus membeli buku, buku dipenuhi iklan dan bab saling bertentangan. Pelaku terdapat masalah, bab tidak sesuai, lama membuka bab dan buku tidak diselesaikan. Tempat terdapat *system error* sebagai permasalahannya. Pada bagian harga terdapat masalah, sistem pembayaran dengan koin, harus membayar setiap membuka bab dan harus melakukan pembelian buku agar bisa membaca. Terakhir untuk bagian promosi, tidak diberikan timbal balik bagi pengguna, aplikasi tidak sesuai dengan ekspektasi pengguna dan tidak diberikan bonus dari aplikasi premium.

Pemecahan masalah untuk aplikasi Dreame didapatkan sebagai berikut. Pihak developer harus memperbaiki sistem agar saat pengguna mencari buku tidak terjadi eror karena sistem yang sibuk, serta perbaikan untuk daftar *keyword*, pihak developer harus melakukan perbaikan sistem saat pengguna tidak gagal dalam pembelian buku dan pengguna tidak perlu kehilangan koin, developer harus melakukan *upgrade database* agar kapasitas dan menurunkan kapasitas memori agar aplikasi tidak berat sehingga proses membuka bisa cepat, pihak developer harus memperbaiki sistem agar dalam klaim koin tidak terjadi kegagalan dan merugikan pengguna, pihak Dreame harus dijelaskan lebih rinci ke pengguna, jika sistem membeli buku di sini menggunakan koin, bisa berbayar atau bisa diperoleh secara gratis dengan menunggu beberapa hari untuk mengumpulkan koin, pihak developer harus memperbaiki sistem pada aplikasi agar iklan tidak menutupi bagian cerita, iklan bisa diberikan tempat tersendiri agar tidak mengganggu kenyamanan dalam membaca, pihak developer harus memberikan filter pada aplikasi selayaknya editor, agar bab dalam buku bisa saling berkaitan, pihak developer juga bisa menjelaskan pada pengguna, bahwa tanggung jawab cerita ada di tangan penulis, pihak Dreame tidak bisa seleksi semua cerita dari bab per bab, pihak developer menjelaskan bahwa dalam cerita sepenuhnya ditanggung oleh penulis, kecuali jika terjadi penyalahgunaan konten, pihak developer melakukan peningkatan sistem agar bisa beralih ke bab selanjutnya dengan cepat dan menjelaskan untuk membuka bab diperlukan koin sebagai royalti, pihak developer dapat membuat sistem yang mampu memberikan peringatan pada penulis yang lama tidak menyelasikan bukunya agar segera menyelesaikan dan akan membatalkan publikasi jika batas waktu tidak terpenuhi, sistem eror terjadi karena aplikasi lama dan aplikasi terlalu berat dan membutuhkan banyak ruang, pengguna sebaiknya menyiapkan lebih banyak ruang untuk aplikasi dan berusaha mengecilkan ukuran aplikasi dengan hapus data, pihak developer sebisa mungkin membuat aplikasi menjadi ringan dengan penghapusan data-data yang tidak perlu agar tidak terjadi banyak eror, pihak developer lebih menjelaskan lagi pada pengguna, untuk sistem koin dapat diperoleh secara gratis dengan menunggu beberapa hari dan bisa juga membeli koin jika tidak mau menunggu, pihak developer menjelaskan bahwa setiap membuka

bab memerlukan koin yang akan diberikan pada penulis agar memperoleh timbal balik dari pembacanya, pihak Dreame menjelaskan bahwa pembelian buku bisa dilakukan dengan koin yang bisa diperoleh secara gratis, pihak developer harus memberikan timbal balik sesuai yang dilakukan saat iklan aplikasi, pihak developer harus lebih jujur dalam melakukan iklan, agar dapat memenuhi ekspektasi pengguna aplikasi Dreame dan pihak developer harus memberikan keistimewaan pada pengguna premium, karena telah membayar untuk berlangganan aplikasi, sehingga ada perbedaan rasa saat mengguakan aplikasi premium dan aplikasi dasar.

BAB V

PENUTUP

5.1 Simpulan

Berdasarkan hasil dan pembahasan yang telah dipaparkan di bab sebelumya, maka diperoleh kesimpulan sebagai berikut:

1. Berdasarkan *rating* aplikasi yang diperoleh dari data *review* di Google Play, sebanyak 33,12% pengguna sangat tidak menyukai, 17,04% tidak suka, 15,71% netral, 11,31% menyukai dan 22,82% sangat menyukai aplikasi Wattpad dari total 4137 *review* pengguna, sedangkan untuk aplikasi Dreame dari 3090 *review* yang diperoleh, sebanyak 10,94% pengguna sangat tidak menyukai, 13,24% tidak menyukai, 27,86% netral, 21,85 menyukai dan 26,12% pengguna sangat menyukai. Maka, berdasakan *rating*, Dreame lebih unggul dari Wattpad.
2. Berdasarkan data latih dan data uji dengan SVM diperoleh tingkat akurasi untuk aplikasi Wattpad sebesar 88,60% artinya dari 342 data yang diujikan, terdapat 303 data yang terdiri dari 230 data kelas positif dan 73 data kelas negatif yang terklasifikasikan dengan tepat, sedangkan sebanyak 39 data yang terdiri dari 23 data negatif dan 16 data positif yang salah klasifikasi. Aplikasi Dreame sebesar 87,45% artinya dari 263 data yang diujikan, terdapat 230 data yang terdiri dari 181 data kelas positif dan 49 data kelas negatif yang terklasifikasikan dengan tepat, sedangkan sebanyak 33 data yang terdiri dari 23 data negatif dan 9 data positif yang terklasifikasi salah.
3. Berdasarkan klasifikasi dan asosiasi kata yang dilakukan, aplikasi Wattpad kelas sentimen positif berkaitan dengan membaca cerita secara *offline* dan gratis, sedangkan untuk aplikasi Dreame berkaitan dengan membaca buku dengan sistem koin yang bisa diperoleh gratis dengan menunggu. Kelas sentimen negatif pada aplikasi Wattpad berkaitan dengan *update* terbaru aplikasi tak bisa digunakan secara *offline*, ada sistem berbayar untuk membaca buku, dan terjadi kesalahan dalam registrasi serta aplikasi gagal

dibuka, sedangkan pada aplikasi Dreame berkaitan dengan lamanya menunggu untuk membuka bab selanjutnya, harus membayar dengan koin untuk membuka bab, dan adanya sistem koin untuk membuka cerita.

4. Berdasarkan analisis dengan diagram sebab – akibat (diagram *fishbone*) pada aplikasi Wattpad terdapat 15 permasalahan dari hasil identifikasi menggunakan asosiasi kata berdasarkan ulasan negatif dan pada aplikasi Dreame terdapat 17 permasalahan yang berhasil teridentifikasi dan dari permaslahan kedua aplikasi diklasifikasikan menjadi 6P faktor yaitu, *Process, Product, People, Place, Price* dan *Promotion*.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, dapat diberikan saran sebagai berikut:

1. Pembaca agar mempertimbangkan baik – baik dalam memilih aplikasi yang sesuai dengan kebutuhannya dan dirasa cocok dari ekstraksi-ekstrasi permasalahan masing-masing aplikasi yang dikeluhkan pengguna.
2. Penelitian selanjutnya, dalam mengambil data perlu ditambah lagi dan dibatasi periode tertentu pada Google Play.
3. Penelitian ini untuk pelabelan data hanya menggunakan metode Vader Lexicon yang ada dalam Python yang menggunakan kata-kata yang kurang akurat dalam pelabelan, untuk penelitian selanjutnya jika dimungkinkan menggunkan pelabelan yang mampu mendeteksi kalimat.
4. Penelitian selanjutnya lebih baik menggunakan data *review* tidak hanya dengan bahasa Inggris, tapi juga bisa ditambah bahasa Indonesia.

DAFTAR PUSTAKA

Andayani, S., & Ryansyah, A. (2017). Implementasi Algoritma TF-IDF Pada Pengukuran Kesamaan Dokumen. *JuSiTik : Jurnal Sistem dan Teknologi Informasi Komunikasi*, 53.

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook.* New York: Cambridge University Press.

Fikria, N. (2018). ANALISIS KLASIFIKASI SENTIMEN REVIEW APLIKASI E-TICKETING MENGGUNAKAN METODE SUPPORT VECTOR MACHINE DAN ASOSIASI (Studi Kasus : Review Aplikasi KAI Access dan Tiket.com pada Google Play).

Heri Murnawan, M. (2014). Perencaan Produktivitas Kerja dari Hasil Evaluasi Produktivitas dengan Metode Fishbone di Perusahaan Percetakaan Kemasan PT.X. *Jurnal Teknik Industri HEURISTIC*, 27-46.

Ihsan, M., Roza, E., & Widodo, E. (2019). Analisis Sentimen Twitter terhadap Bom Bunuh Diri di Surabaya 13 Mei 2018 menggunakan Pendekatan Support Vector Machine. *Prisma 2*, 416-426.

Ilmawan, L. B. (2018). Membangun Web Crawler Berbasis Web Service Untuk. 215-224.

Indraloka, D. S., & Santosa, B. (2017). Penerapan Text Mining untuk Melakukan Clustering Data Tweet Shopee Indonesia. *Jurnal Sains dan Seni ITS*, 6-11.

Lazuardi, D. R. (2014). Analisis Sentimen Untuk Mengetahui Persepsi Kualitas Merek Menggunakan Text Mining Dan Social Network Analysis Pada Konten Percakapan Di Media Sosial Twitter PEMBAHASAN A . Network Telkomsel. 1-9.

Mardi, Y. (2017). Data Mining : Klasifikasi Menggunakan Algoritma C4.5. *Jurnal Edik Informatika*, 213-219.

Masripah, S. (2016). Komparasi Algoritma Klasifikasi Data Mining untuk Evaluasi Pemberian Kredit. *Bina Insani ICT Journal*, 187-193.

Nasution, L. M. (2017). STATISTIK DESKRIPTIF.

Perdana, A., & Furqon, M. T. (2018). Penerapan Algoritma Support Vector Machine (SVM) Pada Pengklasifikasian Penyakit Kejiwaan Skizofrenia (Studi Kasus : RSJ . Radjiman Wediodiningrat , Lawang). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 3162-3167.

Perkasa, T. R., Widyantara, H., & Susanto, P. (2014). Rancang Bangun Pendeteksi Gerak Menggunakan Metode Image Substraction Pada Single Board Computer (SBC). *Journal of Control and Network Systems*, 90-97.

Pratama, A., Wihandika, R. C., & Ratnawati, D. E. (2018). Implementasi Algoritme Support Vector Machine (SVM) untuk Prediksi Ketepatan Waktu Kelulusan Mahasiswa. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 1704-1708.

Purbo, O. W. (2019). *TEXT MINING - ANALISIS MEDSOS, KEAKURATAN BRAND & INTELIJEN DI INTERNET.* Yogyakarta: Andi.

Putri, E. K., & Setiadi, T. (2014). PENERAPAN TEXT MINING PADA SISTEM KLASIFIKASI EMAIL SPAM MENGGUNAKAN NAIVE BAYES. *Sarjana Teknik Informatika*, 73-83.

R, K. R., Rahmansyah, A., & Darwin, W. (2017). Penggunaan Bahasa Pemrograman Python Sebagai Pusat Kendali Pada Robot 10-D. *5th Indonesian Symposium on Robotic Systems and Control*, 23-26.

Sembiring, K. (2007). Penerapan Teknik Support Vector Machine untuk Pendeteksian Intrusi pada Jaringan. *Skripsi : Program Studi Teknik Informatika Institut Teknologi Bandung*.

Setyobudi, W., Alwi, A., & Astuti, I. P. (2018). SENTIMEN ANALISIS TWITTER TERHADAP PENYELENGGARAAN GOJEK TRAVELOKA LIGA 1 INDONESIA. 70-80.

Sudiantoro, A. V., & Zuliarso, E. (2018). ANALISIS SENTIMEN TWITTER MENGGUNAKAN TEXT MINING DENGAN. 398-401.

Telaumbanua, F. D., Hulu, P., Nadeak, T. Z., Lumbantong, R. R., & Dharma, A. (2019). Penggunaan Machine Learning di Bidang Kesehatan. 57-64.

TFIDF Separate for Each Label. (2020, march 15). Retrieved from Stack Overflow: https://stackoverflow.com/questions/60686556/tfidf-separate-for-each-label

Ulfa, S. A. (2018). PERANAN APLIKASI WATTPAD DALAM MENGASAH KEMAMPUAN MENULIS.

Utami, Y. R., & Susyanto, T. (2012). UNJUK KERJA METODE KLASIFIKASI SUPPORT VECTOR MACHINE (SVM) DENGAN LEARNING VECTOR QUANTIZATION (LVQ) PADA APLIKASI PENGENALAN WAJAH. 9-18.

LAMPIRAN

Lampiran 1 – Script Python Scrapping Review Wattpad

```
#load webdriver function from selenium
from selenium import webdriver
from time import sleep
import bs4
import pandas as pd
import requests
from selenium.webdriver.common.keys import Keys
import time


# Change this number to get more or less reviews
x = 100

link =
"https://play.google.com/store/apps/details?id=wp.wattpad&showAllR
eviews=true"

driver = webdriver.Chrome('chromedriver.exe')
driver.get(link)


num_clicks = 0
num_scrolls = 0
while num_clicks <= x and num_scrolls <= x*2:
    try:
        show_more =
driver.find_element_by_xpath('//*[@id="fcxH9b"]/div[4]/c-
wiz[2]/div/div[2]/div/div[1]/div/div/div[1]/div[2]/div/content/spa
n').click()
        num_clicks += 1

    except:
        html = driver.find_element_by_tag_name('html')
        html.send_keys(Keys.END)
        num_scrolls +=1

soup = bs4.BeautifulSoup(driver.page_source, 'html.parser')
h2 = soup.find_all('h2')

results_df = pd.DataFrame()
for ele in h2:
    if ele.text == 'Reviews':
        c_wiz = ele.parent.parent.find_all('c-wiz')
        for sibling in c_wiz[0].next_siblings:
            try:
                #print (sibling)
                comment_shift = 0
                spans = sibling.find_all('span')
                #print (spans)
                for user_block in range(0,len(spans)):
                    i = user_block *1
                    name = spans[i+0+comment_shift].text
                    try:
```

```
                        rating =
spans[i+1+comment_shift].div.next_element['aria-label']
                        rating = str(''.join(filter(str.isdigit,
rating)))
                    except:
                        comment_shift += 2
                        continue
                    date = spans[i+2+comment_shift].text
                    review = spans[i+12+comment_shift].text
                    if review == 'Link to this review':
                        review = spans[i+14+comment_shift].text
                    else :
                        review = spans[i+13+comment_shift].text
                        if review == '':
                            review =
spans[i+12+comment_shift].text
                        else :
                            review =
spans[i+13+comment_shift].text

                    print ('Name: %s\nRating: %s\nDate:
%s\nReview: %s\n' %(name, rating, date, review))
                    temp_df = pd.DataFrame([[date, rating, name,
review]], columns = ['Date','Rating','User','Review'])

                    results_df = results_df.append(temp_df)
            except:
                continue



results_df = results_df.reset_index(drop=True)
results_df.to_csv('reviewsWattpad7.csv', index=False)


driver.close()
```

Lampiran 2 – Script Python Scrapping Review Dreame

```
#load webdriver function from selenium
from selenium import webdriver
from time import sleep
import bs4
import pandas as pd
import requests
from selenium.webdriver.common.keys import Keys
import time


# Change this number to get more or less reviews
x = 200

link =
"https://play.google.com/store/apps/details?id=com.dreame.reader&s
howAllReviews=true"

driver = webdriver.Chrome('chromedriver.exe')
driver.get(link)


num_clicks = 0
num_scrolls = 0
while num_clicks <= x and num_scrolls <= x*3:
    try:
        show_more =
driver.find_element_by_xpath('//*[@id="fcxH9b"]/div[4]/c-
wiz[2]/div/div[2]/div/div[1]/div/div/div[1]/div[2]/div/content/spa
n').click()
        num_clicks += 1

    except:
        html = driver.find_element_by_tag_name('html')
        html.send_keys(Keys.END)
        num_scrolls +=1

soup = bs4.BeautifulSoup(driver.page_source, 'html.parser')
h2 = soup.find_all('h2')

results_df = pd.DataFrame()
for ele in h2:
    if ele.text == 'Reviews':
        c_wiz = ele.parent.parent.find_all('c-wiz')
        for sibling in c_wiz[0].next_siblings:
            try:
                #print (sibling)
                comment_shift = 0
                spans = sibling.find_all('span')
                #print (spans)
                for user_block in range(0,len(spans)):
                    i = user_block *1
                    name = spans[i+0+comment_shift].text
                    try:
```

```
                        rating =
spans[i+1+comment_shift].div.next_element['aria-label']
                        rating = str(''.join(filter(str.isdigit,
rating)))
                    except:
                        comment_shift += 2
                        continue
                    date = spans[i+2+comment_shift].text
                    review = spans[i+12+comment_shift].text
                    if review == 'Link to this review':
                        review = spans[i+14+comment_shift].text
                    else :
                        review = spans[i+13+comment_shift].text
                        if review == '':
                            review =
spans[i+12+comment_shift].text
                        else :
                            review =
spans[i+13+comment_shift].text

                    print ('Name: %s\nRating: %s\nDate:
%s\nReview: %s\n' %(name, rating, date, review))
                    temp_df = pd.DataFrame([[date, rating, name,
review]], columns = ['Date','Rating','User','Review'])

                    results_df = results_df.append(temp_df)
            except:
                continue



results_df = results_df.reset_index(drop=True)
results_df.to_csv('reviewsDreame5.csv', index=False)


driver.close()
```

Lampiran 3 – Script Python Analisis Sentimen dan Visualisasi Wattpad

```
import pickle
import pandas as pd
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
global str
from nltk import FreqDist
import gensim
from gensim import corpora
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


df = pd.read_csv('reviewsWattpad12.csv')

print(df.shape)


df.head()


# # Analisis Deskriptif


# coba kita lihat berapa banyak jumlah data untuk tiap rate
df["Rating"].value_counts()


df["Date"].value_counts()[:100]



# kita lihat dulu distribusi label
import matplotlib.pyplot as plt

_, ax1 = plt.subplots(figsize=(10,4))

stars_histogram = df["Rating"].value_counts().sort_index()

stars_histogram.plot(kind="bar", width=1.0)
plt.tight_layout()
plt.show()


# # preprocessing


df = df.dropna()
df = df.drop([0])
```

```
review = []
for index, row in df.iterrows():

review.append(nltk.stem.WordNetLemmatizer().lemmatize(row["Review"
]))

df["Review"] = review
df.head()

df['Review'] = df[['Review']].astype(str)

df['Review'] = df['Review'].str.replace('\d+', '')
# get rid of special characters
df['Review'] = df['Review'].str.replace(r'[^\w\s]+', '')
# get rid fo double spaces
df['Review'] = df['Review'].str.replace(r'\^[a-zA-Z]\s+', '')
# convert all case to lower
df['Review'] = df['Review'].str.lower()


review_text_list = df['Review'].tolist()

reviews = review_text_list
s = difflib.SequenceMatcher(None, reviews).ratio()
print ("ratio:", s, "\n")


# filter out stop words
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text = str(review_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)


# convert to lower case
tokens = [w.lower() for w in tokens]


# remove punctuation from each word
import string
table = str.maketrans('', '', string.punctuation)
stripped = [w.translate(table) for w in tokens]


# remove remaining tokens that are not alphabetic
words = [word for word in stripped if word.isalpha()]


# filter out stop words
from nltk.corpus import stopwords
```

```
stop_words = set(stopwords.words('english'))
words = [w for w in words if not w in stop_words]
print(words[:100])

df['Review'] = df['Review'].str.replace("[^a-zA-Z#]", " ")


df.head()


df['Review'] = pd.Series(df['Review'])


df.head()


review = []
for index, row in df.iterrows():

review.append(nltk.stem.WordNetLemmatizer().lemmatize(row["Review"
]))

df["Review"] = review
df.head()


from textblob import TextBlob
df['Review'] = df['Review'].apply(lambda x:
str(TextBlob(x).correct()))


df.head()

from nltk.corpus import stopwords
stop_words = stopwords.words('english')


# function to remove stopwords
def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new


# remove short words (length < 3)
df['Review'] = df['Review'].apply(lambda x: ' '.join([w for w in
x.split() if len(w)>2]))



df.head()


# remove stopwords from the text
df['Review'] = [remove_stopwords(r.split()) for r in df['Review']]
```

```
# make entire text lowercase
df['Review'] = [r.lower() for r in df['Review']]


df.head()


# # Frequence and TFIDF


# function to plot most frequent terms
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()

  fdist = FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()),
'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)
  plt.figure(figsize=(20,5))
  ax = sns.barplot(data=d, x= "word", y = "count")
  ax.set(ylabel = 'Count')
  plt.show()

more significant words have come out.

freq_words(df['Review'], 35)


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
cvec = CountVectorizer(stop_words='english', min_df=1, max_df=.5,
ngram_range=(1,2))
cvec


from itertools import islice
cvec.fit(review_text_list)
list(islice(cvec.vocabulary_.items(), 20))

len(cvec.vocabulary_)
cvec = CountVectorizer(stop_words='english', min_df=.0025,
max_df=.5, ngram_range=(1,2))
cvec.fit(review_text_list)
len(cvec.vocabulary_)

cvec_counts = cvec.transform(review_text_list)
print('sparse matrix shape:', cvec_counts.shape)
print('nonzero count:', cvec_counts.nnz)
print('sparsity: %.2f%%' % (100.0 * cvec_counts.nnz /
(cvec_counts.shape[0] * cvec_counts.shape[1])))

# get counts of frequently occurring terms; top 20
occ = np.asarray(cvec_counts.sum(axis=0)).ravel().tolist()
```

```
counts_df = pd.DataFrame({'term': cvec.get_feature_names(),
'occurrences': occ})
counts_df.sort_values(by='occurrences', ascending=False).head(20)



transformer = TfidfTransformer()
transformed_weights = transformer.fit_transform(cvec_counts)
transformed_weights


weights =
np.asarray(transformed_weights.mean(axis=0)).ravel().tolist()
weights_df = pd.DataFrame({'term': cvec.get_feature_names(),
'weight': weights})
weights_df.sort_values(by='weight', ascending=False).head(20)


# # Labeling Sentiment

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
df['sentiment'] = df['Review'].apply(lambda x:
sid.polarity_scores(x))
def convert(x):
    if x <= 0.2:
        return "negative"
    #elif x > .2:
        #return "positive"
    else:
        return "positive"
df['result'] = df['sentiment'].apply(lambda
x:convert(x['compound']))


# df.groupby(['brand','result']).size()
# df.groupby(['brand','result']).count()
x =
df.groupby(['Review','User'])['result'].value_counts(normalize=Tru
e)

x = df.groupby(['Review'])['result'].value_counts(normalize=True)
y = x.loc[(x.index.get_level_values(1) == 'negative')]
#print(y[y>0.2])


df.head()


print(df.shape)


df['result'].value_counts()
```

```
# # SVM


finalWattpad = df
finalWattpad = df.drop(columns=['Date', 'User', 'Rating',
'sentiment'])
finalWattpad.head()


# mengubah result
label = []
for index, row in finalWattpad.iterrows():
    if row["result"] == 'positive':
        label.append(1)
    else:
        label.append(0)

finalWattpad["label"] = label
finalWattpad = finalWattpad.drop(columns=['result'])
finalWattpad.tail()


finalWattpad.head()


from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, svm
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder


Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(finalWattpad['Review'],finalWattp
ad['label'],test_size=0.1, random_state=8)


Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)


Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(finalWattpad['Review'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)


# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)


# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
```

```
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM,
Test_Y)*100)


#SVM RBF
SVM_RBF = svm.SVC(C=1.0, kernel='rbf', gamma = 'scale')
SVM_RBF.fit(Train_X_Tfidf,Train_Y)

# predict the labels on validation dataset
predictions_SVM_RBF = SVM_RBF.predict(Test_X_Tfidf)

print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM_RBF,
Test_Y)*100)


#SVM Polynomial
SVM_poly = svm.SVC(C=1.0, kernel='poly', gamma = 'scale')
SVM_poly.fit(Train_X_Tfidf,Train_Y)

predictions_SVM_poly = SVM_poly.predict(Test_X_Tfidf)

print("SVM Accuracy Score ->
",accuracy_score(predictions_SVM_poly, Test_Y)*100)


#SVM Sigmoid
SVM_sig = svm.SVC(C=1.0, kernel='sigmoid', gamma = 'scale')
SVM_sig.fit(Train_X_Tfidf,Train_Y)

predictions_SVM_sig = SVM_sig.predict(Test_X_Tfidf)

print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM_sig,
Test_Y)*100)


#confusion matrix for train data
#Confusion matrix using heatmap for train data
from sklearn.metrics import confusion_matrix

#predic=SVM.predict(Test_X_Tfidf)
import seaborn as sns
conf_mat = confusion_matrix(Test_Y, predictions_SVM_sig)
class_label = ["negative", "positive"]
test_sig = pd.DataFrame(conf_mat, index = class_label, columns =
class_label)
sns.heatmap(test_sig, annot = True,fmt="d")
plt.title("Confusion Matrix for test data")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()


# # Visualisasi
```

```
WattpadNeg = finalWattpad.loc[finalWattpad['label']== 0]
WattpadNeg.head()


# It looks like the language in body and review_text is very
similar (2 fields in dataframe).  let's check how closely they
match...
# seems like the tone is similar, but the text is not matching at
a high rate...less than 20% match rate
import difflib

#body_list = df['body'].tolist()
neg_text_list = WattpadNeg['Review'].tolist()

#body = body_list
reviews_neg = neg_text_list


# filter out stop words
# these are the most common words such as: "the", "a", and "is".
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text_neg = str(neg_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens_neg = word_tokenize(text_neg)


# filter out stop words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
negative = [w for w in tokens_neg if not w in stop_words]
extend = "dont", "cant", "didnt", "wattpad"
negative = [w for w in negative if not w in extend]
print(negative[:100])


#negative = [nltk.stem.PorterStemmer().stem(word) for word in
negative]


negative = [nltk.stem.WordNetLemmatizer().lemmatize(word) for word
in negative]


negative


import string
table = str.maketrans('', '', string.punctuation)
strippedneg = [w.translate(table) for w in negative]
```

```
#Let’s again plot the most frequent words and see if the more
significant words have come out.
freq_words(strippedneg, 20)


from wordcloud import WordCloud
from wordcloud import ImageColorGenerator

# negative
all_text_negative = ' '.join(str(word) for word in strippedneg)
wordcloud = WordCloud(max_font_size=260, max_words=100,
width=1000, height=1000, mode='RGBA',
background_color='white').generate(all_text_negative)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()


WattpadPos = finalWattpad.loc[finalWattpad['label']==1]
WattpadPos.head()


# It looks like the language in body and review_text is very
similar (2 fields in dataframe).  let's check how closely they
match...
# seems like the tone is similar, but the text is not matching at
a high rate...less than 20% match rate
import difflib

#body_list = df['body'].tolist()
pos_text_list = WattpadPos['Review'].tolist()

#body = body_list
reviews_pos = pos_text_list


# filter out stop words
# these are the most common words such as: “the“, “a“, and “is“.
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text_pos = str(pos_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens_pos = word_tokenize(text_pos)


# filter out stop words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
positive = [w for w in tokens_pos if not w in stop_words]
positive = [w for w in positive if not w in extend]
```

```
print(positive[:100])


#positive = [nltk.stem.PorterStemmer().stem(word) for word in
positive]



positive = [nltk.stem.WordNetLemmatizer().lemmatize(word) for word
in positive]
print(positive[:100])


import string
table = str.maketrans('', '', string.punctuation)
strippedpos = [w.translate(table) for w in positive]


#Let’s again plot the most frequent words and see if the more
significant words have come out.
freq_words(strippedpos, 20)


from wordcloud import WordCloud
from wordcloud import ImageColorGenerator

# positive
all_text_positive = ' '.join(str(word) for word in strippedpos)
wordcloud = WordCloud(max_font_size=260, max_words=100,
width=1000, height=1000, mode='RGBA',
background_color='white').generate(all_text_positive)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```

Lampiran 4 - Script Python Analisis Sentimen dan Visualisasi Dreame

```
import pickle
import pandas as pd
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
global str
from nltk import FreqDist
import gensim
from gensim import corpora
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


df = pd.read_csv('reviewsDreame5.csv')

print(df.shape)



df.head()


# # Analisis Deskriptif


# coba kita lihat berapa banyak jumlah data untuk tiap rate
df["Rating"].value_counts()


df["Date"].value_counts()[:10]


# kita lihat dulu distribusi label
import matplotlib.pyplot as plt

_, ax1 = plt.subplots(figsize=(10,4))

stars_histogram = df["Rating"].value_counts().sort_index()

stars_histogram.plot(kind="bar", width=1.0)
plt.tight_layout()
plt.show()


# # preprocessing


df = df.dropna()
```

```
review = []
for index, row in df.iterrows():

review.append(nltk.stem.WordNetLemmatizer().lemmatize(row["Review"
]))

df["Review"] = review
df.head()


df['Review'] = df[['Review']].astype(str)

df['Review'] = df['Review'].str.replace('\d+', '')
# get rid of special characters
df['Review'] = df['Review'].str.replace(r'[^\w\s]+', '')
# get rid fo double spaces
df['Review'] = df['Review'].str.replace(r'\^[a-zA-Z]\s+', '')
# convert all case to lower
df['Review'] = df['Review'].str.lower()


import difflib
review_text_list = df['Review'].tolist()
reviews = review_text_list
s = difflib.SequenceMatcher(None, reviews).ratio()
print ("ratio:", s, "\n")


# filter out stop words
# these are the most common words such as: "the", "a", and "is".
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text = str(review_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)


# convert to lower case
tokens = [w.lower() for w in tokens]


# remove punctuation from each word
import string
table = str.maketrans('', '', string.punctuation)
stripped = [w.translate(table) for w in tokens]


# remove remaining tokens that are not alphabetic
words = [word for word in stripped if word.isalpha()]


# filter out stop words
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
words = [w for w in words if not w in stop_words]
print(words[:100])


# plot most frequently occurring words in a bar chart
# remove unwanted characters, numbers and symbols
df['Review'] = df['Review'].str.replace("[^a-zA-Z#]", " ")


df.head()


df['Review'] = pd.Series(df['Review'])


df.head()


review = []
for index, row in df.iterrows():

review.append(nltk.stem.WordNetLemmatizer().lemmatize(row["Review"
]))

df["Review"] = review
df.head()


from textblob import TextBlob
df['Review'] = df['Review'].apply(lambda x:
str(TextBlob(x).correct()))


df.head()


#Let’s try to remove the stopwords and short words (<2 letters)
from the reviews.
from nltk.corpus import stopwords
stop_words = stopwords.words('english')


# function to remove stopwords
def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new


# remove short words (length < 3)
df['Review'] = df['Review'].apply(lambda x: ' '.join([w for w in
x.split() if len(w)>2]))


df.head()
```

```
# remove stopwords from the text
df['Review'] = [remove_stopwords(r.split()) for r in df['Review']]


# make entire text lowercase
df['Review'] = [r.lower() for r in df['Review']]


df.head()


# # Frequence and TFIDF



# function to plot most frequent terms
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()

  fdist = FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()),
'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)
  plt.figure(figsize=(20,5))
  ax = sns.barplot(data=d, x= "word", y = "count")
  ax.set(ylabel = 'Count')
  plt.show()


freq_words(df['Review'], 35)


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
cvec = CountVectorizer(stop_words='english', min_df=1, max_df=.5,
ngram_range=(1,2))
cvec


# Calculate all the n-grams found in all documents
from itertools import islice
cvec.fit(review_text_list)
list(islice(cvec.vocabulary_.items(), 20))

len(cvec.vocabulary_)


cvec = CountVectorizer(stop_words='english', min_df=.0025,
max_df=.5, ngram_range=(1,2))
cvec.fit(review_text_list)
len(cvec.vocabulary_)
```

```
cvec_counts = cvec.transform(review_text_list)
print('sparse matrix shape:', cvec_counts.shape)
print('nonzero count:', cvec_counts.nnz)
print('sparsity: %.2f%%' % (100.0 * cvec_counts.nnz /
(cvec_counts.shape[0] * cvec_counts.shape[1])))

occ = np.asarray(cvec_counts.sum(axis=0)).ravel().tolist()
counts_df = pd.DataFrame({'term': cvec.get_feature_names(),
'occurrences': occ})
counts_df.sort_values(by='occurrences', ascending=False).head(20)

# weights for each term in each document
transformer = TfidfTransformer()
transformed_weights = transformer.fit_transform(cvec_counts)
transformed_weights


# we can take a look at the top 20 terms by average tf-idf weight.
weights =
np.asarray(transformed_weights.mean(axis=0)).ravel().tolist()
weights_df = pd.DataFrame({'term': cvec.get_feature_names(),
'weight': weights})
weights_df.sort_values(by='weight', ascending=False).head(20)


# # Labeling Sentiment


from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
df['sentiment'] = df['Review'].apply(lambda x:
sid.polarity_scores(x))
def convert(x):
    if x < 0.2:
        return "negative"
    #elif x > .2:
        #return "positive"
    else:
        return "positive"
df['result'] = df['sentiment'].apply(lambda
x:convert(x['compound']))


# df.groupby(['brand','result']).size()
# df.groupby(['brand','result']).count()
x =
df.groupby(['Review','User'])['result'].value_counts(normalize=Tru
e)

x = df.groupby(['Review'])['result'].value_counts(normalize=True)
y = x.loc[(x.index.get_level_values(1) == 'negative')]
#print(y[y>0.2])
```

```
df['sentiment'][0]


df['Review'][0]


print(df.shape)


df['result'].value_counts()


# # SVM


finalDreame = df
finalDreame = df.drop(columns=['Date', 'User', 'Rating',
'sentiment'])
finalDreame.head()


# mengubah result
label = []
for index, row in finalDreame.iterrows():
    if row["result"] == 'positive':
        label.append(1)
    else:
        label.append(0)

finalDreame["label"] = label
finalDreame = finalDreame.drop(columns=['result'])
finalDreame.tail()

finalDreame.head()


from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, svm
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score


Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(finalDreame['Review'],finalDreame
['label'],test_size=0.1, random_state=9)


Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)


Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(finalDreame['Review'])
```

```
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)


# Classifier - Algorithm - SVM
# fit the training dataset on the classifier

SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)


# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)


#confusion matrix for train data
#Confusion matrix using heatmap for train data
from sklearn.metrics import confusion_matrix

import seaborn as sns
conf_mat = confusion_matrix(Test_Y, predictions_SVM)
class_label = ["negative", "positive"]
test = pd.DataFrame(conf_mat, index = class_label, columns =
class_label)
sns.heatmap(test, annot = True,fmt="d")
plt.title("Confusion Matrix for test data")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()



# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM,
Test_Y)*100)


SVM_RBF = svm.SVC(kernel = 'rbf', random_state = 0, gamma =
'scale')
SVM_RBF.fit(Train_X_Tfidf,Train_Y)

# predict the labels on validation dataset
predictions_SVM_RBF = SVM_RBF.predict(Test_X_Tfidf)

print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM_RBF,
Test_Y)*100)


SVM_poly = svm.SVC(C=1.0, kernel='poly', gamma = 'scale')
SVM_poly.fit(Train_X_Tfidf,Train_Y)

predictions_SVM_poly = SVM_poly.predict(Test_X_Tfidf)

print("SVM Accuracy Score ->
",accuracy_score(predictions_SVM_poly, Test_Y)*100)
```

```
SVM_sig = svm.SVC(C=1.0, kernel='sigmoid', gamma = 'scale')
SVM_sig.fit(Train_X_Tfidf,Train_Y)

predictions_SVM_sig = SVM_sig.predict(Test_X_Tfidf)

print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM_sig,
Test_Y)*100)


# # Visualisasi


DreameNeg = finalDreame.loc[finalDreame['label']== 0]
DreameNeg.head()


import difflib

#body_list = df['body'].tolist()
neg_text_list = DreameNeg['Review'].tolist()

#body = body_list
reviews_neg = neg_text_list

# filter out stop words
# these are the most common words such as: "the", "a", and "is".
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text_neg = str(neg_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens_neg = word_tokenize(text_neg)


# filter out stop words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
negative = [w for w in tokens_neg if not w in stop_words]
extend = "dont", "cant"
negative = [w for w in negative if not w in extend]
print(negative[:100])


negative = [nltk.stem.WordNetLemmatizer().lemmatize(word) for word
in negative]


import string
table = str.maketrans('', '', string.punctuation)
strippedneg = [w.translate(table) for w in negative]
```

```
#Let’s again plot the most frequent words and see if the more
significant words have come out.
freq_words(strippedneg, 20)


from wordcloud import WordCloud
from wordcloud import ImageColorGenerator

# negative
all_text_negative = ' '.join(str(word) for word in strippedneg)
wordcloud = WordCloud(max_font_size=260, max_words=100,
width=1000, height=1000, mode='RGBA',
background_color='white').generate(all_text_negative)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()


DreamePos = finalDreame.loc[finalDreame['label']==1]
DreamePos.head()


#body_list = df['body'].tolist()
pos_text_list = DreamePos['Review'].tolist()

#body = body_list
reviews_pos = pos_text_list


from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')
print(len(english_stopwords))
text_pos = str(pos_text_list)


# split into words
from nltk.tokenize import word_tokenize
tokens_pos = word_tokenize(text_pos)


# filter out stop words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
positive = [w for w in tokens_pos if not w in stop_words]
positive = [w for w in positive if not w in extend]
print(positive[:100])


positive = [nltk.stem.WordNetLemmatizer().lemmatize(word) for word
in positive]
print(positive[:100])


import string
```

```
table = str.maketrans('', '', string.punctuation)
strippedpos = [w.translate(table) for w in positive]


#Let’s again plot the most frequent words and see if the more
significant words have come out.
freq_words(strippedpos, 20)


from wordcloud import WordCloud
from wordcloud import ImageColorGenerator

# positive
all_text_positive = ' '.join(str(word) for word in strippedpos)
wordcloud = WordCloud(max_font_size=260, max_words=100,
width=1000, height=1000, mode='RGBA',
background_color='white').generate(all_text_positive)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```

Lampiran 5 – Script Asosiasi Review Wattpad Positif

```
# grab the comments
corpus <- read.csv('F:/file ta/Wattpad_Pos.csv')
# generate an array
corpus <- unlist(corpus)
library(tm)
# build a corpus, and specify the source to be character vectors
c <- Corpus(VectorSource(corpus))
# convert to lower case
c <- tm_map(c, content_transformer(tolower))
# to check it: str(c)
# remove punctuation
c <- tm_map(c, removePunctuation)
# remove numbers
c <- tm_map(c, removeNumbers)
# stopWords
custom_stopwords <- c(stopwords('english'), "é")
c <- tm_map(c, removeWords, custom_stopwords)
# check what we've done so far
str(c[2])
c <- tm_map(c, stripWhitespace)
#save a copy for later
corpus_copy <- c
#stemming text
c <- tm_map(c, stemDocument, language = "english")
# tf-idf
tfidf <- TermDocumentMatrix(corpus_copy, control = list(wordLengths = c(1, Inf)))
```

```
# inspect frequent words
freq_terms <- findFreqTerms(tfidf, lowfreq = 5)
term_freq <- rowSums(as.matrix(tfidf))
term_freq <- subset(term_freq, term_freq >= 5)
df <- data.frame(term = names(term_freq), freq = term_freq)
ab("Count") + coord_flip()
findAssocs(tfidf, "read", 0.2)
findAssocs(tfidf, "pay", 0.2)
findAssocs(tfidf, "offline", 0.2)
findAssocs(tfidf, "application", 0.2)
```

Lampiran 6 - Script Asosiasi Review Wattpad Negatif

```
# grab the comments
corpus1 <- read.csv('F:/file ta/Wattpad_Neg.csv')
# generate an array
corpus1 <- unlist(corpus1)
library(tm)
# build a corpus, and specify the source to be character vectors
c1 <- Corpus(VectorSource(corpus1))
# convert to lower case
c1 <- tm_map(c1, content_transformer(tolower))
# to check it: str(c)
# remove punctuation
c1 <- tm_map(c1, removePunctuation)
# remove numbers
c1 <- tm_map(c1, removeNumbers)
# stopWords
custom_stopwords <- c(stopwords('english'), "é")
c1 <- tm_map(c1, removeWords, custom_stopwords)
# check what we've done so far
str(c1[2])
c1 <- tm_map(c1, stripWhitespace)
#save a copy for later
corpus_copy1 <- c1
#stemming text
c1 <- tm_map(c1, stemDocument, language = "english")
# tf-idf
tfidf1 <- TermDocumentMatrix(corpus_copy1, control = list(wordLengths = c(1, Inf)))
```

```
# inspect frequent words
freq_terms1 <- findFreqTerms(tfidf1, lowfreq = 5)
term_freq1 <- rowSums(as.matrix(tfidf1))
term_freq1 <- subset(term_freq1, term_freq1 >= 5)
df1 <- data.frame(term = names(term_freq1), freq = term_freq1)
findAssocs(tfidf1, "story", 0.2)
findAssocs(tfidf1, "application", 0.2)
findAssocs(tfidf1, "new", 0.2)
findAssocs(tfidf1, "pay", 0.2)
findAssocs(tfidf1, "time", 0.2)
findAssocs(tfidf1, "library", 0.2)
findAssocs(tfidf1, "update", 0.2)
findAssocs(tfidf1, "book", 0.2)
```

Lampiran 7 – Script Asosiasi Review Dreame Positif

```
# grab the comments
corpus2 <- read.csv('F:/file ta/Dreame_Pos.csv')
# generate an array
corpus2 <- unlist(corpus2)
library(tm)
# build a corpus, and specify the source to be character vectors
c2 <- Corpus(VectorSource(corpus2))
# convert to lower case
c2 <- tm_map(c2, content_transformer(tolower))
# to check it: str(c)
# remove punctuation
c2 <- tm_map(c2, removePunctuation)
# remove numbers
c2 <- tm_map(c2, removeNumbers)
# stopWords
custom_stopwords <- c(stopwords('english'), "é")
c2 <- tm_map(c2, removeWords, custom_stopwords)
# check what we've done so far
str(c2[2])
c2 <- tm_map(c2, stripWhitespace)
#save a copy for later
corpus_copy2 <- c2
#stemming text
c2 <- tm_map(c2, stemDocument, language = "english")
# tf-idf
tfidf2 <- TermDocumentMatrix(corpus_copy2, control = list(wordLengths = c(1, Inf)))
```

```
# inspect frequent words
freq_terms2 <- findFreqTerms(tfidf2, lowfreq = 5)
term_freq2 <- rowSums(as.matrix(tfidf2))
term_freq2 <- subset(term_freq2, term_freq2 >= 5)
df2 <- data.frame(term = names(term_freq2), freq = term_freq2)
findAssocs(tfidf2, "coin", 0.2)
findAssocs(tfidf2, "chapter", 0.2)
findAssocs(tfidf2, "read", 0.2)
findAssocs(tfidf2, "book", 0.2)
findAssocs(tfidf2, "buy", 0.2)
findAssocs(tfidf2, "wait", 0.2)
```

Lampiran 8 – Script Asosiasi Review Dreame Negatif

```
# grab the comments
corpus3 <- read.csv('F:/file ta/Dreame_Neg.csv')
# generate an array
corpus3 <- unlist(corpus3)
library(tm)
# build a corpus, and specify the source to be character vectors
c3 <- Corpus(VectorSource(corpus3))
# convert to lower case
c3 <- tm_map(c3, content_transformer(tolower))
# to check it: str(c)
# remove punctuation
c3 <- tm_map(c3, removePunctuation)
# remove numbers
c3 <- tm_map(c3, removeNumbers)
# stopWords
custom_stopwords <- c(stopwords('english'), "é")
c3 <- tm_map(c3, removeWords, custom_stopwords)
# check what we've done so far
str(c3[2])
c3 <- tm_map(c3, stripWhitespace)
#save a copy for later
corpus_copy3 <- c3
#stemming text
c3 <- tm_map(c3, stemDocument, language = "english")
# tf-idf
tfidf3 <- TermDocumentMatrix(corpus_copy3, control = list(wordLengths = c(1, Inf)))
```

```
# inspect frequent words
freq_terms3 <- findFreqTerms(tfidf3, lowfreq = 5)
term_freq3 <- rowSums(as.matrix(tfidf3))
term_freq3 <- subset(term_freq3, term_freq3 >= 5)
df3 <- data.frame(term = names(term_freq3), freq = term_freq3)
findAssocs(tfidf3, "coin", 0.2)
findAssocs(tfidf3, "chapter", 0.2)
findAssocs(tfidf3, "book", 0.2)
findAssocs(tfidf3, "buy", 0.2)
findAssocs(tfidf3, "wait", 0.2)
findAssocs(tfidf3, "app", 0.2)
findAssocs(tfidf3, "day", 0.2)
findAssocs(tfidf3, "next", 0.2)
```