BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis Sentimen adalah proses penggunaan *text analytics* untuk mendapatkan berbagai sumber data dari internet dan beragam *platform* media sosial. Tujuannya adalah untuk memperoleh opini dari pengguna yang terdapat pada platform tersebut. Analisis Sentimen merupakan salah satu bidang dari *Natural Language Processing*(NLP) yang membangun sistem untuk mengenali dan mengekstraksi opini dalam bentuk teks. Informasi berbentuk teks saat ini banyak terdapat di internet dalam format forum, blog, media sosial, serta situs berisi *review*. Dengan bantuan analisis sentiment, informasi yang tadinya tidak terstruktur dapat diubah menjadi data yang lebih terstruktur. Ada beragam jenis analisis sentiment yang dapat digunakan untuk mengidentifikasi respon pengguna, diantaranya:

1. *Fine-Grained Sentiment Analysis.*
2. *Intent Sentiment Analysis*.
3. *Aspect-Based Sentiment Analysis.*

Menurut G.Vinodhini dan RM.Chandrasekaran pada penelitiannya tahun 2012, analisis sentimen adalah jenis pemrosesan bahasa alami untuk melacak mood publik tentang produk atau topik tertentu. Analisis sentimen digunakan untuk membangun sistem yang dapat memeriksa pendapat tentang produk yang dibuat di postingan blog, komentar, ulasan, atau tweet. Analisis sentimen dapat berguna dalam beberapa kasus, seperti dalam pemasaran. Dimana dapat membantu menilai keberhasilan sebuah iklan kampanye atau peluncuran produk baru[11].

2.2 Sarkasme

Sarkasme merupakan pemakaian kata- kata pedas guna menyakiti hati orang lain, cemoohan maupun ejekan agresif[6]. Kata sarkasme diturunkan dari kata Yunani *sarkamos* yang berarti merobek-robek daging seperti anjing, menggigit

bibir karena marah, atau berbicara dengan kepahitan. Sarkasme memiliki ciri utama, yaitu selalu mengandung kepahitan dan celaan yang getir, menyakiti hati, dan kurang enak didengar. Apabila dibandingkan dengan ironi dan sinisme, maka sarkasme lebih kasar. Sarkasme menurut Kasno Atmo Sukarto pada penelitiannya tahun 2022 adalah gaya bahasa sindiran yang menggunakan kata-kata kasar dan pahit dengan tujuan untuk menyakiti perasaan pendengarnya[12]. Berikut merupakan contoh dari kalimat sarkasme :

1. “Mulut kau harimau kau dan lihat sang raksasa itu.”
2. “Wow, kau benar-benar terlihat hebat dengan rambutmu yang berantakan dan pakaian yang kusut. Apa itu tren fashion terbaru?”
3. “Terima kasih atas kontribusimu yang luar biasa dalam rapat tadi. Saya sangat terkesan dengan kemampuan 5 menit presentasimu yang tidak memberikan apa-apa.”

2.3 Politikus

Politikus adalah individu yang terlibat dalam aktivitas politik dan berperan dalam proses pengambilan keputusan dan pemerintahan negara. Sebagai anggota masyarakat yang terlibat dalam politik, politikus berusaha untuk mempengaruhi pembentukan kebijakan publik, mewakili kepentingan kelompok atau partai politik, dan memperjuangkan agenda politik tertentu. Mereka berpartisipasi dalam berbagai aktivitas politik seperti kampanye pemilihan, pengambilan keputusan legislatif, negosiasi kebijakan, dan interaksi dengan masyarakat. Politikus dapat mencakup pemimpin politik seperti presiden, perdana menteri, anggota parlemen, atau pejabat pemerintahan lainnya. Peran politikus sangat penting dalam demokrasi, di mana mereka menjadi perantara antara masyarakat dan sistem politik, serta bertanggung jawab untuk memperjuangkan kepentingan publik dan menciptakan perubahan sosial melalui proses politik[13].

2.4 *Data Preprocessing*

Data Preprocessing adalah proses pengolahan data asli yang akan dipersiapkan untuk diolah pada tahap selanjutnya[14]. Tahapan *preprocessing* yang akan dilakukan pada penelitian ini diantaranya *case folding*, *tokenization*, *remove HTML encoding*, *Remove Mentions*, *Remove Hashtags*, *Remove Weblinks*, *Remove Punctuations*, *Remove New Lines and Double Whitespaces*, *stopwords removal*, *replace unknown words*, *resampling*, dan GloVe.

2.4.1 *Case Folding*

Case Folding adalah tahap untuk mengonversi text menjadi suatu bentuk yang standar, Pada tahap ini biasanya dipilih *lowercase* untuk membuat huruf besar/kapital menjadi huruf kecil. Berikut merupakan contoh dari penerapan *case folding* pada suatu kalimat :

1. Teks *Input* : UNIKOM adalah salah satu Universitas swasta di Kota Bandung.
2. Hasil : unikom adalah salah satu universitas swasta di kota bandung.

2.4.2 *Tokenization*

Tokenization merupakan proses untuk memecah teks pada suatu dokumen menjadi kata/token. Contoh dari penerapan *tokenization* pada suatu kalimat dapat dilihat pada tabel 2.1:

1. Teks *Input* : unikom adalah salah satu universitas swasta di kota bandung.
2. Hasil :

Tabel 2.1 Contoh Tokenisasi

unikom
adalah
salah

satu
universitas
swasta
di
kota bandung

2.4.3 *Remove HTML Encoding*

Remove HTML Encoding adalah tahap yang dilakukan untuk mengubah *HTML entity* menjadi karakter normal kemudian menghapus tag HTML tersebut. Contoh dari *HTML entity* diantaranya:

1. <
2. >
3.
4. ¥
5. ©

Dan berikut merupakan contoh implementasi pada suatu kalimat:

1. Kalimat Awal :

UNIKOM adalah salah satu universitas swasta di kota Bandung

2. Hasil Proses mengubah *HTML Entity* menjadi karakter normal :

UNIKOM adalah salah satu universitas swasta di kota Bandung

3. Hasil Proses penghapusan tag HTML :

UNIKOM adalah salah satu universitas swasta di kota Bandung

2.4.4 *Remove Mentions*

Remove Mentions adalah tahap untuk menghapus nama akun pengguna twitter pada suatu *text* yang biasanya diawali dengan karakter @. Berikut merupakan contoh dari nama akun pada media sosial twitter:

1. @caknundotcom
2. @aniesbaswedan
3. @puanmaharani_ri

Dan berikut merupakan contoh implementasi *remove mentions* pada suatu kalimat:

1. Kalimat Awal :

@aniesbaswedan prettttt,gagasan mulu yg dibahas

2. Hasil Proses *Remove Mentions* :

prettttt,gagasan mulu yg dibahas

2.4.5 *Remove Hashtags*

Remove Hashtags adalah tahap untuk menghapus tagar(#) dalam sebuah text. Berikut ini merupakan contoh dari tagar pada sebuah kalimat :

1. #CakNun
2. #IdolaMilenial
3. #IkutPakde

Dan berikut merupakan contoh implementasi *remove hashtags* pada suatu kalimat:

1. Kalimat Awal :

Andre menilai pernyataan Cak Nun menunjukkan kalau Prabowo Subianto, merupakan calon presiden paling tepat. #IdolaMilenial #IkutPakde

2. Hasil Proses *Remove Hashtags* :

Andre menilai pernyataan Cak Nun menunjukkan kalau Prabowo Subianto, merupakan calon presiden paling tepat.

2.4.6 *Remove Weblinks*

Remove Weblinks adalah tahap untuk menghapus tautan sebuah web yang terdapat pada sebuah kalimat. Berikut merupakan contoh dari tautan web pada sebuah kalimat :

1. https://t.co/ksziGGp1k3
2. https://t.co/EPZTVGDm5d
3. https://t.co/9h9jU28T2L

Dan berikut merupakan contoh implementasi *remove weblinks* pada suatu kalimat:

1. Kalimat awal :

VIDEO: Sinau Bareng di Masjid Quba Tangerang 21 November 1998 — https://t.co/Hv0vSVqhpC

2. Hasil Proses *Remove Weblinks* :

VIDEO: Sinau Bareng di Masjid Quba Tangerang 21 November 1998 —

2.4.7 *Remove Punctuations*

Remove Punctuations adalah tahap untuk menghapus tanda baca pada suatu kalimat. Berikut merupakan contoh tanda baca yang biasanya muncul pada suatu kalimat :

1. Titik (.)
2. Koma (,)
3. Tanda Tanya (?)
4. Tanda Seru(!)
5. Tanda Petik(“)
6. dll.

Dan berikut merupakan contoh implementasi *remove punctuations* pada suatu kalimat :

1. Kalimat awal :

Andre menilai pernyataan Cak Nun menunjukkan kalau Prabowo Subianto, merupakan calon presiden paling tepat.

2. Hasil Proses *Remove Punctuations* :

Andre menilai pernyataan Cak Nun menunjukkan kalau Prabowo Subianto merupakan calon presiden paling tepat

2.4.8 *Remove New Lines and Double Whitespaces*

Remove New Lines and Double Whitespaces adalah tahap untuk menghapus jarak antar kata yang berlebih dan juga jarak antar paragraf. Berikut merupakan contoh implementasi *remove new lines and double whitespaces* pada suatu kalimat :

1. Kalimat awal :

Mbah nun

BANCI KOMUNIS SENGKUNI:
yg NGUMPET diBELAKANK Akun IreneViena

Mantan JONGOS'y:S.B.Ydyno Kubu 01

👉Lantas PINDAH:Tapi TETAP
nJONGOS juga ke:TIKUS KONTET JUSUF KOLLO(LUHUT.B.P) Kubu 01 juga👈

TETAP nJONGOS ke:Surabaya(NURHADI)

Pak

2. Hasil proses *remove new lines and double whitespaces* :

Mbah nun BANCI KOMUNIS SENGKUNI yg NGUMPET diBELAKANK Akun IreneViena Mantan JONGOS'y:S.B.Ydyno Kubu 01 👉Lantas PINDAH:Tapi TETAP nJONGOS juga ke:TIKUS KONTET JUSUF KOLLO(LUHUT.B.P) Kubu 01 juga👈 TETAP nJONGOS ke:Surabaya(NURHADI) Pak

2.4.9 *Replace Unknown Words*

Replace Unknown Words adalah tahap untuk mengganti kata – kata gaul / kata yang tidak baku menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia(KBBI). Berikut merupakan contoh kata yang termasuk kedalam kategori *unknown words* :

1. Batrey
2. Ancem
3. Sdg
4. Belobang

Contoh implementasi *replace unknown words* pada suatu kata dapat dilihat pada tabel 2.2:

Tabel 2.2 Contoh Replace Unknown Words

Unknown Words	**Replace Words**
Batrey	Baterai
Ancem	Ancam
Sdg	Sedang
Belobang	Berlubang

2.4.10 *Stopwords removal*

Stopwords Removal adalah tahap yang dilakukan untuk menghapus kata-kata yang tidak penting, yang berarti jika kata tersebut dihapus dari suatu kalimat, maka tidak akan mempengaruhi atau merubah makna dari suatu kalimat. Contoh dari kata *stopword* diantaranya :

1. adalah
2. adapun
3. agaknya
4. akan
5. akhir
6. dll

Dan berikut merupakan contoh implementasi pada suatu kalimat:

1. Token :

unikom
adalah
salah
satu
universitas

swasta
di
kota bandung

2. Hasil :

unikom
salah
universitas
swasta
kota
bandung

2.4.11 *Over**sampling***

adalah proses yang bekerja dengan membuat replikasi dari data minoritas, sehingga jumlah dari *class* minoritas sama dengan *class* mayoritas[15].

2.4.12 GloVe

GloVe(*Global Vectors for Word Representation*) adalah salah satu metode untuk menghasilkan representasi *word embedding* dalam pemrosesan bahasa alami. Metode ini bekerja dengan cara menghitung kemunculan satu kata dengan kata lainnya lalu mengonversinya menjadi *vector*. Tujuan utama dari GloVe adalah untuk menghasilkan *vector word embedding* yang memperhitungkan hubungan antara kata-kata yang berbeda dalam suatu korpus teks. Metode ini mengambil pendekatan yang berbeda dengan Word2Vec, yang bertujuan untuk menghasilkan representasi *vector* yang memiliki kemampuan prediksi kata-kata berikutnya dalam suatu kalimat atau dokumen.

GloVe menghasilkan *vector word embedding* dengan menerapkan faktorisasi matriks pada matriks kemunculan kata-kata. Pada dasarnya, metode ini mencari hubungan antara kata-kata berdasarkan distribusi kata-kata dalam korpus. Metode/Algoritma GloVe menghitung frekuensi kemunculan pasangan kata dan kemudian menghitung nilai bobot yang mencerminkan seberapa sering pasangan kata tersebut muncul Bersama-sama dalam konteks tertentu. Dalam proses ini, GloVe menggunakan faktorisasi matriks untuk menemukan korelasi yang lebih besar antara kata-kata, yang memungkinkan pembentukan representasi *vector* yang lebih baik. Metode GloVe telah terbukti efektif dalam tugas-tugas seperti analisis sentimen dan pengelompokkan kata-kata dalam korpus teks, serta dalam aplikasi pengenalan wacana, dan klasifikasi dokumen[16]. Berikut merupakan contoh implementasi metode GloVe pada sebuah teks :

- Teks *Input* : ["kucing", "itu", "sangat", "lucu", "dan", "menggemaskan"]

- Hasil Matriks Co-Occorrence dengan *windows-size* = 2 dapat dilihat pada tabel 2.3.

Tabel 2.3 Contoh Co-Matrix GloVe

	kucing	itu	sangat	lucu	dan	menggemaskan
kucing	0	1	1	0	0	0
itu	1	0	1	1	0	0
sangat	1	1	0	1	1	0
lucu	0	1	1	0	1	1
dan	0	0	1	1	0	1
menggemaskan	0	0	0	1	1	0

2.4.13 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) adalah salah satu turunan dari *oversampling*. SMOTE pertama kali diperkenalkan oleh Nithes V. Chawla. Pendekatan ini bekerja dengan membuat replikasi dari data minoritas.

Replikasi tersebut dikenal dengan data sintetis(*synthetic data*). Metode SMOTE bekerja dengan mencari K-Nearest Neighbors untuk setiap kata di kelas minoritas. Setelah itu dibuat data sintetis sebanyak presentase duplikasi yang diinginkan antara data minor dan K-Nearest Neighbors yang dipilih secara acak[8]. Berikut merupakan Algoritma dari metode SMOTE:

a. Hitung jumlah sampel minoritas (Nmin) dan mayoritas (Nmaj) pada data training.
b. Tentukan rasio oversampling yang diinginkan (pada umumnya 1:1, 1:2, atau 1:3) dan hitung jumlah sampel sintetis yang perlu ditambahkan pada kelas minoritas dengan rumus: Nsyn = (Nmin * rasio_oversampling) – Nmin.
c. Untuk setiap sampel minoritas pada data training, cari *K-nearest neighbors* (KNN) dan pilih satu tetangga secara acak.
d. Untuk setiap tetangga yang dipilih, buatlah sampel sintetis baru dengan menghitung perbedaan antara sampel minoritas dan tetangganya, dan kemudian mengalikan perbedaan tersebut dengan faktor acak dari 0 hingga 1. Rumus perhitungan perbedaan antara dua sampel dapat dihitung dengan rumus:

   Diff = X2 - X1, dimana X2 adalah vektor fitur dari tetangga terpilih, dan X1 adalah vektor fitur dari sampel minoritas.
e. Tambahkan sampel sintetis baru ke data training, dan ulangi langkah 3-5 sampai jumlah sampel sintetis yang ditentukan tercapai.

2.5 *Data Labelling*

Data Labelling adalah proses pemberian label terhadap Datasets. Pada kasus analisis sentimen, label ini memberikan informasi tentang sentimen yang terkandung dalam teks yang sangat berguna dalam memahami pandangan atau reaksi pengguna terhadap suatu topik. Proses *data labelling* dapat dilakukan secara manual oleh manusia yang membaca dan menganalisis teks untuk menentukan sentimen yang tepat, atau dapat juga menggunakan teknik pemodelan otomatis dengan memanfaatkan metode *machine learning*. *Data labelling* yang akurat dan

konsisten merupakan faktor kunci dalam membangun model analisis sentimen yang baik, karena kualitas dan keakuratan label akan mempengaruhi performa model dalam melakukan klasifikasi sentimen dengan tepat[17]. Rumus yang digunakan untuk menentukan sentimen text dapat dilihat pada persamaan 1, 2, dan 3[16]:

$$pos\_ratio_t = \frac{\Sigma\ positive\_word_t}{total\_token_t} \quad (1)$$

$$neg\_ratio_t = \frac{\Sigma\ negative\_word_t}{total\_token_t} \quad (2)$$

$$\begin{aligned} &if\ pos_{ratio_t} > neg_{ratio_t}, then\ positive\ or\ 1 \\ &if\ pos_{ratio_t} < neg_{ratio_t}, then\ negative\ or\ 0 \\ &if\ pos_{ratio_t} = neg_{ratio_t}, then\ neutral\ or\ 2 \end{aligned} \quad (3)$$

Kemudian rumus untuk menentukan sentimen emoji dapat dilihat pada persamaan 4.

$$\begin{aligned} &if\ pos_{count_t} > neg_{count_t}, then\ positive\ or\ 1 \\ &if\ pos_{count_t} < neg_{count_t}, then\ negative\ or\ 0 \\ &if\ pos_{count_t} = neg_{count_t}, then\ neutral\ or\ 2 \end{aligned} \quad (4)$$

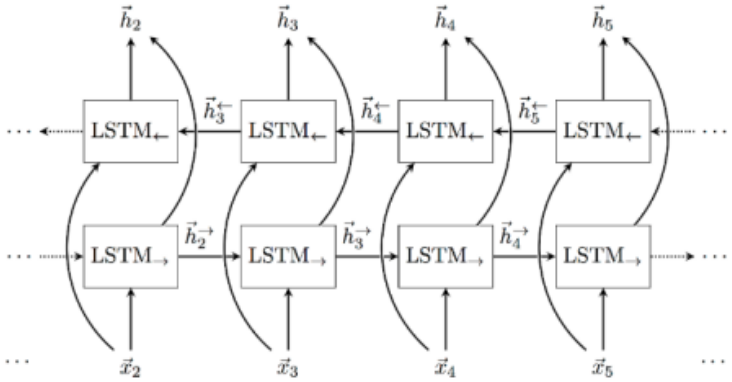
Kemudian yang terakhir rumus untuk menentukan apakah suatu kalimat mengandung unsur sarkas atau tidak dapat dilihat pada persamaan 5.

$$\begin{aligned} &if\ sentiment\_label_t = emoji\_label_t, then\ negative\ or\ 0 \\ &if\ sentiment\_label_t \neq emoji\_label_t, then\ positive\ or\ 1 \end{aligned} \quad (5)$$

2.6 Bi-LSTM

Bidirectional Long Short-Term Memory (Bi-LSTM) adalah salah satu jenis arsitektur jaringan saraf rekursif (RNN) yang digunakan dalam pemrosesan bahasa alami. Arsitektur ini memungkinkan informasi untuk mengalir ke depan dan ke belakang dalam jaringan, sehingga memungkinkan model untuk memperhitungkan konteks dalam teks secara lebih efektif. Arsitektur Bi-LSTM terdiri dari dua layer LSTM, satu layer yang mengalir maju(*forward*) dan satu layer yang mengalir mundur(*backward*), sehingga memungkinkan model untuk mengintegrasikan informasi dari kedua arah.

Bi-LSTM adalah pengembangan dari model LSTM, dimana metode / algoritma ini memanfaatkan konteks sebelumnya dan konteks setelahnya dengan memproses data dari dua arah dengan *hidden layer* terpisah. *Forward Layer* untuk merepresentasikan konteks sebelumnya, dan *backward layer* untuk merepresentasikan konteks setelahnya[18]. Keluaran dari kombinasi dua arah *hidden layer* $\vec{h}_t$ dan $\overleftarrow{h_t}$ adalah : $y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h_t}$. Untuk arsitektur Bi-LSTM dapat dilihat pada gambar 2.4.



Gambar 2.1 Arsitektur Bi-LSTM

Salah satu keuntungan utama dari arsitektur Bi-LSTM adalah kemampuannya untuk mengatasi masalah vanishing gradients yang sering terjadi pada arsitektur

RNN tradisional. Masalah vanishing gradients dapat terjadi ketika gradien yang mengindikasikan seberapa besar pengaruh suatu neuron terhadap kesalahan total model menyebar melalui jaringan. Dalam arsitektur RNN tradisional, gradien tersebut dapat mengalami penurunan nilai secara signifikan ketika mengalir ke belakang melalui banyak layer, sehingga menghambat pelatihan model secara efektif. Dengan adanya dua layer LSTM pada arsitektur Bi-LSTM yang mengalir maju dan mundur, gradien dapat mengalir lebih lancar dan lebih cepat, sehingga memungkinkan model untuk belajar dengan lebih efektif. Arsitektur Bi-LSTM telah terbukti sangat efektif dalam tugas-tugas pemrosesan bahasa alami seperti klasifikasi teks, analisis sentimen, dan pengenalan entitas. Berikut merupakan rumus-rumus perhitungan yang digunakan dalam metode Bi-LSTM :

a) Forget Gate

$$f_t = \sigma(W_f . [h_{t-1}, X_t] + b_f)$$

b) Input Gate

$$i_t = \sigma(W_i . [h_{t-1}, X_t] + b_i)$$

c) Cell State

$$C_t = tanh(W_c.[h_t\_1, X_t] + b_c)$$

d) Output Gate

$$O_t = \sigma(W_o.[h_t\_1, X_t] + b_o)$$

e) Hidden State

$$h_t = O_t.tanh(C_t)$$

2.7 *Confusion Matrix*

Confusion matrix adalah sebuah metode evaluasi performa klasifikasi pada suatu model atau algoritma *machine learning*. Confusion matrix terdiri dari empat nilai utama, yaitu *true positive* (TP), *false positive* (FP), *true negative* (TN), dan *false negative* (FN). True positive adalah jumlah data yang berhasil diprediksi benar sebagai kelas positif, false positive adalah jumlah data yang seharusnya diprediksi

sebagai kelas negatif namun terprediksi sebagai kelas positif, true negative adalah jumlah data yang berhasil diprediksi benar sebagai kelas negatif, dan false negative adalah jumlah data yang seharusnya diprediksi sebagai kelas positif namun terprediksi sebagai kelas negatif.

Confusion matrix sangat penting dalam evaluasi performa model karena mampu memberikan gambaran yang jelas tentang seberapa baik atau buruk performa model dalam melakukan klasifikasi. Selain itu, confusion matrix juga bisa digunakan untuk menghitung berbagai metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1 score*. Hal ini berguna untuk mengevaluasi seberapa baik model dapat membedakan antara kelas positif dan kelas negatif. Dengan demikian, *confusion matrix* menjadi salah satu alat evaluasi penting dalam machine learning untuk mengukur performa model dan meningkatkan akurasi prediksi. Rumus / *formula* dari perhitungan *metrics precision*, *recall*, *F1 score,* dan *accuracy* dapat dilihat pada persamaan 6, 7, 8, dan 9.

$$Precision = \frac{TP}{(TP+FP)} \tag{6}$$

$$Recall = \frac{TP}{(TP+FN)} \tag{7}$$

$$F1 - Score = \frac{(2\ x\ Precision\ x\ Recall)}{(Precision+Recall)} \tag{8}$$

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \tag{9}$$