**BAB II TINJAUAN PUSTAKA**

## 2.1 Emosi

Emosi menurut Jabreel dan Moreno(2019) adalah kunci perasaan dan pikiran orang (Jabreel dan Moreno, 2019). Menurut Graind (2019) perasaan intens yang diarahkan pada sesuatu atau seseorang sebagai respon terhadap peristiwa internal maupun eksternal bagi individu digambarkan sebagai emosi (Gaind et al., 2019). Emosi bersifat umum dan penting dalam semua aspek kehidupan manusia.

Variasi jenis emosi dapat didasarkan kepribadian, jenis kelamin, lokasi, etnis, budaya, situasi, di samping banyak parameter psikologis, sosial dan individu lainnya (Rohman et al., 2020). Model emosi yang sangat populer adalah model emosi Ekman. Model Ekman membagi emosi menjadi 6 label emosi yaitu bahagia, marah, takut, jijik, sedih, dan terkejut (Ekman, 1999), sebenarnya banyak model emosi yang lain namun pada model ekman label-label emosi bersifat universal pada budaya yang berbeda.

## 2.2 Deteksi Emosi

Deteksi Emosi merupakan bagian dari area yang lebih luas dari komputasi afektif bertujuan untuk memungkinkan komputer mengenali dan mengekspresikan emosi (Picard, 1997). Pendekatan klasifikasi teks digunakan untuk melakukan deteksi emosi (Fanesya et al., 2019). Penelitian deteksi emosi memiliki banyak kegunaan dalam mengambil keputusan seperti di bidang pendidikan untuk memprediksi keadaan emosi siswa dengan platform e-learning sehingga dapat digunakan untuk meningkatkan kualitas belajar (Daouas dan Lejmi, 2018), bidang bisnis digunakan untuk review kualitas produk (Nurlaila et al., 2017), selain itu deteksi emosi juga bermanfaat untuk rekrutmen karyawan, koseling, dan prosfil psikologis karena merupakan bagian dari prediksi kepribadian (Dandannavar et al., 2018). Emosi dapat dideteksi melalui suara, ekspresi wajah, gerakan tangan, gerakan tubuh, detak jantung, tekanan darah, teks (Consoli, 2009).

## 2.3 Media Sosial

Media sosial merupakan platform komunikasi, interaksi, dan berbagi informasi yang populer melalui internet (Wilson et al., 2012). Media sosial dapat digunakan oleh penggunanya untuk berekspreksi (Sari & Haranto, 2019). Pada media sosial, setiap acara, berita atau aktivitas di seluruh dunia dapat dibagikan, didiskusikan, diposting dan dikomentari oleh jutaan orang (Gaind et al., 2019). Teks, gambar, suara , dan video dapat dibagikan di media sosial (Kotler et al., 2012). Berdasarkan penjelasan tersebut dapat disimpulkan bahwa media sosial merupakan platform berbagi informasi, acara, aktifitas berupa teks, gambar, suara, video melalui internet untuk mengekspresikan penggunanya, dimana informasi, acara, aktifitas tersebut dapat didiskusikan, dan dikomentari oleh jutaan orang. Contoh media sosial seperti *facebook*, *twitter*, *instagram*, *line*.

## 2.4 Twitter

Twitter adalah salah satu media sosial dimana pengguna dapat membaca dan berbagi pesan singkat dengan jumlah maksimal 280 karakter, *twitter* termasuk media sosial dengan kategori *microblogging* (Setiawan dan Setyohadi, 2017), hal ini yang membuat *twitter* berbeda dengan media sosial yang lain. *Twitter* didirikan oleh Jack Dorsey sejak 2006, *twitter* kini telah berkembang menjadi media sosial populer (Juwiantho et al., 2020). *Twitter* memiliki laju pertumbuhan pengguna tertinggi di indonesia, pengguna aktif *twitter* di Indonesia menempati posisi ketiga di Asia Pasifik dari tahun 2012 sampai tahun 2018 (Saputri et al., 2019). Oleh karena itu *twitter* dipilih menjadi objek pada penelitian ini.

## 2.5 Web Scraping

*Web scraping* atau *screen scraping* merupakan proses pengambilan dan analisis sebuah dokumen semi-terstruktur tertentu yang diambil dari internet, biasanya berupa halaman-halaman web dalam bahasa *markup* seperti HTML, XHTML untuk digunakan bagi kepentingan lain (Turland, 2010).

*Web scaping* pertama kali dilakukan dengan cara manual, yaitu menyalin data dari website ke komputer kita atau penyimpanan lokal. Namun, cara ini kurang efektif jika digunakan untuk mengambil data dalam jumlah banyak, karena membutuhkan tenaga dan waktu yang banyak. Cara lain *web scraping* dapat dilakukan dengan *coding*, aplikasi ataupun *extension browser*. Pada penelitian ini menggunakan *coding* untuk *web scraping* dengan memanfaatkan *twitter* dan *library* dari bahasa pemprograman *python*.

Menurut Josi dkk (2014) langkah-langkah dalam melakukan *web scraping* ada beberapa tahapan, sebagai berikut (Josi et al., 2014) :

1. *Create scraping template* : mempelajari dokumen HTML dari website yang akan diambil informasinya untuk tag HTML yang mengapit informasi yang akan diambil.
2. *Explore site navigation* : mempelajari teknik navigasi pada website yang akan diambil informasinya untuk ditirukan pada aplikasi *web scraper* yang akan dibuat.
3. *Automate navigation and extraction* : berdasarkan informasi yang didapat pada tahap dan tahap 2, aplikasi *web scaper* dibuat untuk mengotomatisasi pengambilan informasi dari website yang ditentukan.
4. *Extracted data and package* : informasi yang didapat dari tahap 3 disimpan dalam format data tertentu, seperti *database*, csv.

## 2.6 Preprocessing

*Preprocessing* data seringkali mempengaruhi kinerja dari *machine learning*, sehingga lebih efektif (S. B. Kotsiantis, 2006). *Preprocessing* berguna untuk membersihkan data dari derau dan bagian tidak informatif yang tidak dibutuhkan, sehingga teks siap untuk diklasifikasi (Haddi et al., 2013). *Preprocessing* akan menjadikan data bersih sehingga proses pembuatan vektor kata dan klasifikasi lebih akurat (Nurrohmat & SN, 2019).

**2.6.1 Case Folding**

*Case Folding* adalah mengubah seluruh huruf dalam teks tersebut menjadi *lower case* atau huruf kecil semua (Salam et al., 2018), dengan huruf yang diterima dari “a” sampai “z”. *Case folding* bertujuan agar terdapat standarisasi dalam penulisan. Misalnya terdapat data “Emosi Saya” setelah proses *case folding* menjadi “emosi saya”. Contoh lain dapat dilihat pada Tabel 2.1.

**Tabel 2.1 *Case Folding***

|  |  |
| --- | --- |
| Data sebelum *case folding* | “Suaranya bagus, apalagi kalau dIiringi gitaR” |
| Data setelah *Case Folding* | “suaranya bagus, apalagi kalau diiringi gitar” |

### 2.6.2 Remove Punctuation

*Remove punctuation* adalah menghapus tanda baca pada teks dengan tujuan mengurangi beban pemprosesan klasifikasi karena dianggap tidak penting dan termasuk *delimiter*, contoh tanda baca yang yang dihapus titik (.), koma(,), tanda tanya (?), *slash* (/), *hastag* (#), tanda seru (!) dan lain-lain. Misal terdapat data “sabar pak?” setelah melalui *remove punctuation* menjadi “sabar pak” Contoh lain dapat dilihat pada Tabel 2.2.

**Tabel 2.2 *Remove Punctuation***

|  |  |
| --- | --- |
| Data sebelum *remove punctuation* | “suaranya bagus !!!, apalagi kalau diiringi gitar” |
| Data setelah *remove punctuation* | “suaranya bagus apalagi kalau diiringi gitar” |

### 2.6.3 Remove Number

*Remove number* adalah menghapus angka pada suatu teks, penghapusan ini sebab angka dianggap tidak memiliki arti dan termasuk *delimiter*, mirip seperti *remove punctuation* hanya berbeda pada objek yang dihapus. Misalnya terdapat data “dokumen2 yang dikumpulkan” setelah melalui proses *remove number* menjadi “dokumen yang dikumpulkan”.Contoh lain dapat dilihat pada Tabel 2.3.

**Tabel 2.3 *Remove number***

|  |  |
| --- | --- |
| Data sebelum *remove number* | “barang yang dijual bagus2” |
| Data setelah *remove number* | “barang yang dijual bagus” |

### 2.6.4 Tokenizing

*Tokenizing* atau tokenisasiadalah pemotongan kalimat berdasarkan tiap-tiap kata penyusunnya, biasanya pemotongan berdasarkan *whitespace* seperti spasi, tab, dan enter. Tiap-tiap kata hasil *tokenizing* disebut token (Juwiantho et al., 2020). Misalnya terdapat data “ada saya semua bahagia” setelah melalui proses *tokenizing* akan menghasilkan 4 token, yaitu : “ada”, “saya”, “semua”, “bahagia”. Contoh lain dapat dilihat pada Tabel 2.4.

**Tabel 2.4 *Tokenizing***

|  |  |
| --- | --- |
| Sebelum proses *Tokenizing* | konten yang bermutu namun sayang suaranya kurang jelas |
| Setelah proses *Tokenizing* | [‘konten’,’yang’,’bermutu’,’namun’,’sayang’,’suaranya’,  ’kurang’,’jelas’] |

### 2.6.5 Stopword Removal

*Stopword removal* atau *filtering* merupakan proses untuk menghilangkan kata-kata yang tidak memiliki arti atau makna, namun tidak akan mengubah makna dari komentar tersebut (Juwiantho et al., 2020). *Stopword removal* akan mengurangi ukuran indeks, waktu pemprosesan klasifikasi, dan *noise* dari suatu data. Biasanya *stopword* berupa kata ganti orang dan kata hubung, seperti “aku”, “kamu”, “kita”, “dan”, “atau”. Contoh proses *stopword removal* yang datanya berasal dari proses *tokenizing* sebelumnya dapat dilihat pada Tabel 2.5.

**Tabel 2.5 *Stopword Removal***

|  |  |
| --- | --- |
| Sebelum Stop Removal | [‘konten’,’yang’,’bermutu’,’namun’,’sayang’,’suaranya’,  ’kurang’,’jelas’] |
| Setelah Stop Removal | [‘konten’,’bermutu’,’sayang’,’suaranya’,’kurang’,’jelas’] |

### 2.6.6 Stemming

*Stemming* atau *lemmatization* merupakan proses untuk mentransformasi kata kerja yang berimbuhan pada suatu dokumen menjadi kata dasar (*root word*), dengan menghapus awalan, akhiran maupun sisipan . *Stemming* bertujuan untuk mengurangi varian kata dengan makna yang hampir sama pada suatu dokumen, serta meningkatkan performa pada tahap *information retrieval*. Pada penelitian ini akan menggunakan *library* sastrawi, karena data pada penelitian ini menggunakan bahasa Indonesia. *Library* sastrawi menerapkan algoritma Nazief dan Adriani. Berikut langkah-langkah algoritma Nazief & Adriani dalam *stemming* sebagai berikut (Nazief dan Adriani, 2005) :

1. Pertama – tama, periksa kata dengan kamus kata, jika pemeriksaan ada kata tersebut akan dianggap kata dasar maka proses berhenti, jika tidak maka akan lanjut proses ke-2
2. Pada tahap ke-2, hapus *inflectional suffixes* yaitu (“-kah”, “-lah”, “-tah”, “-pun”), kemudian hapus *inflectional possessive pronoun suffixes* seperti (“-ku”, “-mu”, “-nya”), kemudian data hasil *infectional suffixes* dan *inflectional possessive pronoun suffixes* dicek pada kamus kata, jika hasilnya ada maka proses akan berhenti, jika tidak maka lanjut pada proses ke-3. Contoh ada data “hasilnyapun” setelah dihapus *inflectional suffixes* menjadi “hasilnya”, kemudian data dihapus *inflectional possessive pronoun suffixes* menjadi “hasil”.
3. Tahap ke-3, hapus *derivational suffixes* (imbuhan turunan) yaitu (“-i”, “-kan”, “-an”), setelah proses penghapusan *derivational suffixes* (“-i”, “-kan”, “-an”)kata akan dicari pada kamus kata, jika ada maka proses berhenti, namun jika tidak kembalikan huruf yang dihapus sebelum *derivational suffixes* (“-i”, “-kan”, “-an”), kemudian lanjut pada proses ke-4. Contoh data “menggabungkan” setelah penghapusan *derivational suffixes* (“-i”, “-kan”, “-an”)menjadi “menggabung”, karena tidak ada dalam kamus, maka hapus huruf sebelum *derivational suffixes* (“-i”, “-kan”, “-an”)menjadi “menggabun”, setelah dicek tidak ada dalam kamus, maka kembalikan kata yang dihapus tadi menjadi “menggabung” dan lanjut pada proses ke-4.
4. Tahap ke-4, hapus *derivational prefixes* yaitu (“be-“, “di-“, “ke-“, “me-“, “pe-“, “se-“, “te-“). Jika pada tahap ke-3 ada *derivational suffixes* (“-i”, “-kan”, “-an”)yang dihapus maka ke langkah 4a, jika tidak ada yang dihapus ke langkah 4b.
5. kata dicek pada Tabel 2.6, jika ada maka proses berhenti, jika tidak maka lanjut ke tahap 4b.

**Tabel 2.6 Kata awalan akhiran dilarang**

|  |  |
| --- | --- |
| **Awalan** | **Akhiran** |
| be- | -i |
| di- | -an |
| ke | -i, -kan |
| me- | -an |
| se- | -i, -kan |
| te- | -an |

Sumber: (Nazief dan Adriani, 2005)

1. Untuk i sama dengan 1 sampai 3, tentukan tipe awalan kemudian hapus awalan. Jika kata dasar belum ditemukan juga lakukan tahap ke-5, jika sudah maka proses berhenti. Aturan peluruhan kata dapat dilihat pada Tabel 2.7dan Tabel 2.8.

**Tabel 2.7 Aturan peluruhan kata dasar**

|  |  |  |
| --- | --- | --- |
| Aturan | Awalan | Akhiran |
| 1 | berV . . . | ber-V . . . | be-rV |
| 2 | belajar | bel-ajar |
| 3 | berClerC2 | be-ClerC2 . . dimana C1!= {‘r’|’l’} |
| 4 | terV. . . | ter-V . . | te-rV . . |
| 5 | terCer. . . | terCer. . . dimana C!=’r’ |
| 6 | teClerC2 | te-CleC2 . . . dimana C1!=’r’ |

**Tabel 2.8 Aturan peluruhan kata dasar (lanjutan)**

|  |  |  |
| --- | --- | --- |
| Aturan | Awalan | Akhiran |
| 7 | me{I|r|w|y}V. . . | me-{I|r|w|y}V. . . |
| 8 | mem{b|fv}. . . | mem-{b|f|v}. . . |
| 9 | mempe. . . | m-pe. . |
| 10 | mem{r|V|V}. . . | me-m{r|V|V}. . . | me-p{r|V|V}. . . |
| 11 | men{c|d|j|z}. . . | Men-{c|d|j|z}. . . |
| 12 | menV. . . | Me-nV. . . | me-tV. . . |
| 13 | meng{g|h|q|k}. . . | Meng-{g|h|q|k}. . . |
| 14 | mengV. . . | meng-V. . . | meng-kV. . . |
| 15 | mengeC. . . | meng-C. . . |
| 16 | menyV. . . | me-ny. . . | men-sV. . . |
| 17 | memV. . . | mem-pV. . . |
| 18 | pe{w|y}V. . . | pe-{w|y}V. . . |
| 19 | perV. . . | per-V. . . | pe-rV. . . |
| 20 | pem{b|f|v}. . . | pem-{b|f|v}. . . |
| 21 | pem{rV|V}. . . | Pe-m{rV|V}. . . | pe-p{rV|V}. . . |
| 22 | pen{c|d|j|z}. . . | Pen-{c|d|j|z}. . . |
| 23 | penV. . . | Pe-nV. . . | pe-tV. . . |
| 24 | peng{g|h|q}. . . | Peng-{g|h|q} |
| 25 | pengV. . . | Peng-V | peng-kV |
| 26 | penyV. . . | Pe-nya | peny-sV |
| 27 | pelV. . . | Pe-IV. . .; kecuali untuk kata “pelajar” |
| 28 | peCP. . . | Pe-CP. . . dimana C!={r|w|y|I|m|n} dan P!=’er’ |
| 29 | perCerV. . . | Per-CerV. . . dimana C!={r|w|y|I|m|n} |

Langkah-langkah penentuan tipe awalan sebagai berikut :

1. Jika awalannya adalah: “di-”, “ke-“ atau “se-“ maka tipe awalannya secara berturut-turut adalah “di-“, “ke-“ atau “se-”.
2. Jika awalannya adalah: “te-“, “me-“, “be-“ atau “pe-“ maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-“, “ke-“, “se-“, “te-“, “be-“, “me-“ atau “pe-“ maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Hapus awalan jika ditemukan.
5. Tahap ke-5 melakukan *recording*, pada awal kata yang dipenggal, diberi tambahan karakter *recording* Tabel 2.7dan Tabel 2.8. Jika terdapat kombinasi awalan yang dihilangkan sebelumnya atau tiga awalan yang telah dihilangkan maka proses berhenti
6. Tahap ke-6, jika semua Langkah sudah selesai, namun belum menemukan kata dasar, maka akan dikembalikan ke kata awal dan dianggap kata awal merupakan kata dasar, proses selesai.

Contoh proses *stemming* dapat dilihat pada Tabel 2.9

**Tabel** **2.9 *Stemming***

|  |  |
| --- | --- |
| Sebelum Stemming | [‘konten’,’bermutu’,’sayang’,’suaranya’,’kurang’,’jelas’] |
| Setelah Stemming | [‘konten’,’mutu’,’sayang’,’suara’,’kurang’,’jelas’] |

## 2.6 Klasifikasi

Klasifikasi merupakan proses mendapatkan model atau fungsi dari analisis atau mempelajari himpunan dokumen teks yang belum diketahui kelasnya, model atau fungsi dapat digunakan untuk mengkategorikan dokumen teks lain yang belum diketahui kelasnya kedalam satu atau lebih kelas yang telah ditentukan berdasarkan isinya (Sebastiani, 2002). Klasifikasi bertujuan untuk mengkategorikan kelas dari suatu dokumen teks yang belum diketahui kelasnya (Gaikwad et al., 2014). Berdasarkan jumlah label klasifikasi dapat dibagi menjadi 2 kategori, yaitu :

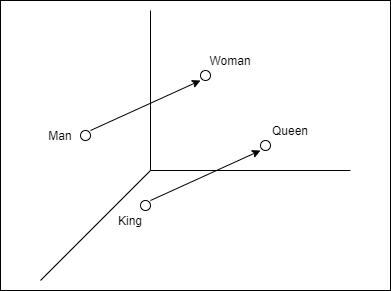
1. *Binaryclass*, mengklasifikasikan menjadi 2 kelas saja misalnya positif dan negatif, hoax dan fakta.
2. *Multiclass*, mengklasifikasikan menjadi lebih dari 3 kelas misalnya bahagia, sedih, marah.

## 2.7 Word Embedding

*Word embedding* merupakan suatu teknik memetakan kata-kata berdasarkan suatu kamus yang sudah ada sehingga menghasilkan vektor-vektor angka yang berisi angka rill (Halim et al., 2020), hal ini dilakukan karena *deep learning* tidak dapat memproses data *string* atau teks, oleh karena itu data harus diubah dari data *string* atau teks menjadi vektor angka menggunakan *word embedding*. *Word embedding* juga dapat menangkap makna semantik dan sintaktik kata (Nurdin et al., 2020). Hasil dari *word embedding* dapat digunakan ke dalam ruang geografis yang dinamakan ruang *embedding* (Faadilah, 2020). Penelitian yang dilakukan oleh Utomo (2020) mendapatkan algoritma LSTM memiliki akurasi yang lebih baik jika menggunakan *word embedding* sebesar 86,76%, sedangkan tanpa *word embedding* hanya mendapatkan akurasi sebesar 84,14% (Utomo, 2020). Misalnya terdapat data “indonesia tanah air ku” akan diubah menjadi vektor menghasilkan kamus kata [“indonesia”, ”tanah”, “air”, “ku”]. Metode *one-hot encoding* akan merepresentasikan kata dengan angka 1, dan angka 0 untuk kata lainnya dalam bentuk vektor. Sehingga kata “indonesia” akan direpresentasikan dalam vektor menjadi [ 1, 0, 0, 0], kata “tanah” menjadi [ 0, 1, 0, 0], kata “air” menjadi [ 0, 0, 1, 0], kata “ku” menjadi [ 0, 0, 0, 1]. Selain metode *one-hot encoding* terdapat metode lain seperti *word2vec*, GloVe, *fasttext*. Penelitian Nurdin (2020) membandingkan *word2vec*, *Global vectors for word representation* (Glo-Ve), *fasttext* dengan *Convolutional Neural Network* (CNN) untuk klasifikasi teks, menghasilkan *fasttext* mendapatkan nilai terbaik dengan f-measure 0.979, sedangkan word2vec mendapatkan nilai f-measure 0,925 dan GloVe mendapatkan nilai f-measure 0,958 (Nurdin et al., 2020).

## 2.8 Word2vec

*Word2vec* merupakan salah satu metode *word embedding* yang diperkenalkan oleh Mikolov dkk pada tahun 2013. *Word2vec* memetakan setiap kata ke dalam vektor yang dapat membawa makna semantik atau sintatikal dari kata tersebut. Proses memetakan kata pada *word embedding* menggunakan *unsupervised neural network* yang terdiri dari sebuah *hidden layer* dan *fully connected layer* (Nurdin et al., 2020). *Word2vec* memiliki kelebihan dalam similaritas makna kata yang didapatkan dari memperhatikan kesamaan kata-kata disekitar kata target (Lim et al., 2020), karena kelebihan itu metode *word2vec* sangat populer. Misalnya ada kata “king” cenderung dimiliki kata “man” maka kata “woman” akan sama artinya dengan kata “queen”, hubungan kata ini dapat dilihat pada gambar 2.1. *Word2vec* memiliki 2 teknik yaitu *continous bag of words* dan *skip gram model*.

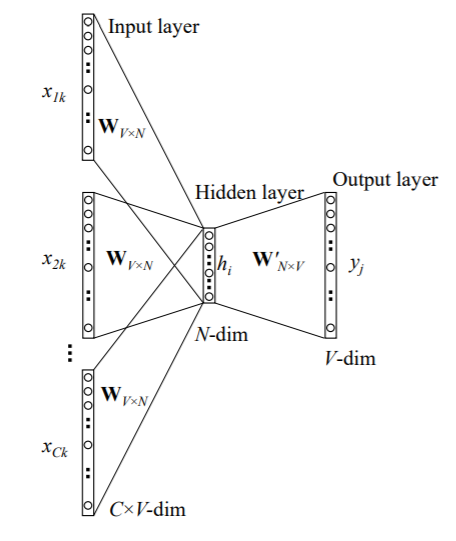


**Gambar 2.1 Hubungan antar kata *word2vec***

### 2.8.1 Continous bag of words

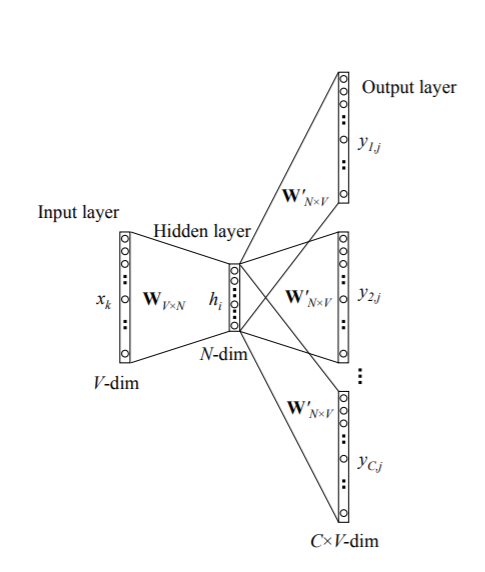
*Continous bag of words* (CBOW)merupakan salah satu arsitektur *word2vec* yang digunakan untuk memprediksi kata saat ini berdasarkan konteksnya (Utomo, 2020), kelebihan dari arsitektur CBOW adalah memiliki waktu *training* yang lebih cepat dan tingkat akurasi yang lebih baik untuk kata yang sering muncul. CBOW menggunakan 3 *layer* yaitu *input layer*, *hidden layer*, *output layer*. Langkah langkah CBOW mulai dari Kata yang akan masuk ke *input layer* akan dikonversikan terlebih dahulu ke bentuk vektor. Kata yang telah berbentuk vektor masuk ke *input layer* untuk pembobotan. Hasil pembobotan diproyeksikan ke *hidden layer*. Langkah terakhir menghitung *error* dengan membandingkan antara *output* dengan data target, tujuannya untuk memperbaiki nilai berdasarkan propagasi balik dari gradient kesalahannya. Ilustrasi CBOW dapat dilihat pada Gambar 2.2.

**Gambar 2.2 Ilustrasi CBOW**Sumber: (Rong, 2016)



### 2.8.2 Skip gram model

*Skip gram model* merupakan salah satu arsitektur *word2vec*, arsitektur ini merupakan kebalikan dari arsitektur CBOW, dimana arsitektur ini berguna untuk memprediksi konteks (*output*) yang berhubungan disekitar *current word* kata input, kelebihan arsitektur ini adalah dapat bekerja dengan baik meski data *training* hanya sedikit dan arsitektur ini mampu merepresentasikan kata yang jarang muncul atau langka. Ilustrasi *Skip gram model* dapat dilihat pada Gambar 2.3.



**Gambar 2.3 Ilustrasi *skip gram model***  
Sumber: (Rong, 2016)

## 2.9 Glo-Ve

*Global vectors for word representation* (Glo-Ve)merupakan salah satu metode *word embedding* yang mengandalkan *co-occurrence* kata atau statistik kemunculan kata dalam kumpulan kata atau korpus yang ditangkap langsung oleh model untuk memperoleh hubungan semantik antar kata dalam korpus. Glo-Ve menggunakan metode *global matrix factorization* yang mewakili jumlah kemunculan atau frekuensi dalam suatu korpus (Pennington et al., 2014). Alur kerja Glo-ve terdiri dari langkah-langkah sebagai berikut (Pennington et al., 2014) :

1. Bentuk matriks *word co-occurence* X dimana didalam matriks tersebut berisi kumpulan statistik *word co-occurence*. Matriks tersebut terdapat elemen Xij yang merepresentasikan jumlah kata *i* muncul dalam konteks kata *j*. Penjabaran langkah ini menurut Pennington (2014) dapat dilihat pada persamaan 2.1.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.1)

Keterangan :

= vektor kata utama.

= vektor kata konteks.

= probabilitas k dalam konteks i.

= matriks *word co-occurrence X*.

1. Langkah selanjutnya menentukan *soft constraints* untuk setiap pasangan kata, rumus *soft constraints* menurut Pennington (2014) dapat dilihat pada persamaan 2.1.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.2)

Keterangan :

= vektor kata utama.

= vektor kata konteks.

= bias skalar kata utama.

= bias skalar kata konteks.

= matriks *word co-occurrence X*.

1. Langkah selanjutnya menentukan *cost function*, rumus *cost function* menurut Pennington (2014) dapat dilihat padapersamaan 2.3.

. . . . . . . . . . . . . . . . . . . . . . . . (2.3)

Keterangan :

= vektor kata utama.

= vektor kata konteks.

= bias skalar kata utama.

= bias skalar kata konteks.

= matriks *word co-occurrence X*.

= fungsi pembobotan.

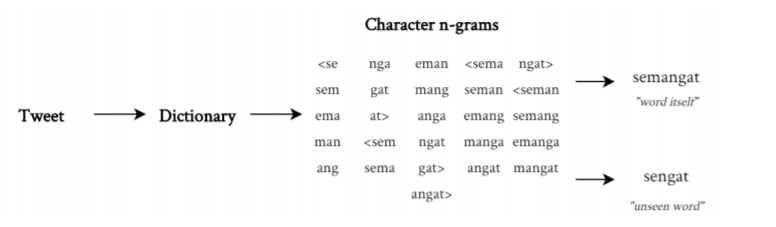
Fungsi pembobotan yang berguna untuk mencegah pembelajaran hanya dari pasangan kata yang sangat umum. Rumus *f* menurut Pennington (2014) dapat dilihat pada persamaan 2.4.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2. 4)

## 2.10 Fasttext

*Fasttext* merupakan metode *word embedding* pengembangan dari *word2vec*. *Fasttext* memliki kelebihan yaitu dapat menangani masalah *out of vocabulary* dimana masalah ini tidak dapat diselesaikan oleh *word embedding word2vec* dan *Glo-Ve*, *fasttext* dapat menangani masalah tersebut karena ketika ada kata yang tidak muncul selama *training* model sehingga tidak ditemukan *embedding* vektornya, maka kata tersebut akan dipecah menggunakan *n-gram* untuk mendapatkan *embedding* vektornya. Sedangkan pada *word embedding* lain jika terdapat kata yang tidak muncul selama *training* model, maka akan menghasilkan *error*. *Fasttext* mempelajari representasi kata dengan memperhatikan informasi *subword* menggunakan *n-gram* ke dalam model skipgram. Hal inilah yang membuat *fasttext* dapat menangkap kata-kata yang lebih pendek dan memahami sufiks dan prefix dari kata (Bojanowski et al., 2017). Langkah- langkah fasttext menurut Bojanowski (2017) sebagai berikut :

1. Representasikan nilai vektor kata dengan menjumlahkan nilai dari tiap *n-gram*. *N-gram* pembentuk kata ini yang memungkinkan memungkinkan muncul pada *n-gram* yang berada dalam korpus, sehingga dapat menangani kata yang belum pernah ditemui. Misalnya n = 3, dalam kata semangat menghasilkan vektor kata pada gambar 2.4.



**Gambar 2.4 Vektor kata semangat**

Sumber: (Alfariqi et al., 2020)

Representasi vektor kata menurut Bojanowski (2017) dapat dihitung menggunakan persamaan 2.5.

Keterangan :

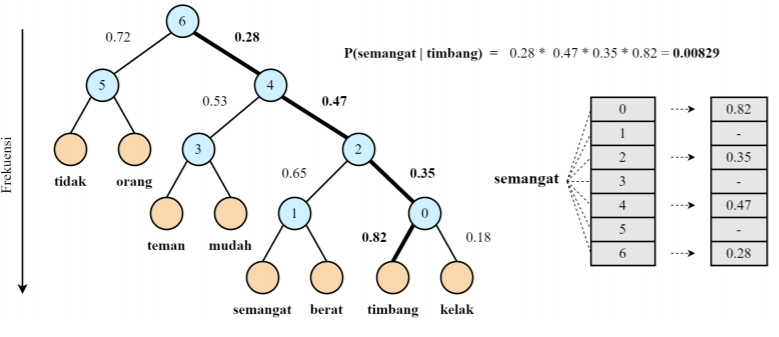
hw = kata yang ingin dicari vektor representasinya.

Gw = kumpulan *n-gram* yang muncul pada kata tersebut.

zg = representasi vektor tiap *n-gram*.

vc = vektor kata dari *context word* c.

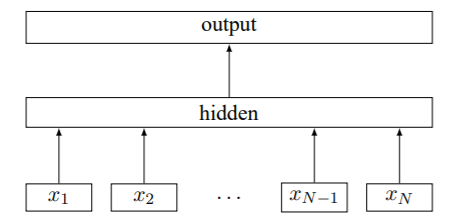
1. Langkah selanjutnya, masuk lapisan *Hierarchical Softmax* dengan metode *Huffman Tree*. Metode ini membangun *binary tree* yang memposisikan *leaf node* dengan *frekuensi* kemunculan sedikit pada level kedalaman yang lebih tinggi, dan sebaliknya. Metode ini juga mempercepat proses komputasi. Cara kerja *Hierarchical Softmax* dapat dilihat pada Gambar 2.5.



**Gambar 2.5 Cara kerja *Hierarchical Softmax***

Sumber: (Alfariqi et al., 2020)

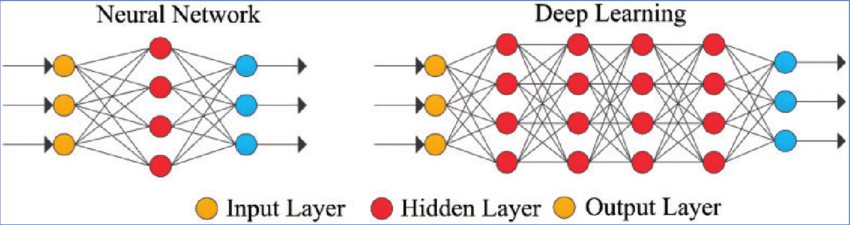
Arsitektur *fasttext* dapat dilihat pada Gambar 2.6.



**Gambar 2.6 Arsitektur *fasttext***Sumber: (Joulin et al., 2016)

## 2.11 Deep Learning

*Deep learning* merupakan salah satu bidang *machine learning* berbasis jaringan syaraf tiruan atau *neural network*, *deep learning* terdiri dari beberapa *hidden layer* yang membentuk tumpukan (Nurfita dan Ariyanto, 2018). *Deep learning* membutuhkan waktu yang sedikit untuk melakukan *training* karena *deep learning* mampu menangani masalah gradien semakin rendah pada propagasi balik (Ahmad, 2017). Arsitektur *deep learning* dapat dilihat pada gambar 2.5.



**Gambar 2.7 Arsitektur *deep learning***

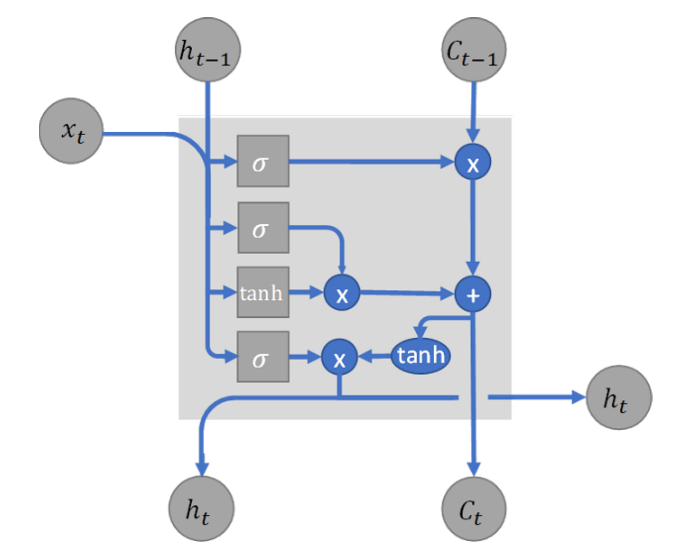
Sumber: (Xing and Du, 2019)

*Deep learning* efektif dalam menangani berbagai permasalahan dengan data skala besar seperti *speech recognition*, *image recognition*, *natural language comprehension*, *playing games*, *biomedical applications* (Temizel et al., 2020). Hasil *training* pada *deep learning* berupa model yang digunakan untuk prediksi. *Deep learning* memiliki kelebihan yaitu, model yang dihasilkan dapat membedakan suatu kelas dengan mudah, hal ini karena *deep learning*  memiliki *feature engineering* yang berfungsi untuk mengekstrak pola yang penting dari suatu data. Contoh algoritma yang menerapkan *deep learning* seperti *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), *Long Short Term Memory* (LSTM).

## 2.12 Long Short Term Memory

*Long Short Term Memory* (LSTM) merupakan algoritma *deep learning* pengembangan dari arsitektur RNN. RNN memiliki masalah *vanishing gradient*, LSTM mampu mengatasi masalah *vanishing gradient* dengan *memory cell* dan *gate units* (*input gate, forget gate, output gate*) sehingga LSTM dapat membaca, menyimpan, dan memperbaharui informasi (Rao dan Spasojevic, 2016). Arsitektur LSTM dapat dilihat pada gambar 2.8.

**Gambar 2.8 Arsitektur LSTM**



Sumber: (Miedema, 2018)

Pada gambar 2.8 terdapat beberapa simbol yang mewakilkan proses LSTM, seperti persegi abu-abu melambangkan *layer neural network*, lingkaran biru melambangkan operasi *pointwise*, garis horizontal pada kanan diagram adalah *cell state*.

Terdapat 3 jenis *gate* pada LSTM yaitu, *forget gate*, *input gate*, *output gate*. *Forget gate* berfungsi untuk menentukan informasi mana yang dihapus dari *cell*. *Input gate* berfungsi untuk menentukan nilai dari *input* untuk diperbaharui pada *state memory*. *Output gate* berfungsi untuk menentukan *output* berdasarkan *input* dan memori pada *cell* (Utomo, 2020). Langkah-langkah LSTM sebagai berikut (Miedema, 2018) :

1. Langkah pertama, LSTM akan menentukan informasi mana yang harus dibuang dari *cell state*, bagian ini disebut dengan *forget gate*. Langkah ini menggunakan *output* dari langkah sebelumnya (ht-1) dan *input* (xt) untuk diproses dengan fungsi aktivasi *sigmoid*. pada proses ini akan menghasilkan *output* berupa nilai 0 atau nilai 1 pada *cell state Ct-1*, nilai 0 berarti informasi akan disingkirkan, sedangkan nilai 1 berarti akan dipertahankan. Persamaan *forget gate* menurut Hochreiter dan Schmidhber (1997)diuraikan pada persamaan 2.6, sedangkan alur informasi pada *forget gate* dapat dilihat pada gambar 2.9.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.6)

Keterangan :

*ft*  : *forget gate*

σ : fungsi *sigmoid*

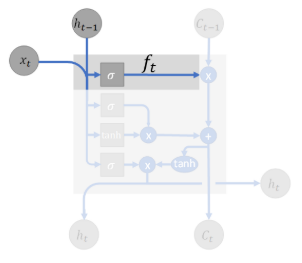
*Wf* : *weight* pada *forget gate*

*Uf* : *recrrent weight* pada *forget gate*

*ht-1*: *output* pada langkah sebelum orde t

*xt* : nilai input pada orde t

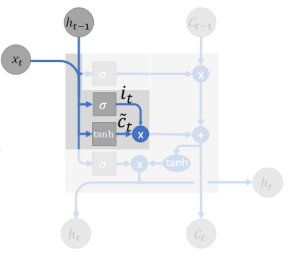
*bf* : nilai bias pada *forget gate*



**Gambar 2.9 Alur informasi pada *forget gate***

Sumber: (Miedema, 2018)

1. Langkah kedua, menentukan informasi yang akan ditambahkan pada *cell state*. Pada langkah ini memiliki memiliki dua bagian, yang pertama *layer sigmoid* dinamakan *input gate layer* berfungsi untuk menentukan nilai mana yang akan diperbaharui. Selanjutnya, *layer tanh* membuat nilai kandidat baru (𝐶̌𝑡)untuk dimasukkan ke *cell state*. *Output* dari *input gate layer* dan *tanh layer* akan digabungkan untuk memperbaharui *cell state*. Proses pada langkah ini digambarkan pada gambar 2.10. Sedangkan, menurut Hochreiter dan Schmidhber (1997) persamaan *input gate layer* dan persamaan nilai kandidat konteks dapat dilihat pada persamaan 2.7 dan persamaan 2.8.



**Gambar 2.10 Proses *input gate layer***

Sumber: (Miedema, 2018)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(2.7)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.8)

Keterangan :

*it* : *input gate*.

𝐶̌𝑡 : kandidat baru.

𝜎 : fungsi *sigmoid*.

*Wi* : *weight* pada *input gate*.

*Ui* : *recurrent weight* pada *input gate*.

*bi* : nilai bias pada *input gate*.

*Wc* : *weight* pada kandidat konteks.

*Uc* : *recurrent weight* pada kandidat baru.

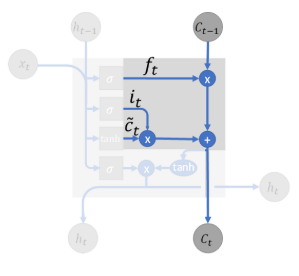
*bc* : nilai bias pada kandidat konteks.

*ht-1* : nilai *output* sebelum orde t.

*xt* : nilai *input* pada orde t.

1. Langkah ketiga, memperbaharui *cell state* lama *Ct-1* menjadi *cell state* baru *Ct*. Pada langkah sebelumnya, telah didapatkan nilai *input gate* dan kandidat konteks. Langkah ketiga akan terjadi proses perkalian *cell state* lama *Ct-1* dengan nilai *forget gate ft*, kemudian hasil perkalian *cell state* lama Ct-1 dengan nilai *forget gate* ditambah dengan hasil perkalian nilai *input gate* dengan nilai kandidat konteks 𝐶̌𝑡, akan menghasilkan *cell state* baru (*Ct*). Proses ini diuraikan pada gambar 2.11. Persamaan pada langkah mendapatkan *cell state* baru (*Ct*) menurut Hochreiter dan Schmidhber (1997) dapat dilihat pada persamaan 2.9.

**Gambar 2.11 Langkah mendapatkan nilai *cell state* baru**



Sumber: (Miedema, 2018)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(2.9)

Keterangan :

*Ct* : *cell state* orde t

*ft* : *forget gate*

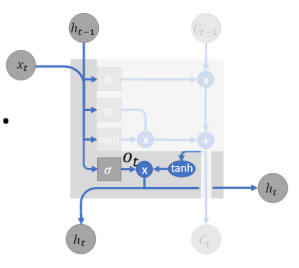
*Ct-1* : *cell state* sebelum orde t

*it* : nilai *input* pada orde t

𝐶̌𝑡 : kadidat konteks

1. Langkah terakhir, menentukan *output* hasil keseluruhan proses (ht). Pertama *layer sigmoid* digunakan *output* sebelumnya (ht-1) dan *input* (xt) untuk menentukan nilai *output gate* (Ot), nilai *output gate* (Ot) antara 0 dan 1 yang menunjukkan bagian yang menjadi *output* dari *cell state*, selanjutnya *cell state* (Ct) diubah menggunakan fungsi aktivasi *tanh*, *cell state* ini mendapatkan nilai antara -1 dan 1. Nilai *cell state* kemudian dikalikan dengan nilai *output gate* (Ot) menghasilkan nilai *output* (ht). Proses ini dijelaskan pada gambar 2.12. Persamaan proses ini menurut Hochreiter dan Schmidhber (1997) dijabarkan pada persamaan 2.10 dan persamaan 2.11.

**Gambar 2.12 *Output gate***



Sumber: (Miedema, 2018)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(2.10)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(2.11)

Keterangan :

*Ot* : *output gate*

𝜎 : fungsi *sigmoid*

*Wo* : nilai *weight* untuk *output gate*

*Uo*: *recurrent weight* pada *output gate*

*ht-1* : nilai *output* sebelum orde t

*xt* : nilai *input* orde t

*bo*: nilai bias pada *output gate*

*Ct* : *cell state*

Dari penjelasan proses LSTM, terdapat dua fungsi aktivasi yaitu, *sigmoid* dan *tanh*. Fungsi aktivasi *sigmoid* dijabarkan pada persamaan 2.12, sedangkan fungsi aktivasi *tanh* dijabarkan pada persamaan 2.13.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.12)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.13)

Keterangan :

𝜎 : fungsi *sigmoid*.

x : data input.

e : fungsi eksponensial.

## 2.13 Confusion Matrix

*Confusion matrix* merupakan cara untuk menguji keakuratandengan menghitung nilai *precision*, *recall*, *accuracy*. Menurut Han dan Kamber (2006) *confusion matrix* merupakan alat untuk menganalisis *classifier* mengenali *tuple* dari kelas yang berbeda dengan baik atau tidak. Terdapat empat istilah pada *confusion matrix* yang merepresentasikan hasil, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (FN). Tabel ilustrasi *confusion matrix* dapat dilihat pada Tabel 2.10.

**Tabel 2.10 Ilustrasi *confusion matrix***

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Actual Values | |
| Positive | Negative |
| Predicted Values | Positive | True Positive (TP) | False Negative(FN) |
| Negative | False Positive (FP) | True Negative (TN) |

Keterangan :

*True Positive* (TP) : jumlah data positive yang diprediksi benar.

*True Negative* : jumlah data negative yang diprediksi benar.

*False Positive* : jumlah data negatif yang diprediksi salah.

*False Negative* : jumlah data positive yang diprediksi salah.

Berdasarkan tabel *confusion matrix* maka dapat digunakan untuk mengukur *accuracy*, *precision*, *recall*. Menurut Han dan Kamber (2006) persamaan *accuracy, precision*, *recall* dijabarkan sebagai berikut (Han dan Kamber, 2006):

1. *Accuracy*

*Accuracy* merupakan nilai yang merepresentasikan tingkat kedekatan antara data yang diprediksi dengan data aktual. Perhitungan *accuracy* dengan membandingkan data prediksi benar dengan keseluruhan data. Rumus *accuracy* dapat dilihat pada persamaan 2.14.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.14)

1. *Precision*

*Precision* merupakan perbandingan antara data yang diprediksi benar *positive* dengan data yang diprediksi *positive*. Rumus untuk menghitung *precision* dapat dilihat pada persamaan 2.15.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(2.15)

1. *Recall*

*Recall* merupakan perbandingan antara data yang diprediksi benar *positive* dengan data yang benar *positive*. Rumus untuk menghitung *recall* dapat dilihat pada persamaan 2.16.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.16)

## 2.16 Penelitian Sebelumnya

Penelitian yang dilakukan oleh Fanesya dkk (2019) menggunakan metode *naïve bayes* dan kombinasi fitur *n-gram*, *dataset* yang digunakan berasal dari *twitter* dan menggunakan model Ekman, penelitian ini menghasilkan akurasi tertinggi 55,5%.

Penelitian yang dilakukan oleh Rohman dkk (2019) membandingkan metode *lexicon* dengan *emolex* dengan metode *Natural Language Processing* (NLP), *dataset* yang digunakan berasal dari *facebook* dan model emosi yang digunakan adalah model Ekman. Penelitian yang dilakukan oleh Rohman dkk (2019) menghasilkan akurasi 55,54% dari 34.872 kata untuk lexicon dengan emolex, sedangkan untuk NLP menghasilkan akurasi 61,53%.

Rohman dkk (2019) melakukan penelitian dengan *dataset* yang berasal dari *facebook* dan model emosi yang digunakan adalah model Ekman, penelitian Rohman dkk (2019) menggunakan metode *Term Frequency-Inverse Document Frequency* (TF-IDF) dengan akurasi 59%.

Penelitian yang dilakukan oleh Ardiada dkk (2019) membandingkan metode *Support Vector Machine* (SVM) dengan metode *K-Nearest Neighbor* (KNN), *dataset* yang digunakan berasal dari *twitter* dan model emosi yang digunakan model 6 emosi (jijik, malu, marah, sedih, senang, takut). Penelitian yang dilakukan oleh Ardiada dkk (2019) menghasilkan *precision* 45,64%, *recall* 50,20%, dan akurasi 81,04% untuk metode SVM, sedangkan KNN *precision* 34,21%, *recall* 45,95%, serta akurasi 79,70%.

Penelitian yang membandingkan SVM tidak hanya dilakukan oleh Ardiada dkk (2019), namun juga dilakukan oleh Haryadi dkk (2019). Penelitian Haryadi dkk (2019) membandingkan metode *Support Vector Machine* (SVM), *Long Short Term Memory* (LSTM) dengan *word embedding word2vec* dan *Nested Long Short Term Memory* (Nested LSTM) dengan *word embedding word2vec*. Penelitian ini menggunakan *dataset* dari *twitter* dan model 7 emosi (marah, takut, bahagia, cinta, sedih, terkejut, bersyukur). Penelitian ini menghasilkan sebagai berikut, SVM memiliki akurasi 98,679%, *precision* 98,53%, *recall* 98,22%, *f1-score* 98,37%. Nested LSTM akurasi 99,167%, *precision* 99,21%, *recall* 98,83%, *f1-score* 99,02%. LSTM memiliki akurasi 99,154%, namun memiliki *precision*, *recall*, *f1-score* paling tinggi dengan *precision* 99,22%, *recall* 98,86%, *f1-score* 99,04%.

Penelitian yang menggunakan metode LSTM juga pernah dilakukan oleh Dwi dkk (2020), penelitian tersebut menggunakan metode LSTM dengan *word embedding Global vector representation for words* (Glo-Ve). *Dataset* yang digunakan berasal dari *twitter* dan model emosi yang digunakan adalah model 5 emosi (marah, cinta, sedih, takut, bahagia). Penelitian ini hanya menghasilkan akurasi 50%, *precision* 33%, *recall* 38%, *f-1 score* 35%, hal ini terjadi karena selama proses *training* terjadi *underfitting*.

Rangkuman penelitian yang telah dijabarkan diatas dapat dilihat pada tabel 2.11 dan tabel 2.12.

**Tabel 2.11** **Penelitian sebelumnya**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Penulis | Metode | Dataset dan Model | Hasil | Keterangan |
| 1 | Fera Fanesya, Randy Cahya Wihandika, Indriati (2019) | Naïve bayes dan kombinasi fitur n-gram | *Twitter* dan model Ekman | Penelitian ini menghasilkan akurasi 55,5%. |  |
| 2 | Rohman, Utami, Raharjo (2019) | Lexicon dengan emolex dan *Natural Language Processing* | *facebook* dan Ekman | Akurasi 55,54% dari 34.872 kata untuk lexicon dengan emolex, NLP menghasilkan akurasi 61,53% |  |
| 3 | Arif Nur Rohman, Riska Dwi Handayani, Ryan Dwi Y.P, Kusrini (2019) | *Term Frequency-Inverse Document Frequency* | *facebook* dan Ekman | Akurasi 59% dari 6 emosi | hal ini karena jumlah data antar emosi tidak sama, sehingga peneliti menguji dengan 1 emosi saja dan menghasilkan 87,23% |

**Tabel 2.12** **Penelitian sebelumnya(lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Penulis | Metode | Dataset dan Model | Hasil | Keterangan |
| 4 | Ardiada, Sudarma, Giriantari (2019) | *Support Vector Machine*(SVM) dan *K-Nearest Neighbor*(KNN) | *Twitter* dan model 6 emosi(jijik, malu, marah, sedih, senang, takut). | *precision* 45,64%, *recall* 50,20%, dan akurasi 81,04% untuk SVM, sedangkan KNN *precision* 34,21%, *recall* 45,95%, serta akurasi 79,70% |  |
| 5 | Daniel Haryadi, Gede Putra Kusuma (2019) | *Support Vector Machine* (SVM), *Long Short Term Memory* (LSTM) dengan *word embedding word2vec* dan *Nested Long Short Term Memory* (Nested LSTM) dengan *word embedding word2vec* | *twitter* dan model 7 emosi (marah, takut, bahagia, cinta, sedih, terkejut, bersyukur) | SVM memiliki akurasi 98,679%, *precision* 98,53%, *recall* 98,22%, *f1-score* 98,37%. Nested LSTM akurasi 99,167%, *precision* 99,21%, *recall* 98,83%, *f1-score* 99,02%. LSTM memiliki akurasi 99,154%, namun memiliki *precision*, *recall*, *f1-score* paling tinggi dengan *precision* 99,22%, *recall* 98,86%, *f1-score* 99,04% |  |
| 6 | Adinda Dwi L, Budi Harijanto, Faisal Rahutomo (2020) | *Long Short Term Memory* (LSTM) dengan *word embedding Global vector representation for words* (Glo-Ve) | *Twitter* dan model 5 emosi (marah, cinta, sedih, takut, bahagia) | Akurasi 50%, *precision* 33%, *recall* 38%, *f-1 score* 35%. | Hasil penelitian kurang maksimal karena selama proses *training* terjadi *underfitting*. |
| 7 | M. Alfa Riza | *Long Short Term Memory* (LSTM) dengan *word embedding Fasttext* | *Twitter* dan model ekman |  |  |

Penelitian terdahulu memiliki kesamaan dalam tema penelitian dan objek penelitian, namun pada penelitian ini memiliki beberapa perbedaan, yaitu :

1. Penelitian sebelumnya belum menangani masalah kesalahan pengetikan (OOV).
2. Penelitian ini menggunakan *word embedding fasttext* untuk menangani kesalahan pengetikan (OOV).