BAB 2
LANDASAN TEORI

2.1 Berita Palsu

Berita palsu adalah berita yang tidak sesuai fakta dengan tujuan untuk propaganda, manipulasi, penipuan, menyindir, dan menciptakan kesalahpahaman [1]. Berita palsu memeliki kecenderungan kontroversial dan memenuhi ekpektasi pembaca [8]. Hal ini menyebabkan mudahnya penyebaran berita palsu karena menarik banyak pembaca. Berita palsu dapat menyebabkan misinformasi dan merugikan beberapa pihak yang berkaitan pada berita palsu jika sampai tersebar luas.

Berita palsu memiliki beberapa ciri yang menonjol diantaranya yaitu [9]:

1. Memiliki judul yang panjang dan menarik perhatian pembaca.
2. Penggunaan frasa kata kerja secara berlebihan pada judul.
3. Isi berita palsu penuh dengan kalimat yang menyindir.
4. Memiliki kalimat yang persuasif dan berlebih-lebihan.
5. Pembuktian kebenaran melalui permisalan heuristis bukan melalui argumen yang valid.
6. Terkadang bahasa yang digunakan tidak baku.

Ciri-ciri berita palsu diatas dapat dijadikan acuan dalam analisis perbedaan berita palsu dan berita asli. Mulai dari format penulisan berita yang harus diperhatikan yaitu cara penulisan judul, nama penulis, sentimen berita, makna kata & kalimat, dan penggunaan bahasa baku. Dari unsur-unsur berita tersebut dapat diambil ciri-ciri sebuah kata atau frasa. Misalnya pada judul dapat dianalisis panjang kalimat dan makna setiap kata. Pada isi berita dapat dianalisis makna kalimatnya, makna setiap kata, dan penggunaan bahasa yang baku.

2.2 Word Embedding Algorithm

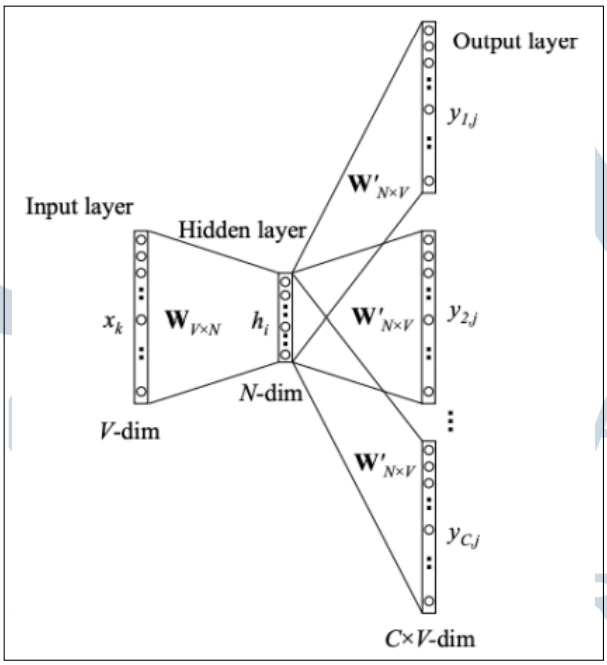
Word embedding merupakan konversi data dalam bentuk teks menjadi sebuah vektor dalam ruang multi-dimensi, dimana kata-kata yang memiliki makna

serupa akan dipetakan berdekatan [10]. *Word embedding* merupakan salah satu teknik *feature extraction* untuk data teks sehingga dapat diproses oleh algoritma klasifikasi. Berikut beberapa contoh *word embedding* yang dapat digunakan antara lain Word2Vec dan GloVe.

2.2.1 Word2Vec

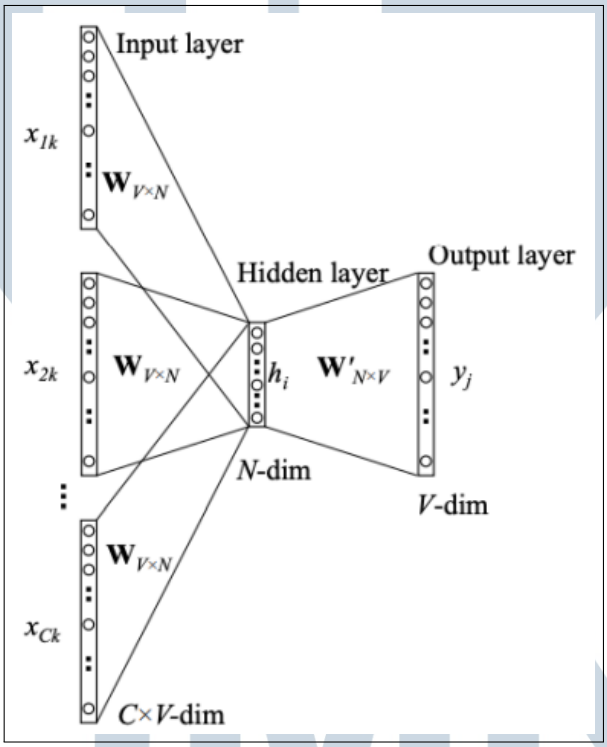
Word2Vec merupakan algoritma *word embedding* yang dikembangkan oleh Google. Cara kerja Word2Vec adalah dengan mempelajari distibusi sebuah kata dengan memprediksi kata-kata yang muncul disekitarnya dalam korpus teks [4]. Kata-kata dengan makna yang serupa cenderung muncul dalam konteks yang serupa. Word2Vec memiliki turunan algoritma yaitu *Continous Bag of Words* (CBOW) dan Skip-gram dimana keduanya termasuk algoritma *deep learning neural network*.

CBOW dan Skip-gram menggunakan fungsi aktivasi yang sama yaitu fungsi softmax. Fungsi softmax cocok untuk klasifikasi data multi kelas seperti data teks. Kesamaan lain dari kedua algoritma ini adalah arsitektur *neural network* yang terdiri dari tiga lapisan yang terhubung sepenuhnya yaitu lapisan input, lapisan tersembunyi, dan lapisan output [11].



Gambar 2.1. Model Skip-gram
Sumber: [11]

Model Skip-gram berfungsi untuk mempelajari konteks dari sebuah kata. Cara kerja Skip-Gram dapat dilihat dari arsitekur *neural networks* pada gambar 2.1. Lapisan input berupa vektor yang lalu memilih satu baris vektor pada lapisan tersembunyi sebagai vektor representasi dari sebuah kata. Kemudian vektor representasi kata tersebut dikalikan dengan setiap kolom pada lapisan output. Lalu, hasil perkalian (*dot product*) tersebut akan melalui fungsi softmax sehingga mendapatkan nilai probabilitas kata tersebut memiliki kesamaan konteks dengan kata-kata di lapisan output.



Gambar 2.2. Model CBOW
Sumber: [11]

Model CBOW berfungsi untuk mempelajari sebuah kata berdasarkan konteksnya. Cara kerja CBOW berkebalikan dengan Skip-gram seperti pada gambar 2.2, dimana vektor representasi kata-kata konteks sebagai input memilih vektor-vektor pada lapisan tersembunyi. Kemudian hasil perkalian (*dot product*) vektor-vektor tersebut dengan kolom pada lapisan keluar akan melalui fungsi softmax. Setelah itu didapatkan hasil probabilitas konteks tersebut memiliki kata tertentu sebagai pusat konteksnya.

2.2.2 GloVe

GloVe adalah model *unsupervised learning* yang sudah dilatih dengan korpus dari situs-situs web yang umum seperti Wikipedia dan Twitter. GloVe bekerja dengan memanfaatkan statistik kemunculan kata secara global dalam korpus untuk menghasilkan *word embedding*. GloVe memiliki input berupa korpus berisi data teks yang diubah menjadi *co-occurrence matrix*. GloVe memiliki output berupa *word embedding* yang merepresentasikan kata menjadi sebuah vektor dalam ruang vektor kontinu [6].

Co-occurence matrix adalah sebuah matriks yang mencatat data statistik sebuah kemunculan kata pada konteks lokal. Sebagai contoh terdapat dua kalimat yaitu ”saya suka makan nasi” dan ”saya tidak suka makan kentang”. Dari kedua kalimat tersebut dapat dibentuk sebuah *co-occurrence matrix* seperti pada tabel 2.1.

Tabel 2.1. *Co-occurrence Matrix*

	saya	suka	makan	tidak	nasi	kentang
saya	0					
suka	2	0				
makan	2	2	0			
tidak	1	1	1	0		
nasi	1	1	1	0	0	
kentang	1	1	1	1	0	0

Setelah *co-occurrence matrix* terbentuk dari korpus, maka akan dilakukan pembobotan menggunakan fungsi *Pointwise Mutual Information* (PMI). Fungsi ini akan mengukur rasio dari *co-occurrence probability* dari dua kata yang saling berhubungan [6].

$$PMI(w_1, w_2) = \log\left(\frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)}\right) \tag{2.1}$$

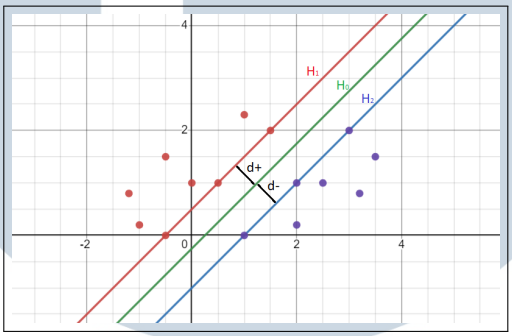
Pada persamaan 2.1, $P(w_1, w_2)$ merupakan nilai *co-occurrence probability* dari dua kata w_1 dan w_2. Lalu, $P(w_1)$ dan $P(w_2)$ merupakan *co-occurrence probability* dari masing-masing kata.

2.3 Support Vector Machine

Algoritma *Support Vector Machine* (SVM) adalah algoritma *supervised machine learning* yang menggunakan persamaan matematika sebagai pemisah

untuk melakukan klasifikasi titik-titik data. Tujuan algoritma ini adalah untuk menemukan persamaan matematika yang ideal, yaitu jarak minimum antara pemisah dan titik-titik terdekat dari kelas yang berbeda [12]. Pemisah yang terbentuk dari persamaan matematika adalah sebuah *hyperplane* yang berdasarkan letak *support vectors* [13].

Secara sederhana *Support Vector Machine* linear dua dimensi dapat didefinisikan dengan fungsi persamaan linear $y = ax + b$. Jika $x = x_1$ dan $y = x_2$ maka persamaan tersebut menjadi $ax_1 - x_2 + b = 0$. Jika x_1 dan x_2 didefinisikan sebagai titik koordinat maka $X = (x_1, x_2)$ dan $W = (a, -1)$ maka kita akan mendapatkan persamaan *hyperplane* yaitu $W \cdot X + b = 0$ [14].



Gambar 2.3. SVM Dua Dimensi

Gambar 2.3 merupakan visualisasi persamaan *hyperlane* tersebut yang terbentuk dari titik-titik yang dijadikan *support vectors*. Dapat dituliskan bahwa $H_1 : W \cdot X_1 + b = +1$ untuk titik-titik positif sedangkan $H_2 : W \cdot X_2 + b = -1$ untuk titik-titik negatif. H_0 didapatkan dari median antara H_1 dan H_2, sehingga didapatkan $H_0 : W \cdot X_0 + b = 0$. Maka dari itu, jarak terdekat antar *support vector* positif $(d+)$ dengan H_0 sama dengan jarak terdekat *support vector* negatif $(d-)$ dengan H_0.

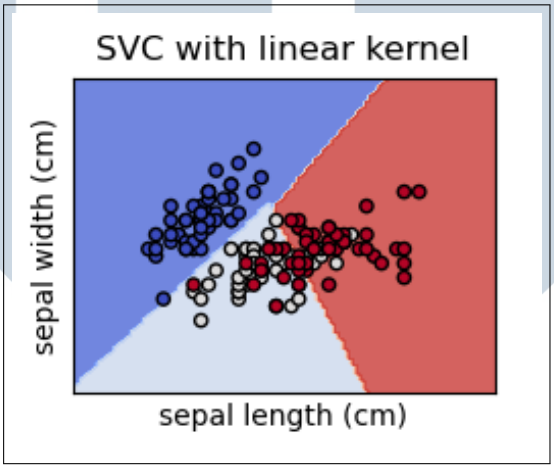
Terdapat sebuah metode agar SVM dapat memproses dataset besar dengan dimensi yang banyak yaitu dengan persamaan *kernel*. *Kernel* memungkinkan SVM untuk melakukan klasifikasi dengan mengikuti sebuah aturan persamaan matematika. Hal ini bermanfaat untuk menghindari overfitting ataupun mengoptimasi bentuk *hyperplane* pemisah sehingga menyesuaikan distribusi data. Terdapat beberapa *kernel* yang umum digunakan dalam model SVM, diantara lain *linear kernel*, *Polynomial kernel*, *radial basis function kernel* (RBF) [15].

A. Linear Kernel

Linear kernel merupakan *kernel* paling sederhana pada SVM. *Kernel* ini digunakan pada data yang dapat dipisahkan secara linear. Berikut merupakan persamaan *linear kernel*.

$$K(x,y) = x^T \cdot y \tag{2.2}$$

Pada persamaan 2.2, x merupakan vektor yang merepresentaiskan *feature* dari dataset dan y adalah label atau kelas. Visualisasi dari SVM *linear kernel* dapat dilihat pada Gambar 2.4.



Gambar 2.4. SVM Linear Kernel

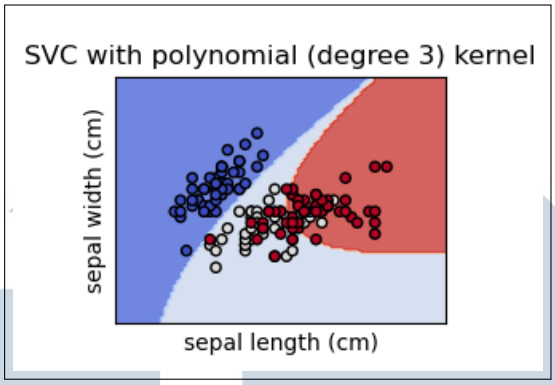
Sumber: [16]

B. Polynomial Kernel

Polynomial kernel merupakan kernel yang fleksibel karena memanfaatkan persamaan polinomial. *Hyperplane* pemisah yang dihasilkan dapat berbentuk bebas sesuai dengan derajat polinomialnya. *Kernel* ini digunakan ketika data tidak dapat dipisahkan secara linear untuk menghindari *under fitting*. Berikut merupakan persamaan *polynomial kernel*.

$$K(x,y) = (\gamma x^T y + r)^d \tag{2.3}$$

Pada persamaan 2.3, x merupakan vektor yang merepresentaiskan *feature* dari dataset dan y adalah label atau kelas. Lalu γ sebagai faktor skala, r adalah koefisien, dan d adalah pangkat polinomial. Visualisasi dari SVM *Polynomial kernel* dapat dilihat pada Gambar 2.5.



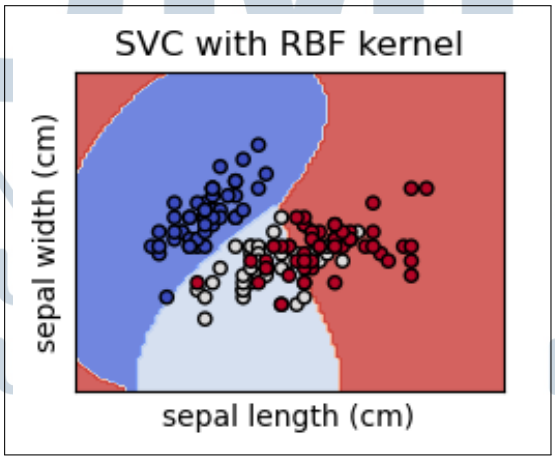
Gambar 2.5. SVM Polynomial Kernel
Sumber: [16]

C. Radial Basis Function Kernel

RBF *kernel* atau dikenal sebagai Gaussian *kernel* non linear yang memanfaatkan fungsi eksponensial. RBF *kernel* juga dimanfaatkan untuk data yang tidak dapat dipisahkan secara linear namun dengan performa yang lebih baik dibanding *polynomial kernel* dengan parameter tertentu [16]. Berikut merupakan persamaan RBF *kernel*.

$$K(x,y) = \exp\left(-\gamma||x-y||^2\right) \tag{2.4}$$

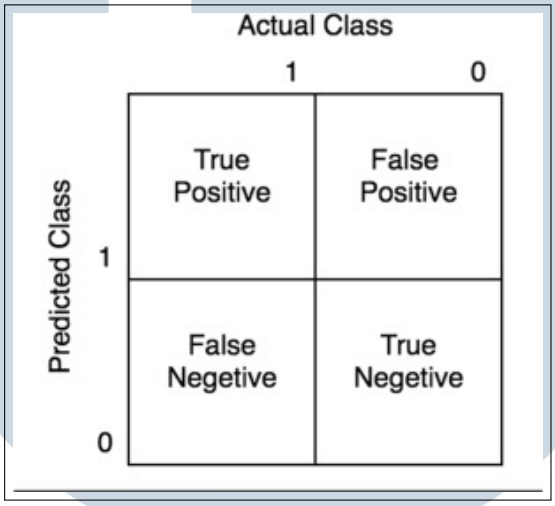
Pada persamaan 2.4, x merupakan vektor yang merepresentaiskan *feature* dari dataset y adalah label atau kelas. Lalu γ sebagai faktor skala dan exp adalah fungsi exponensial. Visualisasi dari SVM RBF *kernel* dapat dilihat pada Gambar 2.6.



Gambar 2.6. SVM RBF Kernel
Sumber: [16]

2.4 Confusion Matrix

Confusion matrix adalah sebuah matriks yang digunakan untuk melakukan evaluasi terhadap model klasifikasi. Confusion matrix berguna untuk visualisasi performa algoritma klasifikasi dengan membandingkan kelas yang sebenarnya dan kelas hasil prediksi [16].



Gambar 2.7. Confusion Matrix
Sumber: [17]

Pada gambar 2.7, terdapat beberapa terminologi yang digunakan dalam mengukur performa algoritma klasifikasi melalui Confusion Matrix, yaitu:

1. *True Positive* (TP): Jumlah entri data yang diprediksi dengan benar sebagai kelas positif.
2. *True Negative* (TN): Jumlah entri data yang diprediksi dengan benar sebagai kelas negatif.
3. *False Positive* (FP): Jumlah entri data yang salah diprediksi sebagai kelas positif ketika sebenarnya mereka kelas negatif.
4. *False Negative* (FN): Jumlah entri data yang salah diprediksi sebagai kelas negatif ketika sebenarnya mereka kelas positif.

Dari hasil visualisasi Confusion Matrix dapat dihitung nilai akurasi, presisi, *recall*, dan F1-score.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.5}$$

Pada persamaan 2.5, akurasi adalah proporsi prediksi yang benar dibagi dengan seluruh prediksi yang dibuat oleh model. Akurasi digunakan sebagai acuan keakuratan sebuah model dalam melakukan klasifikasi.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.6}$$

Presicion atau presisi adalah ketepatan sebuah model klasifikasi yang diukur seperti pada persamaan 2.6 dengan membagi total TP dengan seluruh prediksi positif.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.7}$$

Recall merupakan nilai yang mengukur sensitivitas sebuah model klasifikasi terhadap prediksi positif. Pada persamaan 2.7, *recall* diukur dengan membagi TP dengan jumlah TP + FN.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

F1-score adalah pembobotan nilai *precision* dan *recall* untuk mengukur ketepatan prediksi berdasarkan kelasnya. Pada persamaan 2.8, F1-score adalah dua kali *precision* kali *recall* dibagi dengan jumlah dari *precision* ditambah *recall*.