BAB 2
LANDASAN TEORI

2.1 Bilingual Hate Speech

Hate speech adalah tindakan sengaja dan dengan kesadaran untuk menghina sekelompok orang tertentu, berdasarkan ras, agama, gender, disabilitas, ataupun identitas gender [16]. Sementara *bilingual hate speech* adalah seuatu fenomena dimana seseorang yang bisa menguasai dua bahasa melakukan tindakan menghina terhadap seseorang atau suatu kelompok dengan target tertentu, dengan menggunakan dua bahasa secara bersamaan pada suatu kalimat. Fakta bahwa dunia akademis juga tertarik dalam *hate speech* telah menunjukkan pertumbuhan yang konsisten sejak tahun 2014, hal ini terlihat dari peningkatan volume produksi yang terindeks dalam Web of Science (WoS), yang meningkat dari 42 menjadi 162 antara tahun 2013 dan 2018 [17]. Sedangkan di Indonesia fenomena *hate speech* sudah lama terjadi, tepatnya pada tahun 2008 oleh pemerintah yang di tetapkan pada Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik(UU ITE) [18].

Beberapa contoh dari bentuk *bilingual hate speech* adalah sebagai berikut dengan bahasa Indonesia dan Inggris adalah ”lu kenapa sih kalo main noob banget”. Dari teks tersebut dapat disimpulkan bahwa suatu individu melakukan ujaran kebencian atau hinaan terhadap seseorang yang memiliki kekurangan dalam keahlian, dengan menggunakan istilah bahasa Inggris. Sedangkan salah satu contoh bentuk *bilingual* non-*hate speech* adalah ”memang ya kalo positive thinking itu bisa menenangkan jiwa overthinking”. Pada teks tersebut istilah Inggris digunakan untuk mengujarkan seuatu tindakan dan sekelompok orang tertentu, kesimpulan dari teks tersebut yang didapat adalah berpikir secara positif bisa membantu sekelompok orang yang memiliki masalah terlalu memikirkan suatu hal ke arah negatif.

Beberapa tantangan dalam mendeteksi *hate speech* adalah *hate speech* memiliki banyak bentuk baik secara verbal, non-verbal, dan secara simbolis, lalu *hate speech* bisa diutarakan secara ambigus dengan banyak bentuk dan dengan banyak bahasa [17]. Selain itu *hate speech* terbagi menjadi kalimat yang mengandung kekerasan atau menyinggung, dan *hate speech* terbagi dalam targetnya sepert diskriminasi ras, diskriminasi gender, diskriminasi agama, homofobia, dan

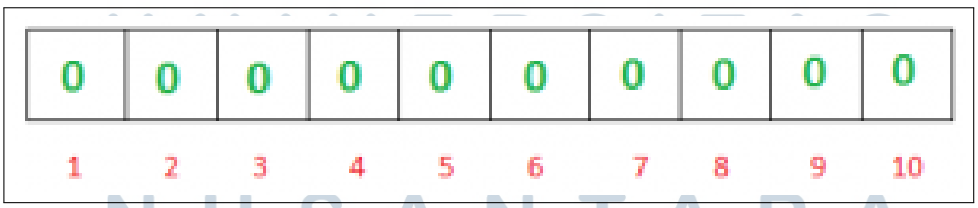
ksenofobia [19]. Namun untuk *bilingual hate speech* yang akan diteliti ini adalah dalam bentuk teks dan yang bersifat menghina atau kebencian terhadap seseorang.

2.2 BLOOM Filter

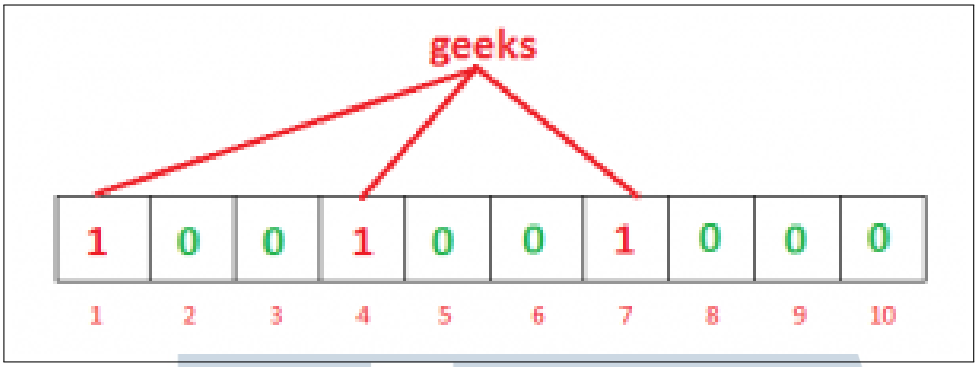
BLOOM (*BigScience Large Open-science Open-access bilingual language model*) adalah sebuah *language model* yang memilki 176 miliar parameter yang diltatih dengan 46 *natural langunges* dan 13 *programming languanges* [20]. BLOOM filter merupakan sebuah tempat penyimpanan data probabilitas yang efisien dalam ruang penyimpanan, BLOOM filter dapat digunakan untuk menguji apakah sebuah elemen termasuk kedalam kelompok tertentu.

Sebagai contoh BLOOM Filter dapat digunakan untuk memeriksa apakah *username* sudah terdaftar atau belum, dengan kelompok pengecekan adalah daftar *username* yang sudah terdaftar, sehingga dapat mengeluarkan hasil "sudah terdaftar" atau "belum terdaftar" [21]. BLOOM Filter memiliki kelemahan, seperti adanya nilai false-positive yang mutlak, namun kelemahan tersebut dapat ditutupi dengn adanya kecepatan dan efisiensi dalam membaca dan menyimpan kumpulan data. BLOOM filter memiliki kemampuan untuk menerima banyak elemen dan menganalisis data tersebut dengan cepat serta efisien, namun mengorbankan akurasi dari analisa nya.

BLOOM memiliki dua parameter yaitu parameter m berupa panjang *array* dan k berupa banyaknya fungsi *hash*. BLOOM beroperasi dengan membuat beberapa *array* kosong dengan ukuran m dengan value 0 pada setiap *array*-nya. Lalu setelah itu BLOOM akan menerima input berupa *string* yang akan dimasukan kedalam k fungsi *hash* yang berbeda. Lalu hasil dari fungsi *hash* tersebut akan menentukan letak *string* tersebut akan disimpan didalam *array* yang sudah dibuat, seperti yang bisa dilihat pada Gambar 2.1 dan Gambar 2.2 [21].



Gambar 2.1. Inisialisasi *array*



Gambar 2.2. BLOOM menyimpan lebih dari satu indeks

Lalu cara pengecekannya adalah dengan melakukan proses yang sama namun dengan urutan yang berkebalikan, kalkulasikan terlebih dahulu *hash* yang bersangkutan dengan menggunakan fungsi *hash* yang sudah dibuat, lalu periksa apakah setiap indeks yang didapat dari fungsi *hash* memiliki nilai 1 di dalam *array*. Jika tiap indeks yang didapatkan memiliki nilai 1 di dalam *array*, maka dapat dikatakan teks "mungkin ada".

Tetapi apabila terdapat nilai 0 pada salah satu indeks yang di dapat, maka teks "pasti tidak ada". Walaupun semua indeks memiliki nilai 1 BLOOM masi mendapatkan hasil "mungkin ada", hal tersebut dikarenakan suatu indeks bisa saja sudah terisi oleh teks lain yang menyebabkan indeks tersebut sudah bernilai 1 sebelumnya sehingga menghasilkan *false positive*. BLOOM bisa mengurangi hasil *false positive* dengan cara menggunakan fungsi *hash* yang lebih banyak dan ukuran *array* yang lebih besar, tetapi hal ini bisa menimbulkan latensi pada BLOOM filter [21].

Model BLOOM dapat dipanggil melalui *notebook* dengan cara meng-*import* BLOOMForSequenceClassification dan BLOOMTokenizerFast untuk proses *Tokenization*. BloomForSequenceClassification akan digunakan karena dari *pre-trained model* tersebut sudah terdapat lapisan-lapisan pemrosesan fitur yang akan membantu dalam proses klasifikasi menggunakan BLOOM Filter sebagai *transformer*. *Tokenization* atau tokenisasi adalah sebuah tahap atau proses pemecahan sebuah kalimat atau paragraf menjadi sebuah kata, simbol, atau elemen lainnya yang bermakna, yang disebut dengan "tokens" [22], tokenisasi dapat dilakukan dengan menggunakan BLOOMTokenizerFast yang bersifat *open source* [23]. Sehingga dataset yang sudah diubah menjadi token dapat diproses oleh model. Model BLOOM sendiri dapat digunakan untuk mendeteksi *hate speech* dengan cara mendeteksi apakah suatu elemen input merupakan bagian dari suatu *array*, dengan

adanya probabilitas dari *false positive*.

Model BLOOM tersedia dalam beberapa varian, seperti varian 350m, 560m, 1b3 dan 1b7, yang membedakan dalam jumlah parameter yang dimiliki setiap variasinya. Pada penelitian ini model BLOOM yang akan dipakai adalah 560m dengan total 560 juta parameter (ukuran array) dan 2048 hash token, karena berdasarkan hasil dari penelitian varian BLOOM dalam melakukan NLP [23], model BLOOM dengan varian 1b7 yang memiliki jumlah parameter lebih besar tidak berarti memiliki performa yang lebih baik dibanding dengan BLOOM varian 560m.

Melakukan *fine tuning* pada BLOOM dapat dilakukan dengan menggunakan *pre-trained model*, optimisasi *dataset*, mengubah jumlah parameter pada model, mengatur cara pelatihan model sehingga menjadi optimal seperti penggunaan *backpropagation*, dan evaluasi hasil dari pelatihan model. Optimisasi data dapat dilakukan dengan *data pre-process* yaitu menghapus teks atau kata yang tidak diperlukan, mengganti semua teks dalam data dengan huruf kecil untuk menghindari kesalahan tokenisasi, menghilangkan *special character* pada teks, membuang data dengan nilai yang hilang, dan mengubah format data menjadi yang diperlukan untuk model. [11].

Selain itu *optimizer* dapat digunakan untuk optimisasi pelatihan model, dengan fungsi mengatur *weights* atau untuk mengurangi *loss* [24], *optimizer* menerima 2 parameter yaitu *learning rate* dan epsilon. *Learning rate* adalah sebuah parameter untuk menghitung nilai koreksi bobot pada waktu proses training, sedangkan epsilon berupa angka yang sangat kecil untuk mencegah pembagian dengan nol dalam implementasi [25]. *optimizer* yang akan dipakai dalam penelitian ini adalah AdamW. AdamW merupakan metode optimasi di mana bobot dipisahkan terlebih dahulu dari perhitungan gradien dan dihitung kembali saat bobot diperbarui [12], AdamW ditujukan untuk optimasi stokastik yang hanya memerlukan *first-order gradient* dengan kebutuhan memori yang sedikit [26]. AdamW sebelumnya juga digunakan pada penelitian terdahulu [13] [12], dan merupakan salah satu optimizer yang paling sering digunakan untuk *text classification*.

2.3 Metrik Evaluasi

Metrik evaluasi digunakan untuk mengevaluasi hasil performa model yang telah dibuat, metrik evaluasi yang paling sering digunakan untuk mengevaluasi penelitian *text classification* adalah *confusion matrix*, diantaranya *accuracy*,

precision, *F1-Score*, dan *recall* [22].

Accuracy adalah metrik yang mengukur seberapa sering model melakukan prediksi dengan benar secara keseluruhan, Rumus dari akurasi dapat dilihat dari Persamaan 2.1

$$accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{2.1}$$

Precision adalah metrik evaluasi yang memberikan informasi seberapa besar model melakukan prediksi positif yang benar, metrik didapat dengan pembagian total prediksi yang benar dibagi dengan total semua positif. Formula dari presisi dapat dilihat dari Persamaan 2.2

$$precision = \frac{TP}{TP+FP} \tag{2.2}$$

Recall adalah sebuah metrik yang menghitung seberapa sering model memprediksi positif dari seluruh data positif yang ada. Formula dari recall dapat dilihat dari Persamaan 2.3

$$recall = \frac{TP}{TP+FN} \tag{2.3}$$

F1-Score adalah salah satu metrik evaluasi yang populer dalam evaluasi klasifikasi, memberikan informasi rata-rata dari *precision* dan *recall* dengan jumlah yang sama, f1-score memiliki nilai terbaik 1 dan nilai terburuk 0. evaluasi ini berguna untuk menemukan trade-off terbaik antara kedua kuantitas. Formula dari f1-score dapat dilihat dari Persamaan 2.4

$$F_1 = \frac{2TP}{2TP+FP+FN} \tag{2.4}$$