BAB 2

LANDASAN TEORI

2.1 Difabel atau Disabilitas

Disabilitas adalah keadaan penurunan atau kelainan fungsi pada fisik, indera, intelektual atau psikososial, yang dikarenakan sebuah penyakit, cedera, atau kondisi kesehatan lainnya yang berdampak pada pembatasan aktivitas, atau pembatasan partisipasi (Matilde Leonardi *et al*, 2006; ILO, 2014). Menurut Lisa dan Diana (2017), beberapa kelainan fungsi tubuh yang dapat disebut disabilitas yaitu.

1. Kesulitan melihat, walaupun menggunakan kacamata.
2. Kesulitan mendengar, walapun sudah menggunakan alat bantu dengar.
3. Kesulitan dalam berjalan atau memanjat.
4. Kesulitan untuk mengingat atau berkonsentrasi.
5. Kesulitan untuk merawat diri sendiri seperti kebersihan tubuh dan berpakaian.
6. Kesulitan dalam berkomunikasi seperti sulit untuk memahami dan dipahami.

2.2 Text Preprocessing

Text preprocessing merupakah tahap pengolahan data untuk membersihkan data dari kata-kata, karakter atau tanda baca yang tidak berguna untuk diolah oleh mesin (Tabassum & Patil, 2020). Teknik *Text Preprocessing* yang digunakan dalam penelitian ini adalah *Tokenization*, *Stop Words Removal, Stemming*, *Back-Translation*.

1. *Tokenization*

   Tokenization adalah proses untuk memecah sebuah kalimat menjadi kata, kriteria dalam proses pemecahan kata adalah dari tanda baca dan spasi. Seperti "Saya suka pemrograman!" yang akan diolah menjadi "Saya", "suka", "pemrograman", "!" (Tabassum & Patil, 2020).

2. *Stop Words Removal*

   Stop Words Removal adalah daftar dari kata-kata yang sering muncul disuatu dokumen dan tidak dapat digunakan untuk mendapatkan konteks dari sebuah kalimat, daftar dari kata-kata yang dimiliki akan berbeda-beda tergantung dari bahasa yang digunakan (Kadhim, 2018; Tabassum & Patil, 2020). Daftar kata-kata yang digunakan untuk *stop words removal* seperti "ada", "adalah", "anda", "empat", "jelas", "saat", dll.

3. *Stemming*

   Stemming adalah proses pengolahan kata yang menghapus imbuhan dari sebuah kata dan hanya menghasilkan kata dasarnya saja (Kadhim, 2018; Tabassum & Patil, 2020). Seperti "Menyesal" dan "Terakhir" yang akan menjadi "sesal dan "akhir".

4. *Back-Translation*

   Back-Translation adalah proses augmentasi data. *Back-Translation* bekerja dengan cara mengartikan sebuah kalimat ke bahasa lain lalu dimengubah kembali ke bahasa sebelumnya (Edunov, Ott, Auli, & Grangier, 2020).

Proses dari *augmentation* menggunakan dua teknik yaitu pertama dengan meng-*augmentation* semua data *train* (Yu, et al., 2018), dan melakukan proses *random oversampling* dengan meng-*augmentasi* data *train* sampai mendekati jumlah dari kategori dengan data terbanyak (Padurariu & Breaban, 2019).

2.3 Word Embedding dengan FastText

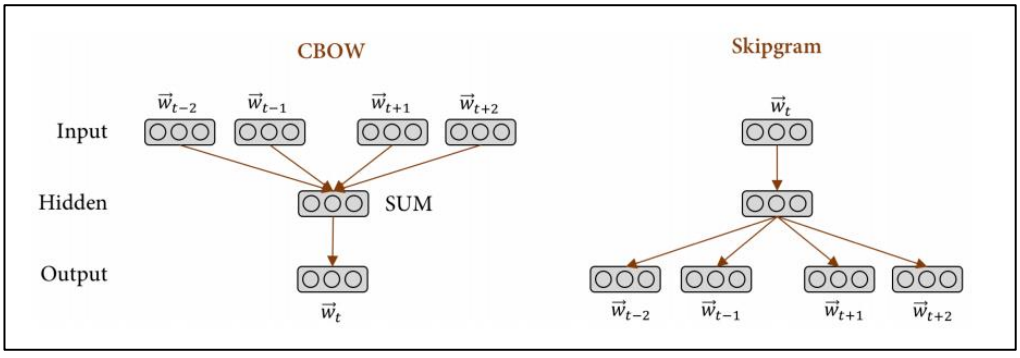
Word Embedding adalah numerical yang mempresentasikan sebuah kata-kata, *word embedding* merupakan metode *unsupervised* yang mempelajari representasi vektor-vektor yang mempunyai kesamaan korelasi yang relatif dengan kesamaan semantik (Amit, 2016). Salah satu *pre-trained* model yang dapat digunakan adalah FastText.

FastText adalah *library* yang digunakan untuk pembelajaran yang effesien dari representasi kata dan klasifikasi kalimat yang dibuat oleh Facebook AI Research (FAIR) (Fasttext.cc, 2019). FastText dapat digunakan untuk membuat *vector* dari semua kata (*Feature Extraction*), termasuk kata-kata yang memiliki kesalahan penulisan karena FastText menciptakan *vector* kata berdasarkan *subwords* yang terdapat pada kata tersebut (Fasttext.cc, 2019). FastText akan menghasilkan model *pre-trained* yang akan digunakan untuk mengubah kata-kata yang ada maupun tidak ada di dalam kamusnya karena FastText mempelajari suku kata dari sebuah kata-kata yang dipelajari mulai dari *prefix* dan *suffix* dalam sebuah kata. FastText dapat mengenali kata-kata yang tidak ada dalam kamusnya dengan cara membuat sebuah kata-kata menjadi *subword*-nya sesuai dengan nilai n-gramnya, seperti contoh kata "*Cities*" dengan n-gram=4, maka kata tersebut akan berubah menjadi beberapa *subword* sebagai berikut <cit>, <citi>, <itie>, <ties>, dan <ies> sehingga jika ada input kata "*Cities"* akan menjumlahkan semua vektor

dari semua *subwords*nya, tetapi jika ada input seperti "*Citiez"* maka *subword* yang diambil adalah <cit>, <citi>, dan <itie>.

FastText juga mempunyai parameter *window* untuk mempelajari konteks dari sebuah kata, contoh *windows* bernilai 1 dengan n-gram 3, maka jika ada kata kalimat "*i love programming*" dengan target kata *love* maka hasil konteks wordnya akan menjadi "<i>, <pr>, <pro>, <rog>, <ogr>, <gram>, <ram>, <amm>, <mmi>, <min>, <ing>, dan <ng>". Semakin banyak kata yang dipelajari oleh FastText maka akan semakin baik model tersebut, FastText menyedeiakan model *pre-trained* menggunakan 157 bahasa berdasarkan *common crawl* dan 1 leighbour.

FastText mempunyai 2 model komputasi yaitu skip-gram dan *continuous-bag-of-words* (CBOW) (Fasttext.cc, 2019). Menurut Camacho-Collados dan Pilehvar (2018) model skip-gram menggunakan sebuah kata untuk memprediksi konteks dari kata tersebut, akan tetapi CBOW menggunakan konteks untuk memprediksi target kata. Ilustrasi model skip-gram dan CBOW terdapat pada Gambar 2. 1.



Gambar 2.1 Model Komputasi CBOW dan Skip-Gram (Camacho-Collados dan Pilehvar, 2018)

2.4 Logistic Regression

Logistic Regression adalah perkembangan dari teknik *Linear Regression* untuk menghasilkan *output* variabel yang terkateogori (Umniy dan Desi, 2018). Hal

yang membedakan Logistic Regression dengan Linear Regression adalah Logistic Regression akan menghasilkan *ouput* garis lengkung atau disebut sigmoidal sedangkan *Linear Regression* hanya garis lurus (Peng, Lee, & Ingersoll, 2002). Perhitungan Logistic Regression dapat dilihat pada persamaan 2.1 (Bahovec, 2013; Shrestha, 2019; Telnoni, et al., 2019).

$$P(X) = \frac{e^{\alpha+\Sigma\beta_i X_i}}{1 + e^{\alpha+\Sigma\beta_i X_i}} \quad ...(2.1)$$

Dari persamaan 2.1, P(X) adalah hasi perhitungan probabilitas dari Logistic Regression, variabel X_i merupakan variabel independen yang mendukung dari hasil probabilitas, sedangkan α dan β merupakan nilai dari estimator yang didapat dari metode yang disebut *maximum likelihood*. *Maximum likelihood* merupakan metode pendekatan statistik untuk menentukan estimasi parameter di model matematika. Sebelum menggunakan metode *maximun likelihood* dalam proses perhitungan Logistic Regression mempunyai kendala dimana hanya mempunyai y-axis atau garis y yang hanya menghasilkan probabilitas yang berarti hasil hanya diantara angka 0 dan 1, maka y-axis harus diubah dari probabilitas menjadi grafik garis dengan y-axis dari *Log(odds)* mempunyai nilai tidak terbatas yang dapat dilihat pada persamaan 2.2.

$$\log(odds) = \left(\frac{P}{1-p}\right) \quad ...(2.2)$$

Setelah y-axis mempunyai jarak nilai yang tidak terbatas, maka Logistic Regression akan membuat garis lurus yang menjadi kandidat dari garis dari model, lalu data akan mengikuti garis tersebut dan setiap data mempunyai nilai log(odds)-nya masing-masing. Pada tahap ini, grafik garis yang dipunyai masih mempunyai nilai y-axis dengan rentang nilai tak terbatas, maka harus mengubahnya kebali

menjadi y-axis dengan rentang nilai probabilitas 0 hingga 1, rumus yang dapat digunakan dapat dilihat pada persamaan ketiga.

$$p = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}} \quad ...(2.3)$$

Dari persamaan 2.3, menggunakan nilai log(odds) pada persamaan 2.2, sehingga nilai p akan membentuk kurva melengkung pada rentang nilai 0 hingga 1 di y-axis. Semua data yang telah menjadi grafik kurva akan diitung log dari probabilitasnya lalu di tambah menjadi *log*(*likelihood*). Dengan hasil log(*likelihood*) dapat menggunakan perkiraan *chi square* sehingga mendapatkan *likelihood* ratio. Logistic Regression akan beberapa kali membuat garis sebagai kandidat dari garis yang akan digunakan pada modelnya, dengan menggunakan *likelihood ratio* sebagai perbandingannya untuk menemukan estimator terbaik.

2.5 *Confusion Matrix*

Confusion matrix adalah teknik yang biasa digunakan di proses klasifikasi untuk mengukur tingkat akurasi dari model klasifikasinya (Liza dan Rinaldo, 2016). *Confusion Matrix* dibagi menjadi dua permasalahan kelas yaitu kelas yang sebenarnya dan kelas yang diprediksi (Jasmina *et al*, 2017). Confussion matrix dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Confusion Matrix*

		Predicted	
		Positive	Negative
Actual	Positive	Tp	Fn
	Negative	Fp	Tn

Penjelasannya Tp (*True Positive*) adalah angka dari hasil yang diperdiksi positif dan itu benar, Tn (*True Negative*) adalah angka dari hasil yang diprediksi negative dan itu benar, Fp (*False Positive*) adalah angka dari hasil yang diprediksi

positif dan itu salah, dan Fn (*False Negative*) adalah angka dari hasil yang diprediksi negatif dan itu salah (Siraj-Ud-Doulah & Alam, 2020). Dari Confussion Matrix dapat dihitung sebagai berikut.

- *Accuracy* adalah seberapa besar nilai model mengklasifikasi dengan benar. Perhitungan *Accuracy* seperti pada persamaan 2.4.

$$Accuracy = \frac{\sum_{i=1}^{n} tpi}{\mathrm{N}} \quad ...(2.4)$$

- *Precision* adalah sebarapa besar akurasi antara data yang diinginkan dengan hasil prediksi. Perhitungan *Precision* seperti pada persamaan 2.5.

$$Precision = \frac{\mathrm{TP}}{\mathrm{Tp} + \mathrm{Fp}} \quad ...(2.5)$$

- *Recall* adalah seberapa besar hasil prediksi model yang benar dibandingkan seluruh data yang benar positif. Perhitungan *Recall* seperti pada persamaan 2.6.

$$Recall = \frac{\mathrm{TP}}{\mathrm{Tp} + \mathrm{Fn}} \quad ...(2.6)$$

- F1 *Score* adalah perbandingan dari rata-rata *Precision* dan *Recall*. Perhitungan F1 *Score* dapat dilihat pada persamaan 2.7.

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

Model yang telah dibuat akan diukur dengan menggunakan *Precision, Recall*, dan *f1-score* dimana F1 score berguna untuk menghitung akurasi dari jumlah data dengan kelas atau label yang tidak merata, sedangkan *Accuracy* menghitung nilai keseluruhan hasil klasifikasi (Kotsiantis, et al., 2006; Ali, et al., 2015).