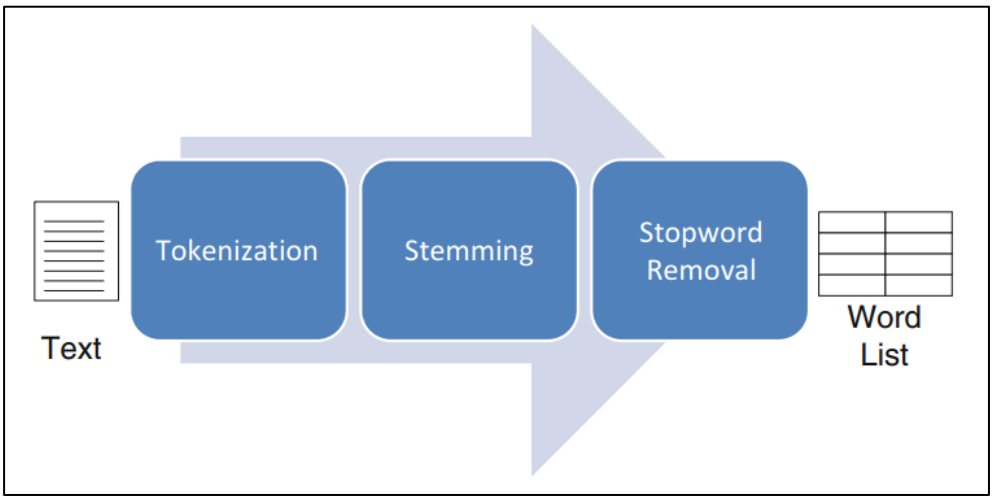
BAB II

LANDASAN TEORI

2.1 Kategorisasi Teks

Kategorisasi teks merupakan sebuah proses untuk memberikan satu atau lebih kategori yang sudah ditetapkan ke tiap-tiap teks tersebut (Jo, 2019: 79). Salah satu contoh implementasi nyata yang sering dijumpai dalam dunia teknologi informasi adalah sistem *spam detection* pada surat elektronik yang secara otomatis mampu memberikan kategori kepada surel yang masuk, yaitu *spam* atau bukan (Ignatow dan Mihalcea, 2016: 117). Manfaat utama dalam melakukan kategorisasi teks adalah untuk memperdalam pengertian terhadap suatu teks dan membuat kesimpulan dari ciri suatu entitas (Zhai dan Massung, 2016: 299-300).

Sebelum kategorisasi dilakukan, mula-mula teks diproses dengan *text indexing*. *Text indexing* mengubah teks yang ada menjadi daftar kata-kata, sehingga menjadi data yang terstruktur dan dapat dikomputasikan (Jo, 2019: 19). Menurut Jo (2019), tiga tahapan dasar dari *text indexing* diilustrasikan seperti Gambar 2.1 berikut.



Gambar 2.1 Tiga tahapan dari *Text Indexing* (Jo, 2019)

Proses *tokenization* mengidentifikasi kata-kata dari urutan input karakter dan membagi-bagikannya berdasarkan tanda baca atau spasi ke dalam *token* dengan tetap memperhatikan beberapa hal, contohnya kontraksi, singkatan, dan seterusnya (Jo, 2019; Ignatow dan Mihalcea, 2016). Dalam proses *tokenization* juga dilakukan penghilangan karakter atau simbol khusus dan mengubah semua karakter menjadi huruf bukan kapital (Jo, 2019).

Proses *stemming* mengonversi tiap *token* yang sudah dibuat pada proses sebelumnya menjadi menjadi bentuk dasarnya (Jo, 2019). Contoh algoritma untuk proses *stemming* dalam Bahasa Indonesia adalah algoritma Nazief dan Adriani. Berdasarkan penelitian yang dilakukan oleh Wahyudi, dkk (2017), algoritma Nazief dan Adriani menghasilkan akurasi tinggi yaitu di atas 95%.

Proses selanjutnya dalam tahapan *text indexing* adalah *stopwords removal*. Proses ini menghilangkan *stopwords* dari daftar *token* atau kata-kata yang sudah di-*stemming* (Jo, 2019). *Stopwords* merujuk kepada kata-kata yang sering muncul dalam sebuah teks, bisa berbentuk kata ganti orang, preposisi, dan lain-lain (Ignatow dan Mihalcea, 2016). Kata-kata tersebut tidak mempunyai informasi yang berarti (Zhai dan Massung, 2016: 66).

Setelah proses awal dilakukan, kategorisasi teks dapat dilakukan dengan menggunakan pendekatan *supervised learning* dalam pembelajaran mesin. *Supervised learning* digunakan ketika ingin memprediksi suatu hasil dari masukkan yang diberikan, dan ada contoh pasangan *input/ouput*-nya (Müller dan Guido, 2017). Müller dan Guido (2017) menjelaskan bahwa salah satu contoh dari tipe permasalahan *supervised learning* adalah klasifikasi yang tujuan utamanya adalah memprediksi *class label* yang sudah ditentukan.

2.2 Algoritma Naive Bayes

Nama Bayes diambil dari seorang ilmuwan asal Inggris, Thomas Bayes. Algoritma Naive Bayes menggunakan teorema Bayes yang berasumsi naif bahwa tiap fitur adalah independen untuk setiap kelas (Tang, 2016). Persamaan dasar dari teorema Bayes yang diambil dari Wu, dkk. (2018), adalah sebagai berikut.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \quad \text{...(2.1)}$$

Dengan $P(A)$ adalah probabilitas *prior* (kelas) dan $P(A|B)$ adalah probabilitas *posterior* (kondisional). Sebagai contoh, untuk mengklasifikasi dokumen D ke dalam kelas C, sebagai kelas yang mempunyai probabilitas *posterior* $P(C|D)$ tertinggi, dapat ditulis ulang berdasarkan Persamaan (2.1) yang diambil dari Shimodaira (2019) sebagai berikut.

$$P(C|D) = \frac{P(D|C)\ P(C)}{P(D)} \propto P(D|C)\ P(C) \quad \text{...(2.2)}$$

Terdapat tiga model distribusi Naive Bayes, yaitu Bernoulli, Poisson, dan Multinomial (Tang, dkk., 2016). Dalam penelitian oleh Tang, dkk. (2016) disebutkan bahwa Multinomial Naive Bayes adalah pendekatan yang terbaik untuk melakukan klasifikasi. Model Multinomial merepresentasikan dokumen sebagai vektor yang isi nilainya merupakan frekuensi kemunculan kata di dokumen tersebut (Shimodaira, 2019). Jika terdapat kamus V yang berisi $|V|$ tipe kata, dimensi vektor fitur $D = |V|$ (Shimodaira, 2019). Probabilitas dokumen $P(D|C)$ dapat ditulis ulang dengan model Multinomial seperti persamaan berikut berdasarkan Shimodaira (2019).

$$P(D|C) = P(x|C_k) \propto \prod_{t=1}^{|V|} P(w_t|C_k) \quad \text{...(2.3)}$$

Untuk x menjadi vektor fitur Multinomial pada dokumen D. Nilai $P(w_t|C_k)$ adalah probabilitas dari frekuensi kata w_t muncul pada kelas k dibagi total kata yang muncul pada kelas k, dihitung menggunakan persamaan berikut yang berdasarkan dari Shimodaira (2019).

$$P(w_t|C_k) = \frac{n_k(w_t)}{\sum_{s=1}^{|V|} n_k(w_s)} \quad ...(2.4)$$

Nilai probabilitas untuk kelas k dapat dihitung menggunakan persamaan berikut, dikutip dari Shimodaira (2019).

$$P(C_k) = \frac{N_k}{N} \quad ...(2.5)$$

Nilai N_k adalah jumlah dokumen dari kelas k dan N adalah jumlah seluruh dokumen dalam data latih. Secara singkat, untuk data latih yang sudah diberi kategori atau kelas, tahapan Multinomial Naive Bayes yang dilakukan adalah sebagai berikut (Shimodaira, 2019).

1. Buat kamus kata V dari data latih. Jumlah kata yang terdapat di dalamnya menentukan jumlah fitur.
2. Hitung nilai berikut dalam data latih:
    a) N = total dari dokumen atau teks
    b) N_k = jumlah dokumen yang diberi kelas k, dengan k = 1,…, K
    c) $n_k(w_t)$ = frekuensi dari kata w_t dalam dokumen yang mempunyai kelas k
3. Gunakan Persamaan (2.4) untuk menentukan probabilitas $P(w_t|C_k)$
4. Gunakan Persamaan (2.5) untuk menentukan probabilitas $P(C_k)$

Untuk data uji dokumen D, dilakukan proses kategorisasi menggunakan persamaan berikut, berdasarkan dari Shimodaira (2019).

$$P(C_k|D) \propto P(C_k) \prod_{j=1}^{len(D)} P(u_j|C_k) \quad \text{...(2.6)}$$

Nilai u_j adalah kata yang ke-j dalam dokumen D. Nilai hasil perkalian $P(u_j|C_k)$ dari seluruh dokumen kemudian dikali dengan $P(C_k)$.

2.3 Laplace Smoothing

Persamaan (2.4) mempunyai kelemahan dalam mengestimasi frekuensi kemunculan suatu kata, yaitu jika nilai $n_k(w_t)$ menghasilkan nol, maka probabilitas $P(D|C)$ akan menghasilkan nol juga, yang menandakan bahwa dokumen atau teks tersebut sama sekali bukan termasuk suatu kelas (Shimodaira, 2019). Hal tersebut tidak selalu benar dikarenakan belum tentu suatu kata w_t tidak muncul pada dokumen lain pada kelas yang sama.

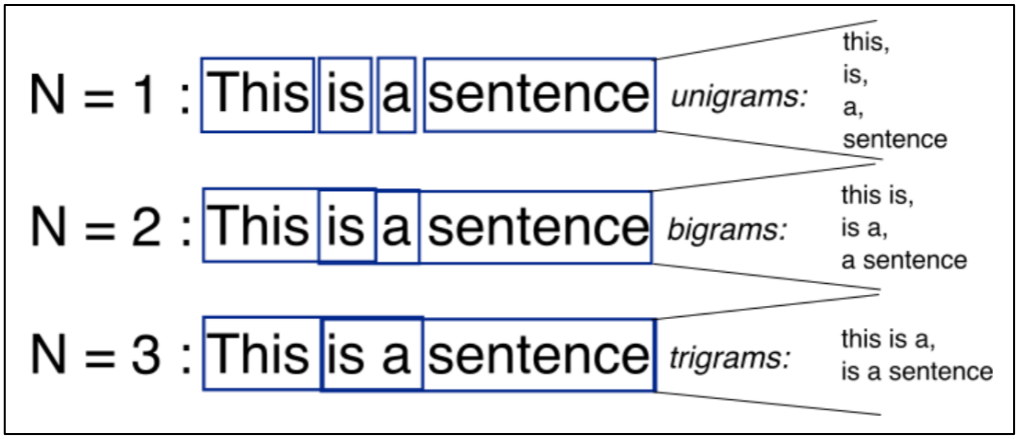
Salah satu cara untuk menyelesaikan permasalahan tersebut adalah dengan menggunakan *add-one smoothing* (Shimodaira, 2019). *Add-one smoothing* atau *Laplace smoothing* didasarkan dari *Laplace's law of succession*. Permasalahan yang terkenal adalah probabilitas matahari terbit esok hari (*sunrise problem*) (Smith, 2009). *Laplace smoothing* biasa disebut *add-one smoothing* dikarenakan pada metode ini, tiap frekuensi kata yang didapat ditambahkan satu (Listiowarni dan Setyaningsih, 2018), tujuannya untuk menghilangkan probabilitas yang menghasilkan nilai nol tadi, sehingga Persamaan (2.4) bisa ditulis ulang sebagai berikut yang berdasarkan dari Shimodaira (2019).

$$P_{Lap}(w_t|C_k) = \frac{1+ n_k(w_t)}{|V|+ \sum_{s=1}^{|V|} n_k(w_s)} \quad ...(2.7)$$

Pada Persamaan (2.7), selain penambahan angka satu, bagian penyebut dari pecahan tersebut juga ditambahkan $|V|$. Penambahan tersebut dilakukan untuk memastikan probabilitas masih dinormalisasi sebagai akibat penambahan angka satu yang menyebabkan tingginya probabilitas pada huruf yang belum pernah muncul (Shimodaira, 2019).

2.4 N-gram

N-gram secara singkat adalah kelompok kata yang berurutan sejumlah n (Jurafsky dan Martin, 2014). Huruf "N" dalam kata "N-gram" menentukan berapa nilai kelompok kata yang berurutan tersebut. Untuk satu kata disebut 1-gram atau *unigram*, untuk dua kata disebut 2-gram atau *bigram*, untuk tiga kata disebut 3-gram atau *trigram*, dan seterusnya (Pustejovsky, 2015). Ilustrasi dari N-gram bisa dilihat pada Gambar 2.2 berikut.



Gambar 2.2 Ilustrasi contoh N-gram (Chang, 2016)

N-gram juga dapat diaplikasikan di huruf pada tiap kata (Pustejovsky, 2015). *Padding* dengan karakter berupa spasi pada awal dan akhir kata biasa dilakukan untuk membantu mengambil potongan-potongan huruf tersebut (Chandra, dkk., 2016). Sebagai contoh untuk kata "bayes" dapat dibuat bentuk N-gram seperti berikut.

1-gram : b,a,y,e,s

2-gram : _b, ba, ay, ye, es, s_

3-gram : _ba, bay, aye, yes, es_

Ahmad (2009) dalam penelitian yang dilakukan oleh Sugianto, dkk. (2013) menyebutkan bahwa dengan menggunakan N-gram, keuntungan yang diperoleh adalah N-gram tidak terlalu sensitif untuk kesalahan ejaan penulisan sehingga dapat diproses dengan baik. Perbandingan antar N-gram kata akan menunjukkan seberapa mirip suatu kata dengan kata lain. Sebagai contoh untuk 2-gram dari kata "bayes" dan 2-gram dari kata "bayse" hanya ada dua perbedaan dari total enam N-gram yang terbentuk, sehingga bisa diasumsikan kata "bayse" memiliki kesamaan dengan kata "bayes".

2.5 Metrik Evaluasi

Algoritma Naive Bayes yang diimplementasikan pada aplikasi diuji dan dievaluasi untuk diketahui nilai *accuracy*, *precision*, *recall,* dan *f-measure*-nya berdasarkan buku oleh Müller dan Guido (2017). Untuk menghitung nilai-nilai tersebut, digunakan *confusion matrix*.

Confusion matrix menggambarkan jumlah data yang benar dan jumlah data yang salah saat proses kategorisasi (Indriani, 2014). *Confusion matrix* dapat dilihat seperti Tabel 2.1 berikut.

Tabel 2.1 *Confusion Matrix*

		Prediction	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Nilai *True Positive* (TP) menandakan data benar diprediksi ke kelas positif, *True Negative* (TN) menandakan data benar diprediksi ke kelas negatif. Nilai *False Negative* (FN) menandakan data salah diprediksi ke kelas negatif. Nilai *False Positive* (FP) menandakan data salah diprediksi ke kelas positif (Avati, 2013).

Accuracy memberikan nilai dari perbandingan seberapa banyak prediksi yang benar dari total keseluruhan (Avati, 2013). Persamaan untuk mencari *accuracy* bisa dilihat pada rumus berikut, dikutip dari Indriani (2014).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad ...(2.8)$$

Penghitungan lainnya untuk membantu memberikan informasi selain *accuracy* ada *precision* dan *recall* (Müller dan Guido, 2017). Nilai *precision* mengukur berapa banyak data benar positif dengan seluruh data yang terprediksi positif, sedangkan nilai *recall* mengukur berapa banyak data benar positif dengan seluruh data asli yang positif (Müller dan Guido, 2017). Persamaan untuk mencari nilai *precision* bisa dilihat pada rumus berikut berdasarkan buku oleh Müller dan Guido (2017).

$$Precision = \frac{TP}{TP+FP} \quad ...(2.9)$$

Persamaan untuk mencari nilai *recall* bisa dilihat pada rumus berikut berdasarkan buku oleh Müller dan Guido (2017).

$$Recall = \frac{TP}{TP+FN} \quad ...(2.10)$$

Untuk melihat *precision* dan *recall* secara utuh, dapat dihitung nilai *f-score* atau *f-measure*, yang merupakan rata-rata harmonis dari *precision* dan *recall* (Müller dan Guido, 2017). Persamaan untuk mencari nilai *f-measure* adalah sebagai berikut berdasarkan buku oleh Müller dan Guido (2017).

$$F\ measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad ...(2.11)$$

2.6 Technology Acceptance Model

Analisis faktor-faktor penerimaan teknologi komputer dapat diukur dengan beberapa model, salah satunya adalah *Technology Acceptance Model* (TAM). Model TAM diperkenalkan oleh Davis (1985). Tujuan menggunakan model ini adalah untuk menjelaskan dan memahami faktor-faktor perilaku pengguna terhadap diterimanya penggunaan teknologi (Loanata dan Tileng, 2016). Dalam memberikan penjelasan perilaku pengguna terhadap suatu teknologi informasi baru, model TAM dianggap sebagai model terbaik yang terbukti secara empiris menjelaskan niat dan perilaku pengguna (Uska, 2017). Model TAM didasarkan dari teori bahwa sikap dan perilaku seseorang akan ditentukan dari reaksi dan persepsi terhadap suatu hal (Wibowo, 2008).

Terdapat dua buah variabel utama yang digunakan untuk menjelaskan tingkat penerimaan pengguna tersebut, yaitu manfaat yang dirasakan (*perceived usefulness*) dan persepsi kemudahan penggunaan (*perceived ease of use*) (Uska, 2017). *Perceived usefulness* adalah ukuran sampai sejauh mana seseorang meyakini bahwa penggunaan suatu sistem teknologi informasi akan meningkatkan kinerja

dari pekerjaannya, sedangkan *perceived ease of use* adalah ukuran sampai sejauh mana seseorang meyakini bahwa dalam menggunakan suatu sistem teknologi informasi akan membantu pengguna bebas dari upaya, baik secara fisik maupun mental (Davis, 1985).

2.7 Skala Likert

Suatu sikap, pendapat, dan persepsi seseorang atau sekelompok orang dapat diukur dengan skala Likert (Gaszella dan Amin, 2018). Skala Likert biasanya terdiri dari beberapa pernyataan yang kemudian seorang responden harus memberikan nilai seberapa tingkat setuju atau tidak setujunya menggunakan beberapa pilihan jawaban, yang dimulai dari "sangat setuju" hingga "sangat tidak setuju" (Albaum, 1997). Skala tersebut bertujuan untuk mengukur kemana arah sikap seseorang dan seberapa tinggi intensitasnya (Albaum, 1997). Tiap pilihan jawaban kemudian diberi nilai bobot sesuai skala (Hamdani dan Kusdiarto, 2017). Contohnya jika memakai 7 skala dalam pilihan jawaban, pilihan "sangat tidak setuju" akan bernilai 1 dan pilihan "sangat setuju" akan bernilai 7. Penarikan kesimpulan dari skala Likert dilakukan dengan cara mengalikan jumlah responden per jawaban dengan skor dari jawaban yang dipilih (Hamdani dan Kusdiarto, 2017). Hasil yang didapatkan kemudian dijumlahkan dan dibagi dengan skor ideal, yaitu jumlah responden dikali dengan skor tertinggi (Hamdani dan Kusdiarto, 2017). Hasil akhir yang didapatkan kemudian dikali dengan 100% untuk menentukan persentase akhir (Hamdani dan Kusdiarto, 2017). Interval persentase dihitung dengan membagi angka 100 dengan skala yang digunakan (Rahardja, dkk., 2018).