BAB II

LANDASAN TEORI

2.1 Sentimen Analisis

Sentimen analisis atau *opinion mining* mengacu pada bidang yang luas dari pengolahan bahasa alami, komputasi linguistik dan *text mining* yang bertujuan menganalisis pendapat, sentimen, evaluasi, sikap, penilaian dan emosi seseorang apakah pembicara atau penulis berkenaan dengan suatu topik, produk, layanan, organisasi, individu, ataupun kegiatan tertentu (Lusiani dkk, 2006).

Bing (2012) mengatakan, sentimen analisis adalah permasalahan *natural language processing*. Dia menyentuh semua aspek dari *natural language processing* seperti *coreference resolution, negation handling,* dan *word sense disambiguation.* Selain itu, sentimen analisis adalah permasalahan *natural language processing* yang sangat terbatas karena sistem tidak perlu sepenuhnya mengerti *semantics* dari setiap kalimat atau dokumen tetapi hanya perlu mengerti beberapa aspek dari hal tersebut, sebagai contoh, penilaian positif dan negatif dan target entitas atau topik. Dalam sentimen analisis, teks dikelompokkan ke dalam kategori seperti “positif”, “negatif”, atau “netral”. Pada penelitian ini, label yang digunakan adalah label “positif”, “negatif”, dan “netral”.

2.2 Natural Language

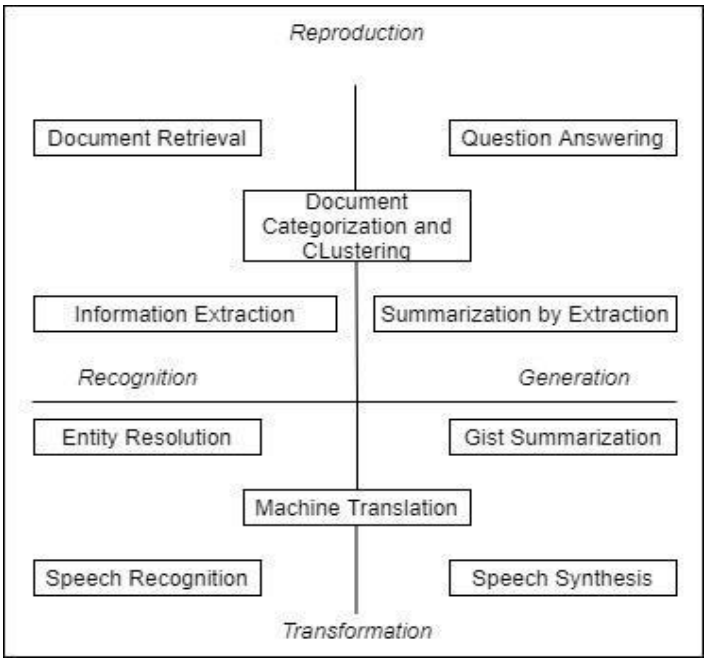
Ela (2011) berpendapat bahwa *natural language* adalah bahasa yang secara natural berevolusi dan digunakan manusia dengan tujuan berkomunikasi, sebagai contoh, Bahasa Inggris, Bahasa Perancis, Bahasa Jerman adalah *natural language*.

Natural Language Processing atau NLP (dapat juga disebut *Computational Linguistics*) adalah studi ilmiah mengenai bahasa dilihat dari perspektif komputasi.

Dalam buku yang ditulis oleh Jackson, dikatakan bahwa ada permasalahan mendasar yang tidak dapat dihindari dan perlu diperhatikan yaitu:

1. Bagaimana sistem pencarian dapat memuaskan kebutuhan pengguna.
2. Kapan sebuah program dapat diandalkan dalam menekstrak hal penting dari sebuah teks bebas.
3. Bagaimana cara mengevaluasi sebuah program *text classification* secara otomatis.
4. Apa yang membuat sebuah ringkasan yang baik dari sebuah dokumen.

Aplikasi pada *natural language* menurut Jackson (2017) dapat dilihat pada Gambar 2.1.



Gambar 2.1 Aplikasi pada *Natural Language* (Jackson, 2017)

2.3 Preprocessing

Dokumen yang perlu dilakukan pada preprocessing yaitu *query, corpus* Wikipedia Bahasa Indonesia, dokumen *Frequently Asked Questions* (FAQ). Tahap pertama yang dilakukan yaitu *case folding*, bertujuan mengubah semua huruf dalam dokumen menjadi huruf kecil dan karakter selain huruf akan dihilangkan. Tahap kedua adalah *tokenization* dilakukan pemisahan tersebut berdasarkan spasi sebagai *delimiter*. Tahapan selanjutnya dilakukan *stopword removal*, yaitu menghilangkan kata-kata tidak penting seperti "di", "yang", "ke" dan lain-lain. Dalam *stopword removal* juga ditambahkan beberapa *term* yang harus dieliminasi. Penambahan ini dilakukan secara manual dengan menambahkan *term* tersebut pada dokumen koleksi *term* yang harus dieliminasi (Tanuwijaya dkk, 2019).

2.4 Word Embedding

Word embedding mengenali distribusi makna kata serupa yang kemudian dikenali pada sebuah *model vector* (Senel dkk, 2018). Dengan menangkap karakteristik kata-kata, baik itu kata aslinya maupun kata yang mirip, perlu dihitung kemiripan kata yang satu dengan kata yang lain. Dengan menggunakan rumus *cosine similarity*, sistem dapat mengenali kemiripan antarkata pada sebuah vektor. *Word embedding* biasanya dipakai dalam tahap melakukan proses *deep learning* sebuah informasi (Young dkk, 2018).

Menurut Ahmad H. Abdullah (2018), komputer bisa mempelajari suatu karakter dari data melalui *feature extraction*. Beragam jenis *feature* diambil dari *dataset*, lalu komputer mempelajari *feature* tersebut. Dalam penelitian ini, *feature* yang akan diekstraksi adalah *word similarity* atau kemiripan makna pada suatu kata.

Word embedding berfungsi mengkonversi sebuah teks menjadi angka yang dapat diolah komputer karena komputer tidak bisa mengolah data selain dalam bentuk angka. Hasil dari konversi menjadi angka tersebut disebut dengan vektor. Metode *word embedding* mempelajari representasi vektor dari kosakata yang konstan yang berasal dari kumpulan teks.

2.5 FastText

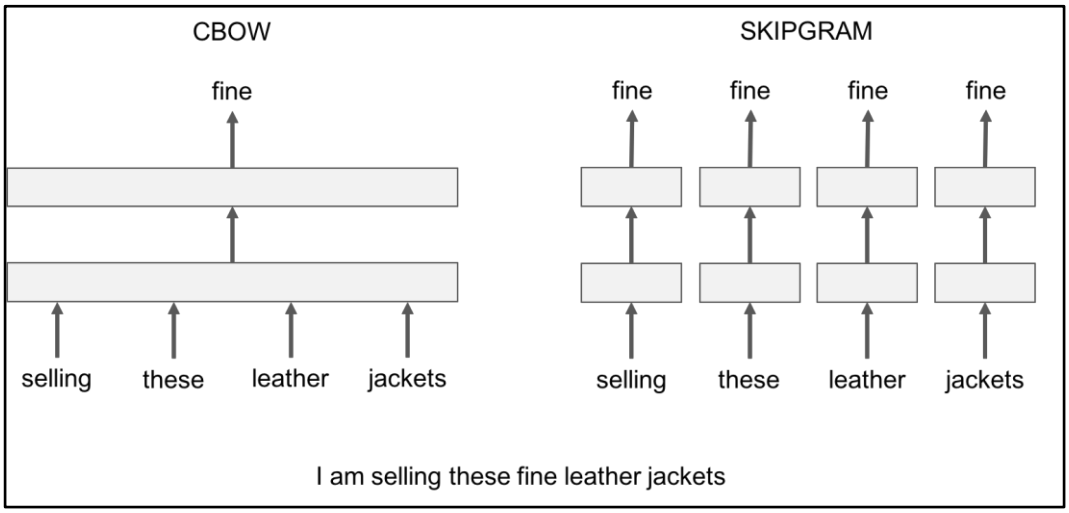
FastText merupakan *library* dari Facebook untuk representasi dan klasifikasi teks. FastText dapat melatih model bahasa yang berbeda seperti *skipgram* atau CBOW dan menerapkan berbagai parameter seperti fungsi pengambilan sampel atau *loss functions*. FastText merupakan modifikasi dari algoritma Word2Vector yang mana simbol khusus "dan" ditambahkan pada batas kata, yang membantu untuk membedakan awalan dan akhiran dari urutan karakter lainnya. Selain itu, kata itu sendiri juga telah dimasukkan dalam *set* n-gram untuk mempelajari representasi untuk setiap kata (Bersama dengan karakter n-gram). Kata yang dihasilkan oleh FastText dapat dianggap sebagai kumpulan kata yang berkelanjutan (Bojanowski dkk, 2017). Yang membuat FastText berbeda dari Word2Vector adalah informasi subkata, dan fungsi penilaian s ditunjukkan pada persamaan (2.1).

$$s(w, c) = \sum g \in Gw \; zgTvc \qquad \ldots(2.1)$$

Dimana G adalah ukuran n-gram, GW berkisar dari 1 hingga G, w adalah kata yang diberikan, Zg adalah representasi vektor untuk setiap n-gram g, VC adalah vektor konteks. Modifikasi sederhana ini memungkinkan representasi kata yang

obyektif, sehingga membantu model mempelajari representasi kata yang dapat dipercaya.

FastText memiliki dua jenis arsitektur, yaitu *Continuous Bag of Words* (CBOW) dan Skip-Gram. CBOW memprediksi kata target sesuai dengan konteksnya. Konteksnya direpresentasikan sebagai *Bag of Words* yang berada pada sekitar kata target. Sedangkan Skip-Gram belajar untuk memprediksi kata target berkat kata terdekat. Pada Gambar 2.2 terdapat visualisasi dari arsitektur CBOW dan Skip-Gram.



Gambar 2.2 Arsitektur CBOW dan Skip-Gram (FastText, 2020)

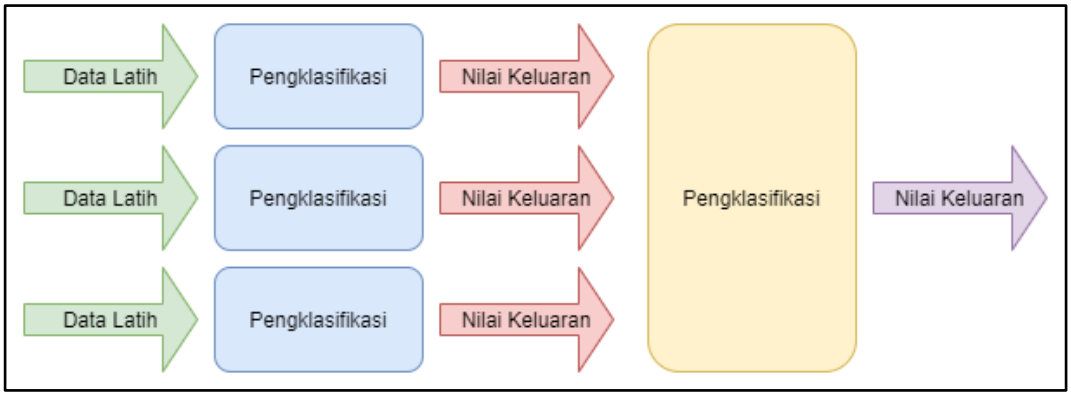
2.6 Text Classification

Menurut Nedjah (2009), *text classification* adalah sebuah pekerjaan penting yang digunakan pada banyak aplikasi. Hal ini dapat dilakukan dengan memberikan label pada dokumen berdasarkan kumpulan dokumen sebelumnya yang sudah diberi label. Langkah pertama adalah merepresentasikan dokumen sebagai vektor dalam sebuah *high-dimension vector*, dimana masing-masing dimensi menyesuaikan kepada nilai sebuah fitur, atau secara spesifik, sebuah *term*. Program

kemudian menggunakan vektor tersebut untuk mengklasifikasikan dokumen ke kategori yang tepat. Untuk melakukan hal itu sangatlah penting untuk menentukan relevansi setiap *term*, atau dikenal sebagai *term weighting*.

2.7 Teknik Ensemble

Teknik *Ensemble* merupakan metode algoritma pembelajaran yang dibangun dari beberapa model pengklasifikasi atau memprediksi untuk selanjutnya digunakan untuk mengklasifiikasi data baru berdasarkan bobot prediksi yang dihasilkan sebelumnya (Dietterich, 2000). Secara umum proses *ensemble* tergambar pada ilustrasi Gambar 2.3. Proses pengklasifikasian dilakukan dengan mengkombinasikan dengan cara-cara tertentu (umumnya berdasarkan bobot atau *voting*). Metode ini seudah sejak lama dikembangkan dan saat ini terus diupayakan untuk dihasilkan metode *ensemble* yang lebih baik. Ini dikarenakan metode *ensemble* dianggap dapat menghasilkan akurasi yang lebih baik dibandingkan pengunaan hanya satu pengklasifikasi (Syahrani, 2019).



Gambar 2.3 Ilustrasi proses *ensemble* dengan beberapa pengklalsifikasi

Untuk mendapatkan gabungan model terbaik, metode *ensemble* memungkinkan untuk menggunakan beberapa model pengklasifikasi yang berbeda jenis. Pemilihan metode pengklasifikasi dapat ditentukan berdasarkan kebutuhan dan jenis data yang akan diolah. Teknik *ensemble* yang umum digunakan adalah *boosting* dan *bagging.*

2.8 Boosting

Konsep *ensemble* dengan *boosting* bekerja dengan cara melatih kelompok model secara sekuensial dan kemudian menggabungkan keseluruhan model tersebut untuk melakukan prediksi, model yang dihasilkan belajar dari kesalahan model sebelumnya (Zhou, 2012). Tiap iterasi dihasilkan model hasil dari pembobotan proses sebelumnya. *Boosting* fokus pada proses belajar baru pada data dengan nilai akurasi rendah dari proses sebelumnya dan dilakukan dnegan proses latih secara sekuensial. Data yang salah dari prediksi sebelumnya dikelompokkan sebagai data "sulit" dan akan digunakan untuk proses prediksi berikutnya sehingga nilai kesesuaian mencapai titik maksimal. Setelah keseluruhan proses prediksi dilakukan, maka selanjutnya dilakukan penggabungan keseluruhan model. *Boosting* mengubah model prediksi yang lemah menjadi predictor kompleks yang handal. Tahapan dari proses belajar ini adalah memprediksi untuk regresi kemudian dilakukan kalulasi atas kesalahan dari residu dan terakhir proses belajar untuk mengolah residu (Syahrani, 2019).

2.9 Boosting

Bagging atau *bootstrap aggregating* adalah metode dari *machine learning* yang dibangun secara *ensemble* untuk stabilitas dan nilai akurasi yang baik dalam klasifikasi dan regresi. Konsep *ensemble* dengan *bagging* dilakukan dengan menggabungkan banyak nilai dugaan menjadi suatu nilai dugaan. Dengan menggunakan sampel berupa *bootstrap* untuk menghasilkan sampel-sampel data secara acak yang akan melakukan mekanisme *vote* untuk digabungkan sebagai pengklasifikasi utama dari model akhir yang dihasilkan (Zhou, 2012).

Mekanismenya adalah dengan membuat sampel data D berukuran n, kemudian diproduksi data latih baru sebanyak m dimana tiap *set* berukuran n berdasarkan pengambilan acak dari data D serta dilakukan penggantian pada data isinya (*replacement*). Pengklasifikasi dibuat berdasarkan sampel-sampel m tersebut. Tiap samepl memiliki probabilitas untuk terpilih sebagai data uji (Syahrani, 2019).

2.10 Extreme Gradient Boosting (XGBoost)

XGBoost merupakan kombinasi antara *Gradient Descent* dan *Boosting* yang disebut *Gradient Boosting Machine* (GBM). *Boosting* adalah algoritma *Ensemble Learning* yang memberikan bobot yang berbeda untuk distribusi data pelatihan pada setiap iterasi. Setiap peningkatan iterasi, ditambahkan bobot untuk sampel kesalahan klasifikasi yang salah dan mengurangi bobot untuk sampel yang diklasifikasikan benar, sehingga mengubah distribusi data pelatihan secara efektif (Bisri dan Wahono, 2015). GBM menggunakan statistik *Gradient* orde kedua untuk

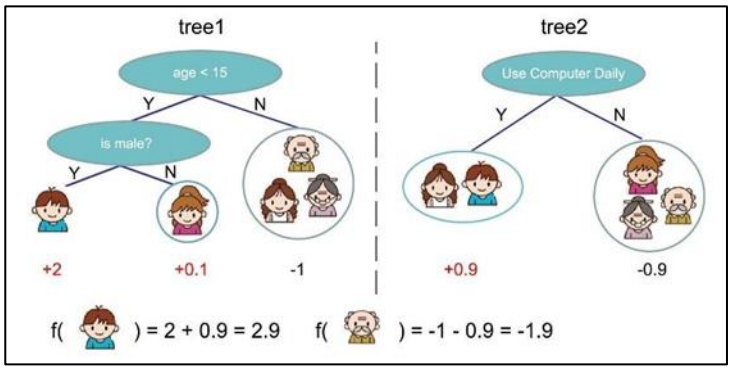
meminimalkan tujuan yang diatur yang ditunjukkan pada persamaan (2.2) dan (2.3).

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad \text{...(2.2)}$$

$$\text{Dimana } \Omega(f) = \gamma T + \frac{1}{2}\lambda |w|^2 \quad \text{...(2.3)}$$

Dengan l adalah fungsi yang mengukur perbedaan antara prediksi $\hat{y}_i$ dan target y_i, dan regulasi Ω untuk menghindari *overfitting* model (Chen dan Guestrin, 2016).

Extreme Gradient Boosting (XGBoost) dikembangkan oleh Chen dan Guestrin (2016). XGBoost merupakan varian dari algoritma *Tree Gradient Boosting*. Algoritma ini merupakan interpretasi dari metode *Newton Boosting* karena kesamaan dalam konsep algoritmanya. Optimasi yang dilakukan oleh algoritma XGBoost lebih cepat 10 kali dibandingkan dengan implementasi Gradient Boosting lainnya (Chen dan Guestrin, 2016).



Gambar 2.4 *Model Ensemble Tree* (Chen dan Guestrin, 2016)

XGBoost memiliki parameter tambahan dibandingkan *Gradient Boosting* lain. Parameter tersebut berupa nilai pinalti pada tiap *tree*. Hal ini terlihat pada Gambar 2.4 yang mana pinalti tersebut memberikan pengaruh terhadap nilai pada

struktur pohon baru dan memberikan bobot dari pencabangan yang bertujuan untuk mengurangi variasi pada setiap pohon. Ilustrasi pada Gambar 2.4 menjelaskan mekanisme XGBoost dalam mengklasifikasi minat terhadap permainan *game online* dengan cara memberikan bobot untuk tiap fitur di setiap *tree* yang ditumbuhkan. Bobot nilai negatif atau pinalti juga diberikan untuk fitur yang berada pada *leaf* pencabangan dengan nilai *false*. Keseluruhan bobot dari seluruh *tree* akan diakumulasi untuk mendapatkan gabungan model yang akan menghasilkan tingkat akurasi yang baik. XGBoost juga memberikan nilai parameter acak yang berfungsi untuk menihilkan korelasi pada pohon tertentu sehingga juga dapat mengurangi variasi secara keseluruhan (Nielsen, 2016).

2.11 Media Sosial

Perkembangan media yang begitu pesat, memunculkan banyak media online dari media berita sampai media sosial. Dalam artikel yang ditulis oleh Tea (2014), media sosial adalah saluran atau sarana pergaulan sosial secara online di dunia maya. Para pengguna media sosial berkomunikasi, berinteraksi, saling kirim pesan, dan saling berbagi. Media sosial saja sudah begitu banyak, dari Facebook, Twitter, Path, Instagram, Google+, Tumblr, Linkedin dan sebagainya masih banyak lagi (Evadollzz, 2014). Media sosial sekarang ini tidak hanya digunakan sebagai sarana pertemanan, mencari teman, tapi sudah banyak digunakan untuk kegiatan lain. Promo dagangan, jual beli apa saja sampai promo partai politik atau kampanye calon-calon legislatif dan presiden (Buntoro, 2017).

2.12 Facebook

Facebook adalah situs *web* media sosial paling populer. Facebook adalah *framework*, berita *online* dan komunikasi orang ke orang dimana pengguna memposting dan berinteraksi dengan pesan, yang dikenal sebagai "komentar". Pesan-pesan ini awalnya hanya dibatasi hingga 140 karakter (Kaur dkk, 2019). Menurut KOMINFO (2019), pengguna Internet Indonesia mencapai 150 juta orang. Dengan jumlah pengguna yang sangat besar, Facebook memiliki 130 juta pengguna (Statista, 2020) sehingga memiliki banyak sekali data yang secara implisit dapat digali lebih lanjut dengan berbagai metode *data mining*.

2.13 Evaluasi Performa

Metode yang digunakan dalam menilai performa dari model adalah *confussion matrix* (CM). CM memiliki empat komponen yang mempresentasikan hasil klasifikasi yaitu *Ture Positive (TP), True Negative (TN), False Positive (FP),* dan *False Negative (FN).* TN berupa jumlah data negatif yang bernilai benar, FP merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, TP merupakan data positif yang terdeteksi benar. FN merupakan kebalikan dari TP, sehingga datanya bernilai positif, namun terdeteksi sebagai data negatif (Syahrani, 2019).

Berdasarkan nilai-nilai tersebut dapat diperoleh nilai *accuracy, precision,* dan *recall*. Nilai *accuracy* menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara benar. Dengan kata lain, nilai *accuracy* merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Nilai *accuracy* dapat diperoleh dengan Persamaan 2.4. Nilai *precision* menggambarkan

jumlah data kategori positif yang diklasifikasikan secara benar dibagi dengan total data yang diklasifikasi positif. *Precision* dapat diperoleh dengan Persamaan 2.5. Sementara itu, *recall* menunjukkan berapa persen data kategori positif yang terklasifikasikan dengan benar oleh sistem. Nilai *recall* diperoleh dengan Persamaan 2.6. F1 *score* merupakan perbandingan rata-rata *precision* dan *recall.* Nilai F1 diperoleh dengan Persamaan 2.7.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \ldots(2.4)$$

$$Precision = \frac{TP}{(TP + FP)} \quad \ldots(2.5)$$

$$Recall = \frac{TP}{(TP + FN)} \quad \ldots(2.6)$$

$$F1 = \frac{(2\ x\ Precision\ x\ Recall)}{(Precision + Recall)} \quad \ldots(2.7)$$

Klasifikasi dengan jumlah kategorri lebih dari dua (*multi-class*), penghitungan *accuracy, precision,* dan *recall* dilakukan dengan menghitung rerata nilai *accuracy, precision,* dan *recall* pada setiap kategori. Persamaan 2.8, 2.9, dan 2.10 merupakan formula untuk menghitung nilai *accuracy, precision,* dan *recall* dari sistem klasifikasi *multi-class.*

$$Accuracy = \frac{\sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}}{l} \quad \ldots(2.8)$$

$$Precision = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} (FP_i + TP_{i+})} \quad \ldots(2.9)$$

$$Recall = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} (FN_i + TP_{i+})} \quad \ldots(2.10)$$

TP_i adalah *Ture Positive,* yaitu jumlah data positif yang terklasifikasi dengan benar oleh sistem untuk kelas ke-i. TN_i adalah *True Negative*, yaotu jumlah data negatif yang terklasifikasi dengan benar oleh sistem untuk kelas ke-i. FN_i

adalah *False Negative*, yaitu jumlah data negatif namun terklasifikasi salah oleh sistem untuk kelas ke-i. FP_i adalah *False Positive,* yaitu jumlah data positif namun terklasifikasi salah oleh sistem untuk kelas ke-i, l adalah jumlah kelas (Syahrani, 2019).