PENANDAAN BAHASA INDONESIA-INGGRIS DARI DATA *CODE-SWITCHING* DAN *CODE-MIXING* DENGAN METODE MNN DAN *CONTEXT CAPTURE*

Disusun Oleh:

N a m a : Nabil Muhammad Firdaus

NIM : 13523198

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENANDAAN BAHASA INDONESIA-INGGRIS DARI DATA *CODE-SWITCHING* DAN *CODE-MIXING* DENGAN METODE MNN DAN *CONTEXT CAPTURE*

TUGAS AKHIR

Disusun Oleh:

N a m a : Nabil Muhammad Firdaus
NIM : 13523198

Yogyakarta, 28 September 2020

Pembimbing,

(Ahmad Fathan Hidayatullah, S.T., M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENANDAAN BAHASA INDONESIA-INGGRIS DARI DATA *CODE-SWITCHING* DAN *CODE-MIXING* DENGAN METODE MNN DAN *CONTEXT CAPTURE*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 9 Oktober 2020

Tim Penguji

Ahmad Fathan Hidayatullah, ST., M.Cs. _____

Anggota 1

Beni Suranto, S.T., M.Soft.Eng. _____

Anggota 2

Irving Vitra Paputungan, S.T., M.Sc. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Nabil Muhammad Firdaus

NIM : 13523198

Tugas akhir dengan judul:

PENANDAAN BAHASA INDONESIA-INGGRIS DARI DATA *CODE-SWITCHING* DAN *CODE-MIXING* DENGAN METODE MNN DAN *CONTEXT CAPTURE*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 28 September 2020

(Nabil Muhammad Firdaus)

HALAMAN PERSEMBAHAN

Bismillah

Tugas akhir ini penulis persembahkan untuk kedua orang tua dan dosen Informatika UII tercinta. Dengan ini, saya menyatakan akan segera lulus dari UII. Sekian terimakasih

HALAMAN MOTO

"Maka bersabarlah engkau dengan sabar yang baik."

(QS. Al-Ma'arij : 5)

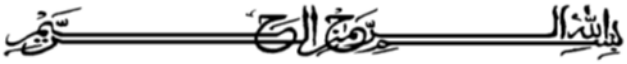
Lau kaana khairan, lasabaquna ilaihi

"Seandainya suatu perbuatan itu baik, pasti mereka (para Sahabat *Radhiyallahu'anhu*) telah melakukannya."

(Disadur dari buku karya Ustadz Abdul Hakim bin Amir Abdat)

"*Never make excuses. Your friends don't need them, and your foes won't believe them*" - John Wooden

KATA PENGANTAR



Assalaamu'alaikum Warahmatullaahi Wabarakaatuh

Alhamdulillah, puji dan syukur kepada Allah *'azza wa jalla*, yang dengan limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan tugas akhir ini dengan baik. Shalawat dan salam semoga selalu tercurah kepada kekasih-Nya, manusia terbaik sepanjang masa, Nabi Muhammad *Shallallahu'alaihi wasallam*, kepada istri-istri beliau, para sabahat, keturunan dan para pengikutnya hingga akhir zaman.

Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar sarjana dari Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Dalam menyelesaikan tugas akhir ini, penulis mendapatkan banyak bantuan, bimbingan, serta do'a dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada:

1. Allah *'azza wa jalla* atas segala nikmat, rahmat, dan karunia-Nya yang telah diberikan kepada penulis.
2. Nabi Muhammad *Shallallahu'alaihi wasallam*, yang telah menjadi suri tauladan terbaik.
3. Orang tua dan keluarga yang selalu memberikan dukungan dan do'a kepada penulis. Tiada henti-hentinya memberi arahan kepada penulis untuk menyelesaikan kuliahnya.
4. Orang yang paling spesial bagi penulis, Kartika Nur Kholidah, S.Kom. *Sorry for being so long*. *Baarakallahufiikum*
5. Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Pak Ahmad Fathan Hidayatullah, S.T., M.Cs., selaku Dosen Pembimbing yang telah memberikan bimbingan dan arahannya dengan baik. Mohon maaf jika saya selaku mahasiswa bimbingan banyak merepotkan.
7. Terimakasih kepada bapak ibu dosen yang tidak bisa saya tuliskan satu persatu, terimakasih atas ilmunya selama ini. Tanpa kalian, penulis tidak akan bisa menjadi seorang mahasiswa yang baik.

8. Terimakasih kepada semua teman-teman seperjuangan, kenangan kalian tak akan terlupakan. Teman-teman Invose, Asisten Lab, PSC, dan lain-lain. See you soon!
9. Semua pihak yang telah banyak membantu yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa tugas akhir ini masih begitu banyak kekurangan, sehingga penulis sangat terbuka jika terdapat kritik dan saran yang membangun demi kesempurnaan tugas akhir ini. Akhir kata, penulis berharap semoga tugas akhir ini dapat bermanfaat bagi semua pihak. *Jazaakumullahu khairan*

Wassalaamu'alaikum Warahmatullaahi Wabarakaatuh

Yogyakarta, 27 September 2020

(Nabil Muhammad Firdaus)

SARI

Media sosial telah memberikan dampak yang cukup signifikan dalam perkembangan sebuah bahasa dewasa ini. Percakapan menggunakan multibahasa merupakan hal yang sangat sering kita temui di berbagai media sosial, bahkan di berbagai macam platform yang berada pada internet. Pengguna seringkali beralih dari satu bahasa ke bahasa yang lain, mencampurkan berbagai macam bahasa ke dalam sebuah kalimat secara sadar maupun tidak sadar. Fenomena ini dikenal dengan istilah *code-mixing* dan *code-switching.* Dengan adanya fenomena tersebut, maka menjadi tugas yang sulit untuk melakukan identifikasi pada kalimat tersebut. Oleh karena itu, pada penelitian ini dilakukan proses identifikasi bahasa pada data *code-mixing* dan *code-switching*. Pada penelitian ini, penulis melakukan pelatihan model dengan menggunakan algoritma MNN (CNN dan LSTM) dan Bi-LSTM-CRF. MNN digunakan untuk melakukan pelatihan model pada level kata, yang memperoleh hasil akurasi sebesar 95.2% dan 81.59% nilai f1 *score* untuk Bi-LSTM-CRF yang digunakan untuk melakukan pelatihan model pada level kalimat.

Kata kunci: MNN, CNN, Bi-LSTM-CRF, *code-mixing*, *code-switching*

GLOSARIUM

Compile	proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh sistem operasi secara langsung.
Epoch	jumlah tahapan pelatihan untuk seluruh data.
Batch Size	jumlah sampel yang dilatih pada tiap *epoch*.
Overfitting	kondisi dimana model memiliki kinerja terlalu baik pada data latih, namun tidak untuk data baru.
Convolution Layer	Layer utama dalam membangun CNN.
Optimizer	Digunakan dalam pelatihan data untuk mengoptimalkan performa model
FC layer	Layer yang terhubung sempurna sebagai keluaran jaringan buatan.
Character Encoding	Proses mengubah sebuah text menjadi berbentuk *binary matrix*, sehingga masing-masing huruf dipetakan dengan index angka.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
NOTASI MATEMATIS xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan Penelitian 3
1.4 Batasan Masalah 3
1.5 Manfaat Penelitian 3
1.6 Metode Penelitian 3
BAB II LANDASAN TEORI 5
2.1 Penelitian Sebelumnya 5
2.2 Dasar Teori 6
2.2.1 *Term Frequency – Inverse Document Frequency* (TF-IDF) 6
2.2.2 *Support Vector Machine (SVM)* 8
2.2.3 *Recurrent Neural Networks (RNN)* 11
2.2.4 Jaringan *Long Short-Term Memory* (LSTM) 11
2.2.5 Metode Leksikon 12
2.2.6 *Convolutional Neural Network* (CNN) 12
2.2.7 *Conditional Random Field* (CRF) 16
2.2.8 *Synthetic Minority Oversampling Technique* (SMOTE) 17
BAB III METODOLOGI PENELITIAN 20
3.1 Analisis Masalah 20
3.2 Langkah Penelitian 21
3.3 Uraian Metodologi 22
3.3.1 Pengumpulan Data 22
3.3.2 *Preprocessing* 22
3.3.3 Pelabelan *Dataset* (Mesin) 23
3.3.4 Pelabelan *Dataset* (Manual) 24
3.3.5 Penandaan Bahasa (MNN) 25
3.3.6 Penandaan Bahasa (Bi-LSTM-CRF) 26
3.3.7 Evaluasi 27
BAB IV HASIL DAN PEMBAHASAN 29
4.1 Penyelesaian Masalah 29
4.2 Hasil Pengumpulan Data 30
4.3 *Preprocessing* 30
4.4 Pelabelan *Dataset* (Mesin) 31

4.5 Pelabelan *Dataset* (Manual) 34
4.6 Penandaan Bahasa (MNN) 34
4.7 Penandaan Bahasa (Bi-LSTM-CRF) 38
BAB V KESIMPULAN DAN SARAN 41
5.1 Kesimpulan 41
5.2 Saran 41
DAFTAR PUSTAKA 42
LAMPIRAN 44

DAFTAR TABEL

Tabel 2.1 Penelitian Sebelumnya 5
Tabel 3.1 Statistik penyaringan untuk α dan β 22
Tabel 3.2 Tabel daftar label prediksi 23
Tabel 3.3 Contoh *Confusion Matrix* 27

DAFTAR GAMBAR

Gambar 2.1 SVM Linier (a) 8
Gambar 2.2 SVM Linier (b) 9
Gambar 2.3 SVM Non Linier 10
Gambar 2.4 Arsitektur Jaringan LSTM 12
Gambar 2.5 Contoh Operasi Konvolusi 13
Gambar 2.6 Contoh pengambilan vektor *max pooling* & *average pooling* 14
Gambar 2.7 Perbedaan penggunaan *dropout layer* 15
Gambar 2.8 Ilustrasi *Fully Connected* (FC) *Layer* 16
Gambar 2.9 *Linear-chain* CRF 16
Gambar 2.10 *Imbalance Data Binary Classification* 18
Gambar 2.11 *Dataset* setelah proses SMOTE 19
Gambar 3.1 Langkah-Langkah penelitian 21
Gambar 3.2 Arsitektur MNN untuk pelatihan level kata 26
Gambar 3.3 Arsitektur Bi-LSTM-CRF untuk pelatihan level kalimat 27
Gambar 4.1 Grafik distribusi keseluruhan kata 31
Gambar 4.2 Grafik distribusi data latih SVM 32
Gambar 4.3 Grafik distribusi data latih SVM setelah *oversampling* 33
Gambar 4.4 *Classification Report* SVM 33
Gambar 4.5 *Confusion Matrix* SVM 34
Gambar 4.6 Riwayat pelatihan MNN 35
Gambar 4.7 Riwayat akurasi MNN 36
Gambar 4.8 Riwayat *loss* MNN 36
Gambar 4.9 *Classification Report* MNN 37
Gambar 4.10 *Confusion Matrix* MNN 37
Gambar 4.11 Hasil prediksi model MNN 38
Gambar 4.12 Riwayat Pelatihan Bi-LSTM-CRF 39
Gambar 4.13 Hasil prediksi model Bi-LSTM-CRF 40

NOTASI MATEMATIS

Simbol	Nama Simbol	Definisi	Contoh
x	variabel x	nilai yang perlu untuk diketahui	jika 4x=8, maka x=2
f (x)	fungsi dari x	memetakan nilai dari x ke f (x)	f (x) = 2x + 6
$\sum$	Sigma	jumlah dari keseluruhan nilai	$\sum x_i = x_1+x_2+\ldots+x_n$
$\sum\sum$	Sigma	jumlah dari keseluruhan nilai, dimana nilai diperoleh dari jumlah keseluruhan sub nilai	$\sum_{j=1}^{2}\sum_{i=1}^{8} x_{i,j} = \sum_{i=1}^{8} x_{i,1} + \sum_{i=1}^{8} x_{i,2}$
$\prod$	Simbol kapital dari phi	hasil perkalian (*dot product*) dari keseluruhan nilai pada suatu rentang nilai	$\prod x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_n$
log (x)	Logaritma	hasil log dari suatu nilai x	log 10 = 1
Φ (x)	Fi	dalam SVM, fungsi ini memetakan nilai x dari ruang input ke ruang vektor	
a_i	index variabel	nilai a pada index ke-i	$a_i = 5$
a_{ij}	index variabel	nilai a pada index ke-i dan ke-j	$a_{ij} = 10$
>	ketidaksamaan ketat	lebih dari	5 > 2
<	ketidaksamaan ketat	kurang dari	3 < 10
≥	ketidaksamaan	lebih dari atau sama dengan	5 ≥ 2

$\leq$	ketidaksamaan	kurang dari atau sama dengan	$3 \leq 10$
.	perkalian titik (*multiplication dot*)	perkalian	4 . 5 = 20
a^b	pangkat	eksponen	$5^2 = 25$
ψ	Psi	pada CRF, ini disebut sebagai fungsi fitur, yaitu beberapa fitur yang didapatkan pada masing-masing *sequence*	ψ (x, y)
$\vec{x}$	vektor	sebuah matrix yang hanya memiliki 1 kolom atau 1 baris saja	$\vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
A^T	transpose	transpose matrix	$(A^T)_{ij} = (A)_{ji}$
$P(A \mid B)$	fungsi probabilitas kondisional	probabilitas A dalam kondisi B	P (A | B) = 0.3
$\max\{f(x): x \in A\}$	nilai maksimal	nilai maksimal pada suatu rentang nilai, dimana nilai x merupakan anggota dari A	jika f(x) = 2x, A = 1,2,3,4,5,6 maka: $\max\{f(x): x \in A\}$ = max{2,4,6,8,10,12} = 12

BAB I
PENDAHULUAN

1.1 Latar Belakang

Media sosial memaikan peran yang sangat penting dalam kehidupan saat ini, media sosial merupakan alat yang memungkinkan orang untuk menjelajahi dan belajar tentang suatu informasi, membagikan ide, berkomunikasi dengan orang maupun organisasi. Media sosial telah mengubah cara orang menjalani hidup mereka saat ini, media sosial telah menjadikan komunikasi menjadi lebih mudah. Kita akan menemukan lebih dari jutaan pengguna di media sosial saat ini, pada situs-situs seperti Facebook, Twitter, Instagram, LinkedIn, hingga WhatsApp. Dengan media sosial, seseorang dapat berkomunikasi dengan siapapun, tanpa terbatas oleh ruang dan waktu. Akibatnya, orang yang berasal dari Indonesia dapat dengan mudah berkomunikasi dengan orang yang berasal dari Amerika sekalipun mereka tidak saling mengenal. Fenomera tersebut menjadi bukti bahwa telah terjadi sebuah pergeseran yang cukup signifikan dari yang awalnya sebuah komunikasi terbatas pada lingkup suatu negara menjadi lingkup antar-negara. Sebagai dampaknya, seseorang perlu untuk memahami bahasa lawan bicaranya agar dapat berkomunikasi dengan baik dan benar. Oleh sebab itu, bahasa memegang peranan penting dalam kehidupan manusia saat ini. Dengan bahasa seseorang dapat menyampaikan ide, pikiran, perasaan atau informasi kepada orang lain, baik secara lisan maupun tulisan (Firmansyah, 2019).

Media sosial telah memberikan dampak yang cukup signifikan dalam perkembangan sebuah bahasa dewasa ini. Percakapan menggunakan multibahasa merupakan hal yang sangat sering kita temui di berbagai media sosial, bahkan di berbagai macam platform yang berada pada internet. Pengguna seringkali beralih dari satu bahasa ke bahasa yang lain, mencampurkan berbagai macam bahasa ke dalam sebuah kalimat secara sadar maupun tidak sadar, contohnya seperti mencampurkan bahasa Inggris (bahasa kedua) ke bahasa Indonesia (bahasa utama). Fenomena ini dikenal dengan istilah *code-mixing* dan *code-switching*. *Code-switching* merupakan fenomena lingustik yang umumnya terjadi pada pengguna dua atau lebih bahasa (Mahootian, 2006) melalui (Yuliana, Luziana, & Sarwendah, 2015). *Code-switching* mengacu pada penggunaan dua bahasa dalam kalimat atau wacana yang sama. Itu berarti pergantian kata, frasa, atau kalimat dari satu bahasa ke bahasa yang lain terjadi dalam satu ucapan yang sama (Marasigan, 1983) melalui (Asror, 2009). Misalnya, seseorang berkata "*I want to eat*" dan

"Saya ingin makan" pada satu ucapan. Sedangkan *code-mixing* merupakan pencampuran dua bahasa yang dilakukan secara sengaja, tanpa mengubah topik pembicaraan. Itu berarti terjadi perpaduan antara dua bahasa yang berbeda pada sebuah aturan bahasa. Contohnya ketika seseorang berkata "Kita akan mem-*prepare everything* bersama-sama" (Gumperz, 1982) melalui (Asror, 2009). *Code-Mixing* pada contoh tersebut juga dapat disebut sebagai *intra-sentential Code-Switching*. Sehingga, ketika penulis menyebutkan *Code-Switching*, yang dimaksud adalah *intra-sentential Code-Switching.*

Fenomena *code-switching* dan *code-mixing* menjadi menarik ketika kita membahas mengenai analisis sentimen. Penelitian mengenai analisis sentimen umumnya menggunakan *dataset* yang terbatas pada satu bahasa saja. Seperti contohnya pada penelitian yang dilakukan oleh (Nayoan, 2019), di sana hanya menggunakan *dataset* berbahasa Indonesia. Sedangkan telah kita ketahui bahwa pengguna internet dewasa ini sering menggunakan multibahasa, sehingga bisa jadi hasil sentimen yang diperoleh kurang akurat. Penelitian yang dilakukan dalam rangka identifikasi atau penandaan bahasa pada data *code-switching* dan *code-mixing* telah banyak dilakukan sebelumnya, seperti yang dilakukan oleh (Mandal & Singh, 2019). Penelitian tersebut melakukan identifikasi bahasa Bengali-Inggris dan Hindi-Inggris dengan menggunakan metode MNN (*Multichannel Neural Network*) dan *Context Capture*, yang menghasilkan akurasi mencapai 93.32%. Metode MNN pada penelitian tersebut memadukan antara metode CNN (*Convolutional Neural Network*) dan LSTM (*Long Short-term Memory*), sedangkan *Context Capture* menggunakan Bi-LSTM-CRF (*Bidirectional, Long Short-term Memory, Conditional Random Field*).

Oleh karena itu, pada penelitian ini penulis akan menggunakan metode yang sama untuk melakukan penandaan bahasa Indonesia-Inggris pada data *code-switching* dan *code-mixing*. Penulis akan menggunakan data yang diperoleh dari media sosial Twitter. Harapannya, dengan adanya penandaan bahasa yang sesuai pada *dataset* nantinya dapat dilakukan proses lebih lanjut, seperti penerjemahan bahasa kedua (Inggris) ke bahasa utama (Indonesia). Sehingga, ketika *dataset* digunakan untuk analisis sentimen akan mendapatkan hasil yang lebih akurat.

1.2 Rumusan Masalah

Berdasarkan uraian dari latar belakang di atas, maka rumusan masalah yang menjadi fokus dan akan diselesaikan dalam tugas akhir ini yaitu bagaimana cara melakukan penandaan bahasa Indonesia-Inggris dari data *code-switching* dan *code-mixing*.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat sebuah sistem yang dapat melakukan penandaan bahasa Indonesia-Inggris dari data *code-switching* dan *code-mixing*, serta untuk mengetahui apakah metode MNN dan *Context Capture* dapat dengan baik melakukan penandaan bahasa tersebut.

1.4 Batasan Masalah

Adapun batasan masalah agar penelitian ini tetap fokus terhadap penyelesaian masalah, yaitu sebagai berikut:

1. Data yang digunakan adalah data percakapan yang diperoleh dari pengguna media sosial Twitter.
2. Sistem hanya akan melakukan fokus penandaan bahasa pada bahasa indonesia dan bahasa inggris.

1.5 Manfaat Penelitian

Adapun manfaat yang ingin didapatkan pada penelitian ini adalah sebagai berikut:

1. Mengetahui apakah implementasi metode MNN & *Context Capture* dapat diterapkan dengan baik pada data *code-switching* dan *code-mixing* (Indonesia-Inggris).
2. Membantu mendapatkan penandaan bahasa pada sebuah kalimat berbahasa Indonesia-Inggris.
3. Dengan mengimplementasikan *Deep Learning* pada domain tersebut, diharapkan dapat menjadi salah satu kontribusi penting untuk topik terkait.
4. Mengajak pembaca untuk lebih peduli terhadap bahasa Indonesia.

1.6 Metode Penelitian

Sistematika penulisan pada laporan skripsi tersusun dari beberapa bab berikut:

BAB I Pendahuluan, memberikan gambaran umum dan latar belakang mengenai penelitian "Penandaan Bahasa Indonesia-Inggris dari Data *Code-Switching* dan *Code-Mixing* dengan metode MNN dan *Context Capture*", rumusan masalah, tujuan, batasan masalah, dan manfaat penelitian.

BAB II Landasan Teori, memberikan penjelasan mengenai penelitian yang sudah dilakukan sebelumya, teori-teori penelitian, maupun hal-hal yang berkaitan sebagai

pendukung, pembanding, ataupun sebagai acuan dalam penelitian terkait *code-switching* dan *code-mixing*.

BAB III Metodologi Penelitian, menjelaskan mengenai seluruh tahapan yang digunakan dalam melakukan penelitian terkait *code-switching* dan *code-mixing*.

BAB IV Hasil & Pembahasan, memberikan penjabaran hasil penelitian yang dilakukan. Di dalamnya memuat penjelasan lebih detail mengenai hasil penelitian pada bagian pembahasan, serta evaluasi terhadap penelitian yang dilakukan.

BAB V Kesimpulan & Saran, memuat kesimpulan yang didapatkan dari penelitian, serta saran yang dapat diimplementasikan pada penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 Penelitian Sebelumnya

Bagian ini menjelaskan mengenai penelitian-penelitian yang telah dilakukan sebelumnya serta menjadi acuan terhadap penelitian ini. Daftar penelitian yang mendasari penelitian ini ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penelitian Sebelumnya

Judul Penelitian	Metode	Rangkuman
Character Embedding for Language Identification in Hindi-English Code-mixed Social Media Text	SVM	Penelitian ini melakukan identifikasi bahasa hindi dan inggris pada *code-mixed dataset*, dengan menggunakan metode SVM. Penulis memanfaatkan *word embedding* dan *character embedding* sebagai *feature vector*. Kemudian masing-masing fitur tersebut dibuat 3-gram dan 5-gramnya, lalu dilakukan pelatihan menggunakan metode SVM. Penulis mendapatkan hasil yang lebih akurat ketika menggunakan *character embedding*, dibanding *word embedding*.
Language Identification of Bengali-English Code-Mixed data using Character & Phonetic based LSTM Models	LSTM	Penelitian ini melakukan identifikasi bahasa bengali dan inggris dengan menggunakan *character encoding* dan *phonetic encoding*, pada *code-mixed dataset*. Kemudian penulis melakukan pelatihan model dengan menggunakan algoritma LSTM. Penulis lalu menambahkan teknik penggabungan (menggabungkan *character encoding* dan *phonetic encoding*) dan teknik *thresholding*. Model menghasilkan akurasi sebesar 91.78% untuk teknik penggabungan, dan 92.35% untuk teknik *tresholding*. Penulis menerangkan jika kekurangan dari model yang dibuat yaitu pelatihan hanya berdasarkan kata, bukan kalimat. Sehingga mengabaikan konteks dari kata tesebut.
Language Identification in Code-Mixed Data using Multichannel Neural Networks	MNN dan Bi-LSTM-CRF	Penelitian ini melakukan identifikasi bahasa hindi-inggris (Hi) dan bengali-inggris (Bn) dengan menggunakan metode MNN dan Bi-LSTM-CRF. Metode MNN disini merupakan penggabungan antara metode CNN dan LSTM. MNN digunakan untuk melakukan identifikasi bahasa pada level

and Context Capture		kata. Sedangkan penulis menambahkan Bi-LSTM-CRF untuk melakukan identifikasi bahasa pada level kalimat, sehingga memungkinkan untuk menangkap konteks sebuah kata. Penulis melatih model pada level kata, kemudian hasil keluaran dari model (berupa nilai *sigmoid*), digabungkan dengan *character encoding*, lalu digunakan sebagai input pada model Bi-LSTM-CRF. Model MNN menghasilkan akurasi sebesar 92.87% untuk Hi dan 92.65% untuk Bn. Sedangkan model Bi-LSTM-CRF menghasilkan akurasi sebesar 93.28% untuk Hi dan 93.32% untuk Bn.

Penelitian yang saya lakukan mengacu pada penelitian yang dilakukan oleh (Mandal & Singh, 2019). Penelitian yang saya lakukan menggunakan metode, arsitektur dan langkah yang mirip seperti yang dilakukan pada penelitian tersebut. Perbedaannya pada penelitian yang saya lakukan, saya membangun model untuk melakukan prediksi bahasa indonesia-inggris pada data *code-mixing* dan *code-switching*. Selain itu, pada penelitian ini saya menambahkan *output* selain bahasa indonesia dan bahasa inggris (lihat pada bab 3). Sehingga, *output* pada metode MNN saya berupa nilai *softmax*, bukan *sigmoid*. Oleh sebab itu, nilai yang saya jadikan input pada metode Bi-LSTM-CRF merupakan nilai *softmax* yang saya gabungkan dengan *character encoding*.

2.2 Dasar Teori

2.2.1 *Term Frequency – Inverse Document Frequency* (TF-IDF)

TF (*Term Frequency*) adalah frekuensi dari kemunculan sebuah term dalam dokumen yang bersangkutan. Semakin besar jumlah kemunculan suatu term (TF tinggi) dalam dokumen, semakin besar pula bobotnya atau akan memberikan nilai kesesuaian yang semakin besar.

Pada *Term Frequency* (TF), terdapat beberapa jenis formula yang dapat digunakan :

a. TF biner (*binary* TF), hanya memperhatikan apakah suatu kata atau term ada atau tidak dalam dokumen, jika ada diberi nilai satu (1), jika tidak diberi nilai nol (0).
b. TF murni (raw TF), nilai TF diberikan berdasarkan jumlah kemunculan suatu term di dokumen. Contohnya, jika muncul lima (5) kali maka kata tersebut akan bernilai lima (5).
c. TF logaritmik, hal ini untuk menghindari dominansi dokumen yang mengandung sedikit term dalam query, namun mempunyai frekuensi yang tinggi.

$$\mathrm{T}F = \begin{cases} 1 + log_{10}(f_{t,d}), f_{t,d} > 0 \\ 0, \quad f_{t,d} = 0 \end{cases} \quad (2.1)$$

Di mana nilai $f_{t,d}$ adalah frekuensi term (t) pada document (d).

Sehingga, ketika suatu kata atau term terdapat dalam suatu dokumen sebanyak 10 kali maka diperoleh bobot = 1 + log (10) = 2. Tetapi jika term tidak terdapat dalam dokumen tersebut, bobotnya adalah nol (0).

d. TF normalisasi, menggunakan perbandingan antara frekuensi sebuah term dengan nilai maksimum dari keseluruhan atau kumpulan frekuensi term yang ada pada suatu dokumen.

$$\mathrm{T}F = 0.5 + 0.5 \times \left[\frac{f_{t,d}}{max\{f_{t\prime,d:t\prime,d\in d}\}}\right] \quad (2.2)$$

Sedangkan IDF (*Inverse Document Frequency*) adalah sebuah perhitungan dari bagaimana term didistribusikan secara luas pada koleksi dokumen yang bersangkutan. IDF menunjukkan hubungan ketersediaan sebuah term dalam seluruh dokumen. Semakin sedikit jumlah dokumen yang mengandung term yang dimaksud, maka nilai IDF semakin besar.

Rumus IDF adalah sebagai berikut:

$$IDF_j = log(D/df_j) \quad (2.3)$$

Lalu, untuk mendapatkan nilai TF-IDF, pertama hitung nilai TF-nya. Nilai TF yaitu jumlah dari query dalam suatu dokumen tersebut. Kemudian, hitung nilai IDF-nya. Untuk mengetahui nilai TF-IDF nya, kalikan nilai TF dengan IDF.

$$TF - IDF_{ij} = TF_{ij} \times IDF_j + 1 \quad (2.4)$$

Berapapun besarnya nilai TF_{ij}, apabila D = df_j, maka akan didapatkan hasil 0 (nol), dikarenakan hasil dari log 1 = 0, untuk perhitungan IDF. Untuk itu dapat ditambahkan nilai 1 pada sisi IDF.

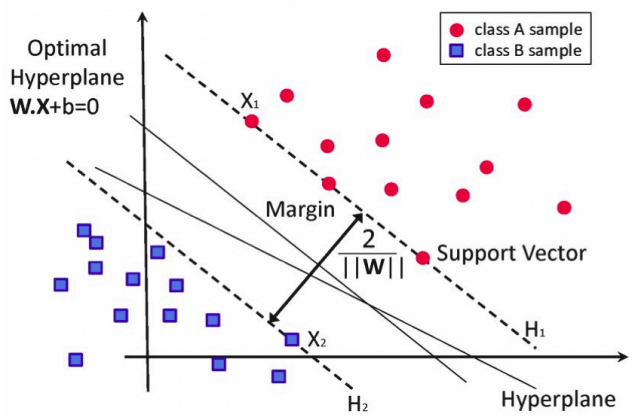
2.2.2 *Support Vector Machine (SVM)*

Support Vector Machines (SVM) adalah kelas populer dari algoritma *supervised* dari *machine learning*. SVM berupaya menemukan *hyperplane* pemisah antara dua kelas data berlabel. SVM biasanya digunakan untuk klasifikasi (*Support Vector Classification*) dan regresi (*Support Vector Regression*). Namun, sebagian besar digunakan dalam masalah klasifikasi. Ide dasar metode SVM adalah mengubah fitur input menjadi ruang dimensi yang lebih tinggi untuk memisahkan data secara linier menjadi dua kelas dengan *hyperplane*, yaitu dengan cara memaksimalkan margin dan meminimalkan kesalahan.

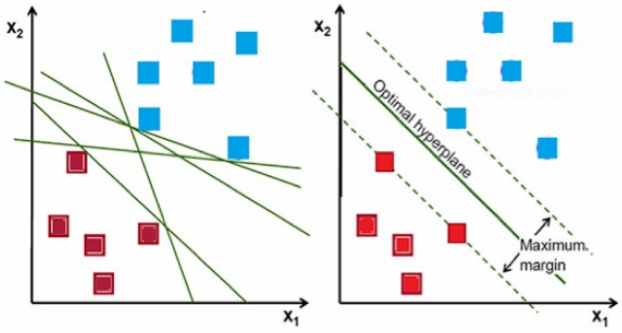
Support Vector Machine (SVM) adalah sebuah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi linier dalam sebuah ruang fitur (*feature space*) berdimensi tinggi yang dilatih dengan algoritma pembelajaran berdasarkan teori optimasi dengan mengimplementasikan *learning bias* yang berasal dari teori statistik. SVM dapat dibangun dalam dua jenis data, diantaranya adalah sebagai berikut:

a. Linear Separate

Linearly separable data merupakan data yang dapat di pisahkan secara linier. Klasifikasi SVM melakukan pencarian garis *hyperplane* yang terbaik untuk memisahkan *dataset* sesuai dengan kelasnya. *Hyperplane* terbaik selain dapat memisahkan data sesuai dengan kelasnya tetapi juga memiliki margin yang paling besar. Adapun bentuk klasifikasi menggunakan SVM dengan data linier dapat dilihat pada gambar di bawah ini.



Gambar 2.1 SVM Linier (a)



Gambar 2.2 SVM Linier (b)

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada *input space*. *Hyperplane* pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur margin dari *hyperplane* tersebut dan mencari titik maksimalnya. Margin adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector*. *Hyperplane* yang terbaik yaitu yang terletak tepat pada tengah-tengah kedua kelas, sedangkan titik merah dan kuning yang berada di dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM.

Masalah dari klasifikasi ini dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Jika data pelatihan terpisah secara linier, kita dapat memilih dua *hyperplane* paralel yang memisahkan dua kelas data, sehingga jarak di antara mereka sama besarnya. Wilayah yang dibatasi oleh dua *plane* ini disebut "margin", dan *hyperplane* margin maksimum adalah *hyperplane* yang terletak di tengah-tengahnya dengan *dataset* yang dinormalisasi atau terstandarisasi. *Hyperplane* ini dapat dijelaskan oleh persamaan:

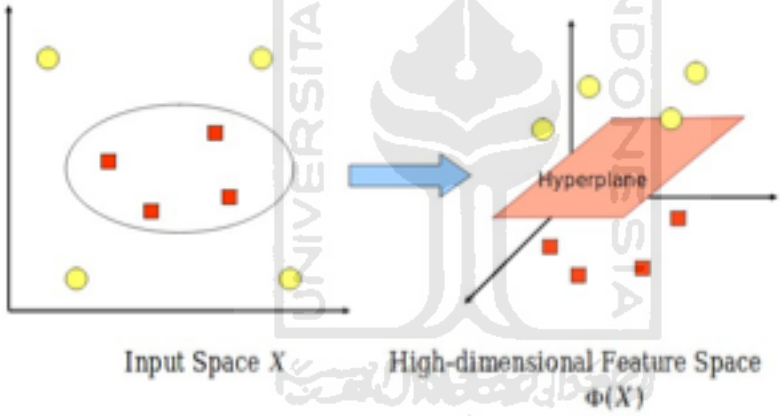
$$W^T x + b \geq 1 \; for \; d_i = +1 \qquad (2.5)$$

$$W^T x + b \leq 1 \; for \; d_i = -1 \qquad (2.6)$$

Setiap vektor input yang berada di atas *hyperplane* ini diklasifikasikan sebagai *instance* positif atau bernilai positif. Input dan vektor yang berada di bawah *hyperplane* ini diklasifikasikan sebagai negatif atau bernilai negatif.

b. Non Liniear Separate

Umumnya SVM digunakan untuk melakukan klasifikasi linier. Namun data yang berupa *non liniear separate* tidak dapat langsung dikerjakan dengan menggunakan teknik klasifikasi linier. Untuk menyelesaikan itu, SVM melakukan klasifikasi non-linier menggunakan trik kernel. Dalam non linier SVM, pertama-tama data $\overrightarrow{x_i}$ dipetakan oleh fungsi $\Phi(\vec{x})$ ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua kelas tersebut dapat dikonstruksikan. Hal ini sejalan dengan teori *Cover* yang menyatakan "Jika suatu transformasi bersifat non linier dan dimensi dari *feature space* cukup tinggi, maka data pada *input space* dapat dipetakan ke *feature space* yang baru, di mana *pattern-pattern* tersebut pada probabilitas tinggi dapat dipisahkan secara linier" (Nugroho, Witarto, & Handoko, 2011). Adapun bentuk klasifikasi menggunakan SVM non linier dapat dilihat pada gambar di bawah ini.



Gambar 2.3 SVM Non Linier

Ilustrasi dari konsep ini dapat dilihat pada gambar Gambar 2.3. Pada gambar sebelah kiri, diperlihatkan data pada kelas kuning dan data pada kelas merah yang berada pada *input space* berdimensi dua tidak dapat dipisahkan secara linier. Selanjutnya gambar sebelah kanan, menunjukkan bahwa fungsi Φ memetakan tiap data pada *input space* tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3), di mana kedua kelas dapat dipisahkan secara linier oleh sebuah *hyperplane*. Notasi matematika dari *mapping* ini adalah sebagai berikut (Nugroho et al., 2011).

$$\Phi: R^d \rightarrow R^q d < q \qquad (2.7)$$

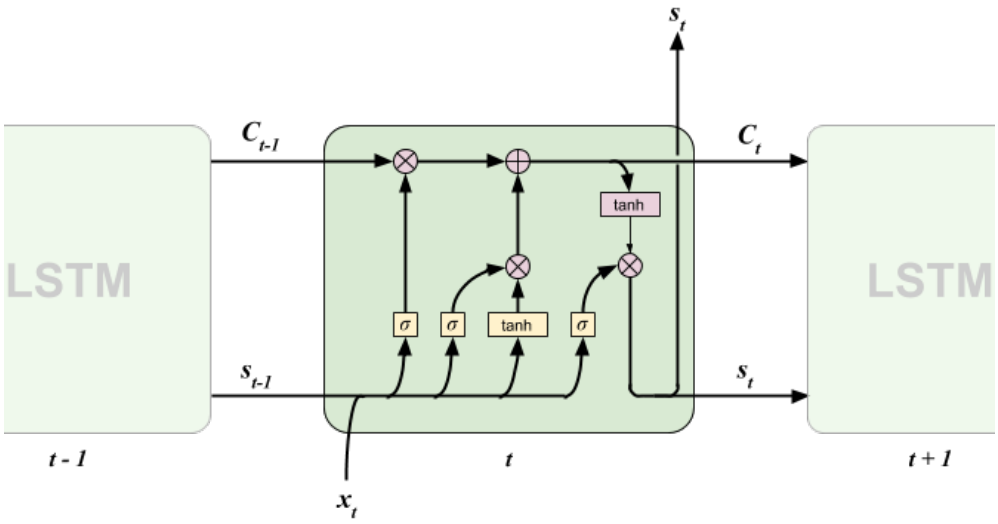
Menurut (Nugroho et al., 2011), pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada *input space* akan berjarak dekat juga pada *feature space*, sebaliknya dua data yang berjarak jauh pada *input space* akan juga berjarak jauh pada *feature space*. Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi, yaitu $\Phi(\overrightarrow{x_i}) . \Phi(\overrightarrow{x_j})$.

2.2.3 *Recurrent Neural Networks (RNN)*

RNN (*Recurrent Neural Networks*) merupakan salah satu metode dalam *Deep Learning*, di mana pemrosesan input dilakukan secara berulang-ulang yang biasanya berupa data sekuensial. Data sekuensial mempunyai karakteristik di mana sampel diproses dengan suatu urutan (misalnya waktu), dan suatu sampel dalam urutan mempunyai hubungan erat antara satu dengan yang lainnya.

2.2.4 Jaringan *Long Short-Term Memory* (LSTM)

LSTM (*Long Short-Term Memory*) merupakan salah satu jenis dari RNN (*Recurrent Neural Network*) yang dapat memproses informasi mana saja yang akan dibutuhkan untuk disimpan atau diabaikan. LSTM merupakan pengembangan dari RNN, dengan cara menambahkan *memory cell* yang dapat menyimpan informasi untuk jangka waktu yang lama (Manaswi, 2018). LSTM merupakan metode yang diciptakan untuk mengatasi keterbatasan model RNN dalam memproses data yang relatif panjang (*long term dependency*). Ketika RNN memproses input *sequence* yang panjang, maka semakin banyak *hidden layer* yang tebentuk. *Hidden layer* yang terbentuk dari input *sequence* yang panjang akan menjadi masalah pada RNN, sehingga dapat menyulitkan proses pembelajaran. Masalah seperti ini disebut *vanishing gradient*. *Vanishing gradient* adalah keadaan di mana *training* yang dilakukan oleh model RNN semakin sulit karena nilai *gradient* yang terlalu kecil, sehingga seperti tidak ada pembelajaran di model RNN yang dibangun.



Gambar 2.4 Arsitektur Jaringan LSTM

Di atas merupakan ilustrasi dari arsitektur jaringan LSTM. *Cell state* merupakan tempat untuk menyimpan informasi yang diberikan dari satu langkah waktu ke langkah waktu berikutnya. *Gate units* berperan dalam memproses informasi yang dibutuhkan atau dibuang. *Gate units* terdiri dari *input gate, forget gate,* dan *output gate*. *Input gate* merupakan *gate* yang berfungsi untuk memutuskan nilai input yang akan diteruskan pada *cell state* untuk diperbarui. *Forget gate* marupakan *gate* yang memutuskan informasi mana yang perlu dibuang dari *cell state*. *Output gate* merupakan *gate* yang memutuskan *output* yang akan dihasilkan.

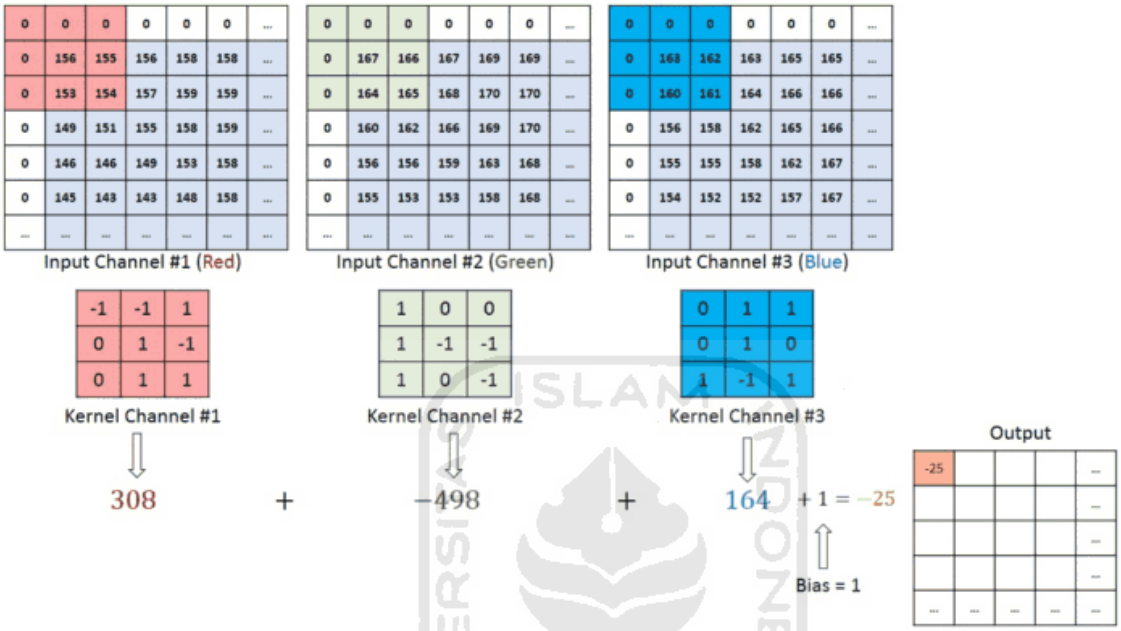
2.2.5 Metode Leksikon

Metode leksikon merupakan metode berbasis kamus (Aribowo, 2018). Metode ini menggunakan kamus sebagai acuan dalam memberikan pelabelan terhadap dokumen. Metode ini mampu mengidentifikasi konteks ataupun sentimen pada dokumen.

2.2.6 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) memiliki kemampuan baik dalam menyelesaikan masalah visi komputer karena dapat beroperasi secara konvolusional, yaitu melakukan ekstraksi fitur dari *patch* masukan lokal yang memungkinkan modularitas representasi dan efisiensi data. CNN sangat signifikan untuk pemrosesan berurutan. Lapisan konvolusi 1D juga tidak mengalami perubahan translasi karena masukan transformasi dilakukan pada setiap *patch*, sebuah pola yang dipelajari pada posisi tertentu dalam kalimat kemudian dapat dikenali pada posisi yang berbeda. Mirip dengan CNN 2D, *patch* 1D dapat diekstraksi dari masukan

dan memberikan keluaran nilai maksimum atau nilai rata-rata, proses ini disebut dengan *Max Pooling* dan *Average Pooling*. Penggunaannya pun sama dengan CNN 2D, yaitu untuk mengurangi panjang dari masukan 1D (secara teknik disebut dengan *subsampling*) (Chollet, 2017). Gambar 2.5 merupakan contoh operasi konvolusi.



Gambar 2.5 Contoh Operasi Konvolusi

CNN merupakan sebuah konstruk matematika yang biasanya disusun oleh 3 tipe layer, yaitu *convolution*, *pooling*, dan FC *layer*. Dua layer pertama, *convolution* dan *pooling* melakukan ekstraksi fitur, sedangkan layer ketiga, menempatkan fitur yang terekstraksi menjadi *output* seperti klasifikasi. Konvolusi adalah sebuah tipe operasi linier khusus yang digunakan untuk ekstraksi fitur, di mana sebuah array angka, yang disebut *kernel* diimplementasikan pada input yang merupakan array angka yang disebut tensor. Kalkulasi yang dilakukan pada masing-masing elemen matriks *kernel* dan input tensor, lalu dijumlahkan untuk mendapatkan keluaran yang disebut dengan *feature map*. Tahap ini dilakukan berulang hingga didapatkan *feature map* yang merepresentasikan karakteristik dari input tensor. Oleh karena itu, kernel dengan ukuran lain bisa memiliki hasil ekstraksi yang berbeda pula. Perhitungan propagasi konvolusi ditunjukkan pada persamaan 2.8.

$$a_{i\,j}^{(k)} = \sum_{s=0}^{m-1}\sum_{t=0}^{n-1} W_{st}^{(k)} x_{(i+s)(j+t)} + b^k \qquad (2.8)$$

Keterangan :

x : masukan

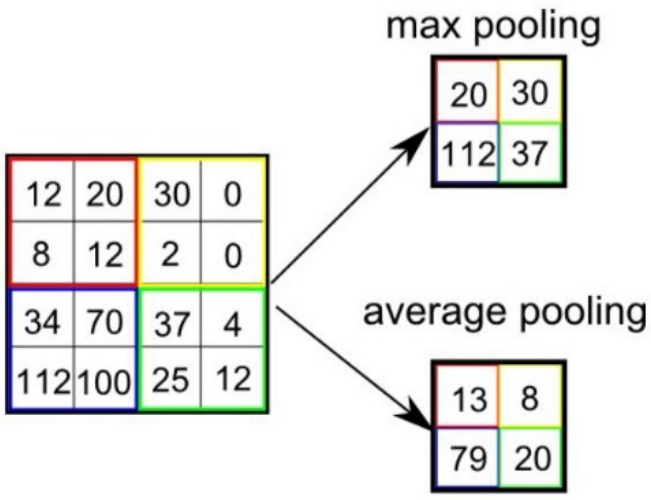
a^k : setelah konvolusi

k : indeks kernel (*weight filter*)

W : kernel (*weight filter*)

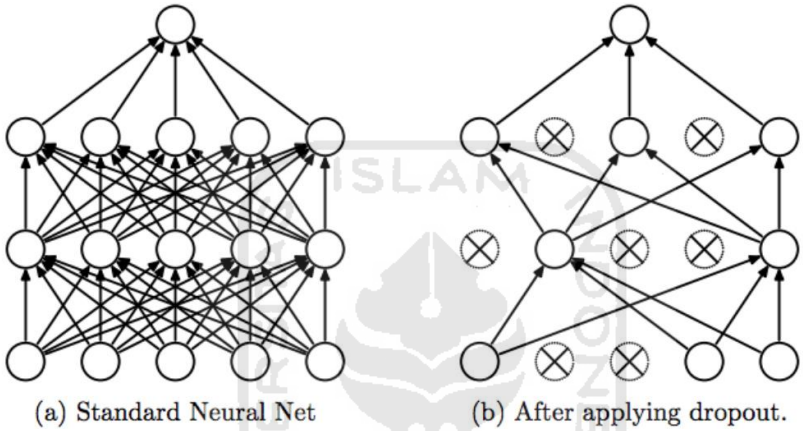
b : bias

Pooling disebut juga dengan penurunan sampel (*downsampling*) yang mengurangi dimensionalitas bidang namun tetap menjaga informasi yang penting. Operasi *pooling* yang sering digunakan yaitu *max pooling*, dimana operasi ini mampu melakukan ekstraksi dari *filter* masukan *feature maps*, kemudian memilih nilai tertinggi pada setiap *filter* dan membuang nilai lainnya. *Pooling* memiliki beberapa jenis, diantaranya yaitu *global max pooling* dan *average pooling*. 1D *Global Max Pooling* mengambil sebuah vektor dan menghitung nilai maksimum dari semua nilai untuk masing-masing saluran input. Sedangkan *Average Pooling* mengambil sebuah vektor dan menghitung nilai rata-rata dari semua nilai untuk masing-masing saluran input.



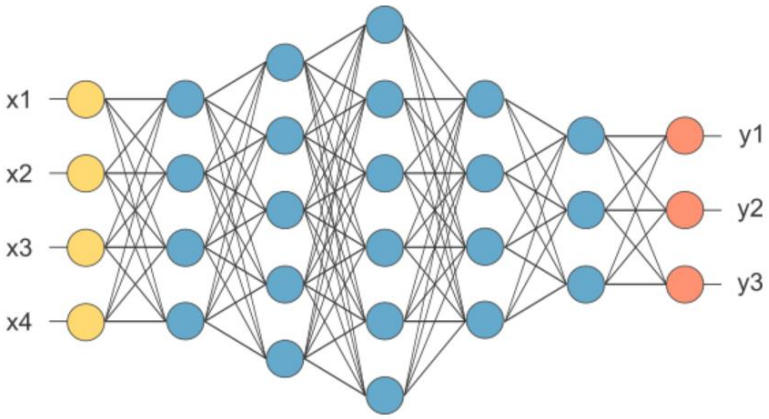
Gambar 2.6 Contoh pengambilan vektor *max pooling* & *average pooling*

FC *(Fully Connected) Layer* yaitu lapisan neuron yang terhubung sepenuhnya. Seluruh neuron pada FC *Layer* memiliki hubungan dengan seluruh aktivasi pada lapisan sebelumnya. Lapisan ini ditempatkan sebelum hasil klasifikasi CNN. *Dropout Layer* juga diaplikasikan pada CNN untuk mengatasi *overfitting*. Teknik *dropout* mengatasi *overfitting* dengan cara menghapus sementara kontribusi terhadap aktivasi neuron ketika *forward pass*, dan setiap pembaruan bobot tidak diterapkan pada neuron ketika *backward pass*. Gambar 2.7 menunjukkan bagaimana perbedaan ketika tidak menggunakan teknik *dropout* dan menggunakan teknik *dropout*.



Gambar 2.7 Perbedaan penggunaan *dropout layer*

Keluaran *feature maps* dari konvolusi akhir atau *layer pooling* biasanya berbentuk *flattened* (diubah menjadi satu dimensi angka/*vector array)* dan dihubungkan ke satu/lebih *layer* yang terhubung secara sempurna (FC) atau disebut juga *dense layer* yang ditunjukkan Gambar 2.8, di mana setiap masukan yang terhubung ke setiap keluaran memiliki bobot. Setelah melalui ekstraksi fitur pada *layer* konvolusi dan dilakukan penurunan sampel pada *pooling layer*, lalu keluaran dipetakan oleh *FC Layer* menjadi keluaran akhir jaringan, seperti probabilitas setiap kelas pada tugas klasifikasi. Hasil akhir *FC Layer* biasanya memiliki jumlah *node* yang sama dengan jumlah kelasnya.

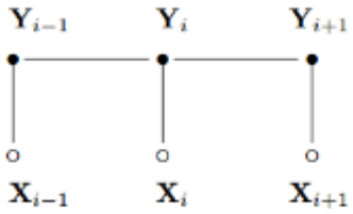


Gambar 2.8 Ilustrasi *Fully Connected* (FC) *Layer*

2.2.7 *Conditional Random Field* (CRF)

CRF merupakan suatu model probabilistik untuk segmentasi dan pelabelan suatu data sekuensial. CRF memiliki banyak kelebihan dibandingkan model probabilitas lain seperti *Hidden Markov Model* (HMM) dan *Maximum Entropy Markov Model* (MEMM). CRF mengatasi permasalahan ketergantungan asumsi yang tinggi pada HMM. Hal ini karena CRF dapat menentukan sendiri seberapa banyak fitur yang diinginkan untuk membangun sebuah model. CRF tidak seperti HMM yang bersifat lokal di mana setiap kata hanya bergantung pada label saat ini dan setiap label sebelumnya. Selain itu CRF dapat mempunyai bobot yang bebas sedangkan HMM harus memenuhi bobot tertentu. CRF juga mengatasi permasalahan label bias pada MEMM karena CRF memformulasikan distribusi kondisional label untuk data sekuensial secara kesuluruhan dibandingkan MEMM yang memformulasikan distribusi kondisional label untuk setiap elemen data (Fachri, 2014) melalui (Jaariyah & Rainarli, 2017).

CRF digunakan pada banyak aplikasi termasuk *Natural Language Processing* (NLP), *Computer Vision,* dan *bioinformatic*. Salah satu bentuk dari CRF adalah *linear-chain* CRF yang ditunjukkan oleh Gambar 2.9.



Gambar 2.9 *Linear-chain* CRF

Linear-chain CRF diberikan dua peubah acak di mana himpunan data $\bar{x} = (x_1, x_2, \ldots, x_n)$ observasi dan $\bar{y} = (y_1, y_2, \ldots, y_n)$ adalah himpunan label yang mungkin. Probabilitas kondisional dari y (kelas entitas bernama atau dalam hal ini bahasa) terhadap x (kata) dapat dituliskan dalam persamaan (2.9).

$$P(y|x) = \frac{1}{Z(x)} \prod_C \psi_C(y_C, x) \qquad (2.9)$$

Di mana $\psi_c(y_c, x)$ adalah fungsi positif yang selanjutnya disebut fungsi potensial dan Z(x) menunjukkan fungsi normalisasi dari distribusi probabilitas kondisional label (y) untuk semua sekuen data (x) dan dinotasikan dengan persamaan (2.10).

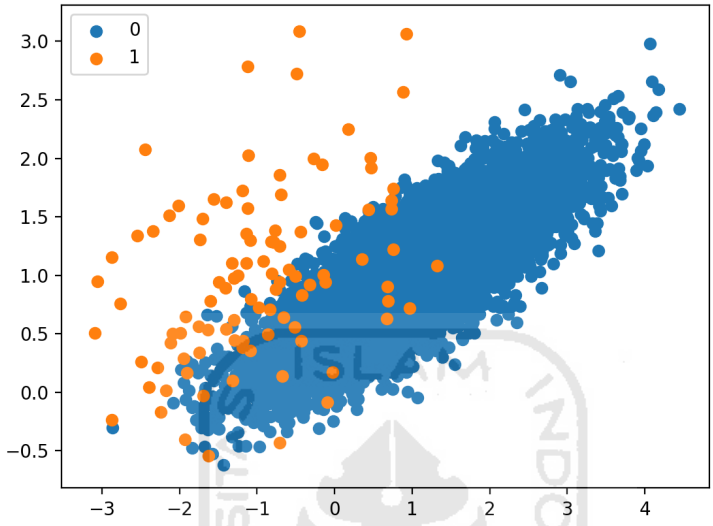
$$Z(x) = \sum_c \prod_C \psi_C(y_C, x) \qquad (2.10)$$

Fungsi potensial mengacu pada jumlah yang mengikat label data dengan fitur pada waktu yang sama. Fitur potensial biasanya disebut juga sebagai *data pattern* pada beberapa literatur CRF. Dalam kasus yang sederhana fungsi potensial merupakan eksponensial dari jumlah bobot semua fungsi fitur. Secara umum proses CRF merupakan proses penaksiran parameter kemudian diikuti oleh inferensi.

2.2.8 *Synthetic Minority Oversampling Technique* (SMOTE)

Akurasi dari sebuah model merupakan hal yang sangat penting dalam klasifikasi *machine learning*. Semakin akurat *dataset*, maka akan semakin akurat pula akurasi yang dihasilkan. Namun, tidak selalu semua berjalan sesuai dengan yang diharapkan. Terkadang, ketika kita membangan sebuah model *machine learning*, kita dihadapkan pada kondisi di mana *dataset* kita tidak seimbang (*imbalance*). Alhasil, model yang kita buat menjadi tidak akurat. Sehingga, jika kita ingin membuat model yang kita miliki menghasilkan akurasi yang lebih baik, maka kita perlu terlebih dahulu membuat *dataset*-nya seimbang.

Terdapat banyak cara untuk mengatasi data yang tidak seimbang, salah satunya yaitu dengan menerapkan metode *oversampling*. Cara termudah untuk menyeimbangkan *dataset* kita, yaitu dengan melakukan *oversampling* pada kelas minoritas. Misalnya ketika kita mempunyai sebuah masalah data *imbalance* pada *binary classification problem*, bisa dilihat contohnya pada Gambar 2.10.

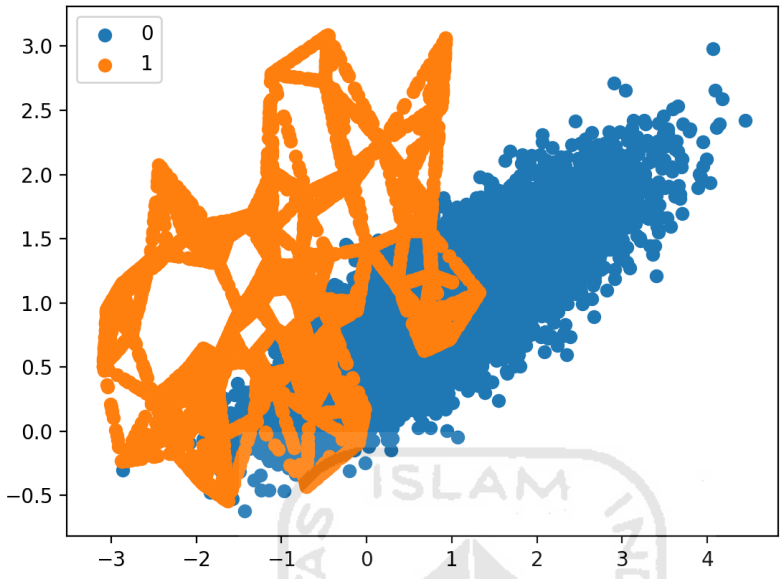


Gambar 2.10 *Imbalance Data Binary Classification*

Pada gambar di atas, terlihat jika data yang ada tidak seimbang. Kelas biru mempunyai lebih banyak data dibandingkan kelas oranye. Salah satu hal yang ditimbulkan dengan adanya data *imbalance* yaitu prediksi sebuah model biasanya akan lebih condong dengan kelas mayoritas. Hal tersebut tentu tidak akan terjadi ketika kita mempunyai data yang seimbang. Secara teori, metode *oversampling* dapat dilakukan dengan mudah yaitu dengan melakukan duplikasi data pada kelas minoritas, hingga *dataset* menjadi seimbang. Namun, dengan cara seperti itu model tidak akan menjadi lebih baik, karena model mendapatkan input yang sama berulang-ulang.

Pada 2002, (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) memberikan solusi untuk mengatasi permasalahan tersebut. Mereka memperkenalkan metode untuk melakukan oversampling dengan tepat yang diberi nama *Synthetic Minority Oversampling Technique* (SMOTE). Caranya yaitu dengan membuat data sintesis dari kelas minoritas. SMOTE bekerja dengan cara memilih fitur yang paling dekat, atau dengan cara memilih data secara acak pada data minoritas, kita sebut X. Kemudian, SMOTE akan mencari *k-nearest neighbour* dari data

tersebut, kita sebut Y. Setelah itu, SMOTE akan membuat data sintesis di antara X dan Y. Begitu seterusnya, hingga dataset menjadi seimbang seperti yang terlihat pada Gambar 2.11.



Gambar 2.11 *Dataset* setelah proses SMOTE

Namun, metode ini juga memiliki kelemahan yaitu metode tidak mempertimbangkan mayoritas kelas pada suatu titik. Sehingga, ketika di sana terdapat persamaan yang banyak antara kelas minoritas dan kelas mayoritas, maka SMOTE akan menghasilkan hasil yang ambigu. Hanya saja, metode ini sudah cukup bagus untuk melakukan *oversampling* data.

BAB III
METODOLOGI PENELITIAN

3.1 Analisis Masalah

Penandaan bahasa pada suatu kalimat *code-mixing* dan *code-switching* merupakan hal yang mungkin mudah dilakukan oleh manusia yang mempunyai kemampuan bahasa yang bagus. Dalam artian manusia dapat dengan mudah memahami sebuah kata pada suatu kalimat berasal dari bahasa mana serta dengan mudah mengetahui artinya. Hanya saja, ketika penandaan bahasa dilakukan oleh mesin, proses penandaan bahasa tersebut sulit untuk dilakukan, mengingat suatu kalimat ditulis dengan berbagai macam bentuk dan masing-masing bahasa dapat mempunyai kata yang sama dalam bahasa lain namun memiliki makna atau arti yang berbeda.

Dalam memahami sulitnya mesin untuk dapat melakukan penandaan bahasa, dapat dilihat pada contoh kalimat berikut

"Knp nama org Bali sering diawali dengan kata I? cth *simple*nya I Gusti Ngurah Rai. "
(Kenapa nama orang Bali sering diawali dengan kata I? Contoh mudahnya I Gusti Ngurah Rai.)

Dari kalimat di atas, mungkin bagi keumuman orang Indonesia dapat dengan mudah mengetahui artinya. Hanya saja, mesin tidak bisa membedakan apakah *org* yang dimaksud pada kalimat tersebut merupakan kependekan dari orang atau *organization*. Begitu juga dengan kata *I*, mesin tidak bisa membedakan apakah *I* yang dimaksud merupakan *I* (saya dalam bahasa Inggris) atau I (nama orang bali). Juga untuk kata *simple*nya, mesin tidak bisa mengetahui kata tersebut masuk ke dalam bahasa apa, mengingat di sana terdapat perpaduan antara bahasa Inggris dan sufiks Indonesia berupa -nya.

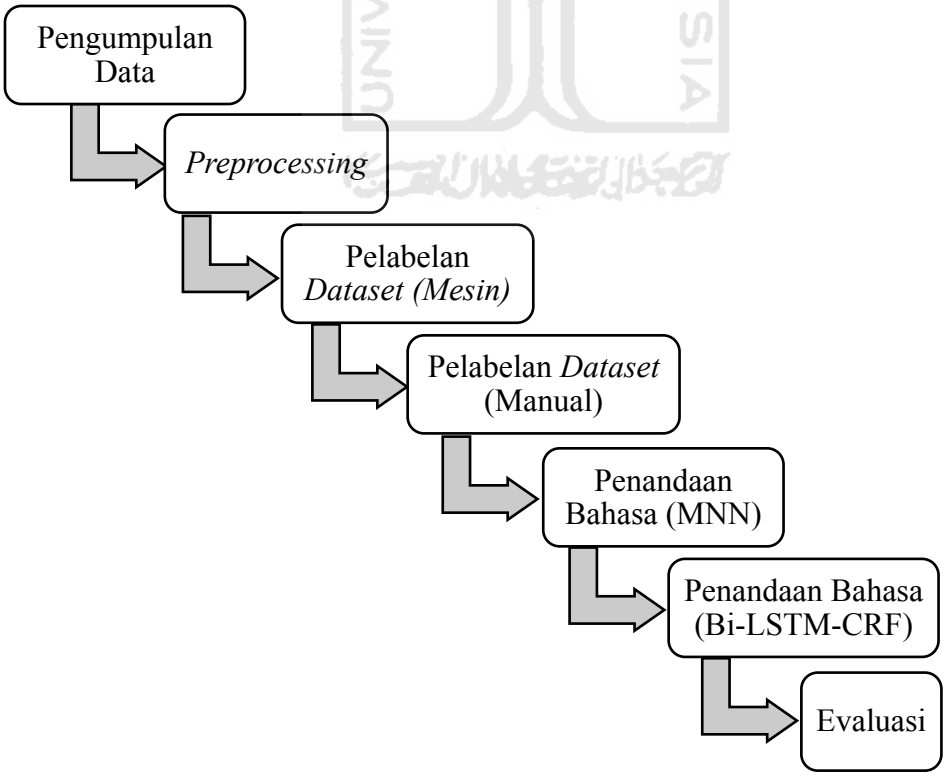
Jika kita melakukan analisa lebih jauh, penandaan bahasa merupakan hal yang sangat krusial bagi para peneliti yang ingin melakukan penelitian dengan menggunakan data dari media sosial. Sebagai contoh ketika kita melakukan sebuah analisis sentimen dengan menggunakan data dari media sosial Twitter, kita mungkin akan menjumpai kalimat seperti berikut

"Pak Jokowi itu gak *well* orangnya, mosok kejaannya *nyengsarakke* rakyate."
(Pak Jokowi itu orangnya tidak baik, masa kerjaannya menyengsarakan rakyatnya.)

Kalimat di atas, jika digunakan pada analisis sentimen pada umumnya (yang berfokus pada bahasa Indonesia), tidak diketahui hasil sentimen dari kalimat tersebut, karena kata yang bernada negatif ditulis dengan menggunakan bahasa Inggris dan bahasa Jawa. Sedangkan dapat kita ketahui, seharusnya kalimat tersebut masuk ke dalam sentimen negatif. Namun, jika kita dapat mengetahui bahasa dari masing-masing kata sesuai konteksnya, kita selanjutnya dapat melengkapinya dengan NER (*Named Entity Recognition*), yang kemudian dapat dikembangkan lebih lanjut menjadi *machine translation* (seperti *Google Translate*) untuk melakukan translasi kata pada bahasa yang diinginkan, sehingga proses analisis sentimen bisa bekerja dengan baik.

3.2 Langkah Penelitian

Berikut ini merupakan langkah-langkah yang digunakan dalam mengerjakan penandaan bahasa Indonesia-Inggris pada data *Code-Switching* dan *Code-Mixing,* seperti yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Langkah-Langkah penelitian

3.3 Uraian Metodologi

3.3.1 Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil data twit dari media sosial Twitter kemudian menyimpan data tersebut pada berkas berekstensi .csv. Data dikumpulkan dengan menggunakan *library* python bernama twint. Untuk detail dari *library* nya dapat dilihat pada tautan berikut https://github.com/twintproject/twint. Penulis menggunakan *library* twint dikarenakan *library* ini mampu untuk mengambil keseluruhan data twit pada sebuah akun Twitter, tanpa perlu menggunakan akses token dari Twitter.

Selain itu, penulis juga melakukan pengumpulan data berupa daftar kata-kata berbahasa indonesia, berbahasa inggris, *code-mixing* pada level kata, dan bahasa lainnya (seperti bahasa jawa, sunda, dan sebagainya). Data ini nantinya akan digunakan pada saat proses pelabelan *dataset*. Data ini dikumpulkan dari berbagai tempat, seperti Wikipedia, github, dan sebagainya.

3.3.2 *Preprocessing*

Tahap *preprocessing* merupakan tahap yang perlu dilakukan sebelum data digunakan untuk melakukan pelatihan model. Tahap *preprocessing* merupakan tahap yang krusial, karena jika tahap ini tidak dilakukan dengan benar, maka akan mempengaruhi hasil dari proses pelatihan. Umumnya tahap *preprocessing* akan memerlukan beberapa langkah, seperti pengahpusan *stop words* (kata tidak bermakna), penghapusan *emoticon*/*emoji*, penghapusan URL, dan sebagainya. Namun, pada penelitian yang saya lakukan terdapat perbedaan, mengingat data-data tersebut berperan sama pentingnya (semua penting) pada penelitian ini.

Pada tahap pertama, penulis melakukan *preprocessing* terhadap data bahasa. Pada tahap ini dilakukan penghapusan kata-kata yang tidak sesuai dengan bahasa yang dimaksud. Dikarenakan data dikumpulkan dari berbagai tempat, terdapat data yang tidak sesuai. Proses penghapusan kata-kata tersebut akan dilakukan secara manual oleh penulis. Pada tahap kedua, penulis melakukan *preprocessing* terhadap data twit. Pada tahap ini dilakukan proses penyaringan twit, berdasarkan kriteria seperti di bawah ini.

Tabel 3.1 Statistik penyaringan untuk α dan β

N	α	β
0	4	2
30	8	4

N merupakan jumlah minimal token pada kalimat, α merupakan jumlah minimal token bahasa indonesia, β sedangkan merupakan jumlah minimal token bahasa inggris. Proses ini perlu dilakukan agar *code-mixing* dan *code-switching* pada *dataset* terpenuhi. Selain itu, penulis juga melakukan penyaringan pada data retwit, dengan tujuan agar *dataset* hanya mengandung kumpulan twit pemilik akun yang diambil datanya. Dengan begitu, *dataset* dapat digunakan nantinya seperti penelitian terhadap kepribadian pemilik twit, atau yang semisalnya.

3.3.3 Pelabelan *Dataset* (Mesin)

Pada tahap ini, penulis melakukan pembuatan sistem sederhana yang bertujuan untuk mengurangi proses pelabelan *dataset* secara manual yang dilakukan oleh penulis. Penulis melakukan pembuatan model dengan menggunakan algoritma SVM, yang diimplementasikan dengan menggunakan *library scikit-learn*. Tujuan dari SVM ini yaitu untuk melakukan pelatihan model dalam level kata, dengan mengabaikan konteks. Dalam melakukan pelatihan model, penulis menggunakan data daftar kata yang telah dikumpulkan sebelumnya. Kemudian penulis membuat N-gram (2, 3) dari daftar kata tersebut, dengan menggunakan algoritma TF-IDF. Tujuan dari penggunaan N-gram ini yaitu untuk mendapatkan sub kata yang umum terdapat pada masing-masing bahasa. Data N-gram tersebut kemudian menjadi fitur pada metode SVM. Untuk mendapatkan bahasa target, penulis melakukan penggabungan antara pencarian leksikon dengan prediksi yang diperoleh dari proses pembelajaran model. Pertama sistem akan mencari kata pada daftar leksikon yang dimiliki, jika ditemukan maka sistem akan memberikan label bahasa sesuai leksikon. Namun, jika tidak ditemukan maka sistem akan menggunakan model SVM yang telah dilatih tersebut untuk memprediksi label bahasa yang sesuai.

Tabel 3.2 Tabel daftar label prediksi

Label	**Keterangan**
id	Bahasa Indonesia
en	Bahasa Inggris
id-en	Bahasa Indonesia-Inggris (*Code-mixed* level kata)
univ	*Universal.* Berlaku umum.
unk	*Unknown.* Bukan bahasa Indonesia / Inggris

Model SVM dilatih untuk melakukan prediksi pada label id, en, id-en dan unk. Sedangkan untuk mendapatkan label *universal*, ditambahkan sebuah *wrapper* sistem untuk mendapatkan kata yang termasuk ke dalam kategori *universal*. Penamaan label yang digunakan oleh penulis diadaptasi dari penamaan label yang digunakan pada *International Conference on Natural Language Processing* (ICON-2016).

3.3.4 Pelabelan *Dataset* (Manual)

Pada tahap ini, setelah twit telah dilakukan pelabelan secara otomatis oleh sistem, maka selanjutnya penulis melakukan koreksi manual terhadap hasil tersebut. Umumnya pengkoreksian ini dilakukan oleh 2 atau lebih anotator, agar dapat diperoleh hasil yang jelas dan konsisten. Dikarenakan model SVM melakukan pelatihan pada level kata (bukan level kalimat), akan dijumpai banyak kesalahan pemberian label yang tidak sesuai dengan konteksnya. Untuk melakukan proses koreksi, dibuatlah daftar panduan yang perlu diperhatikan.

a. Lihat apakah sebuah kata (kata) layak menjadi indonesia atau inggris. Jika layak menjadi indonesia, maka token sebelumnya / setelahnya harus berbahasa indonesia. Perhatikan apakah token tersebut lebih sesuai ikut kata setelahnya atau sesudahnya. Begitu juga untuk bahasa inggris. Namun jika token sebelumnya bukan bahasa indonesia / bahasa inggris, misal tanda baca, maka buat asumsi. Jika tidak layak, maka tandai dengan unknown.
b. Jika kata merupakan kata umum, seperti nama orang, nama jalan, nama negara. Ikuti kaidah poin a. Untuk nama negara, sudah ada panduan dari KBBI.
c. Jika kata merupakan kata indonesia, walaupun tidak baku (cak), selama kata tersebut layak menjadi bahasa indonesia, maka masukkan token tersebut ke dalam bahasa indonesia.
d. Jika sebuah token merupakan singkatan dari suatu kata, selama token tersebut tidak saltik (tertukar depan belakangnya, misalnya mnjadi, bukan mjndi) maka masukkan token tersebut ke dalam bahasa tujuan. Selanjutnya ikuti kaidah poin a.

Sedangkan dalam prosesnya, terdapat beberapa kebingungan yang terjadi, contohnya terdapat pada penjelasan di bawah ini

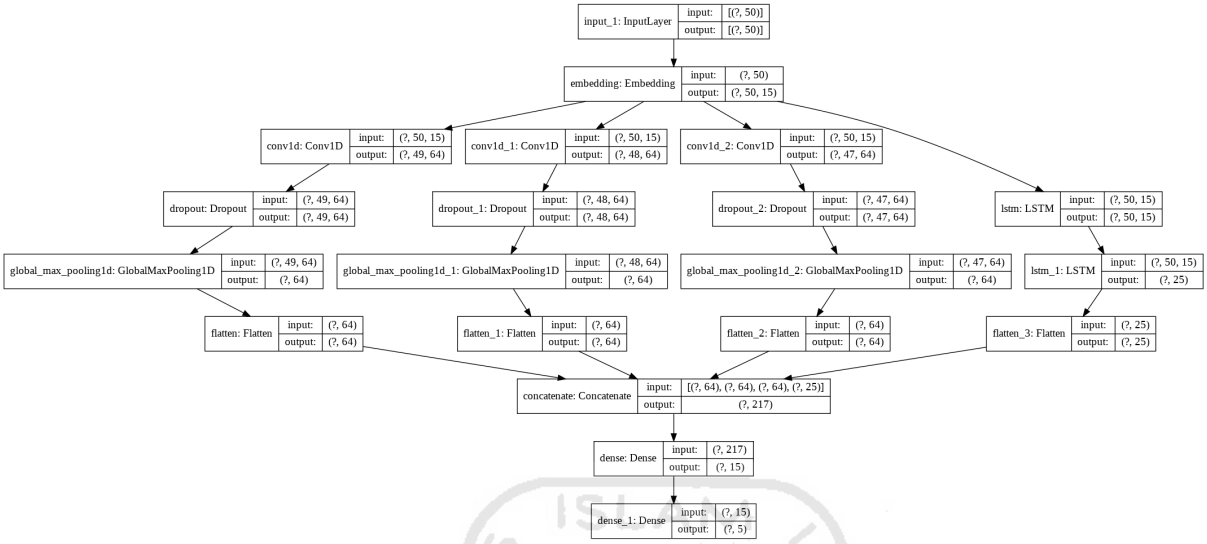
a. Apakah kata berikut masuk ke dalam bahasa Indonesia atau *Unknown*. Pada akhirnya masuk ke dalam bahasa Indonesia: Dy (dia), keapus (kehapus), ngamau (nggak mau), ngapapa (nggak apa-apa), nasi+sayur. Pada akhirnya masuk ke dalam *Unknown*: sifat/karakter/*look*, pekerjaannnya.
b. Apakah kata berikut masuk ke dalam bahasa Inggris atau *Unknown.* Pada akhirnya masuk ke dalam bahasa Inggris: *I-assume-my*.
c. Apakah kata berikut masuk ke dalam bahasa Indonesia atau Universal. Pada akhirnya masuk ke dalam bahasa Indonesia: 2jt (juta), 10rb (ribu).
d. Apakah kata berikut masuk ke dalam bahasa Inggris atau *Context Dependant* (sesuai konteks). Pada akhrinya sesuai konteks: hahaha, wkwkwk, 1M (*Million* / Miliar), 1T (Triliun).
e. Apakah kata berikut masuk ke dalam bahasa Inggris atau *Universal*. Pada akhirnya masuk ke dalam bahasa Inggris: 1B (Billion). Pada akhirnya masuk ke dalam *Universal*: 10Mbps.
f. Apakah kata berikut masuk ke dalam *Context Dependant* atau *Universal*. Pada akhirnya masuk ke dalam *Context Dependant*: "2." (2 titik). Pada akhirnya masuk ke dalam *Universal*: 2x (2 *times* / 2 kali), 10K (kilo), 28+6, 7+.

3.3.5 Penandaan Bahasa (MNN)

Pada tahap ini, akan dilakukan pelatihan model pada level kata dengan menggunakan metode MNN, yaitu penggabungan ataran metode CNN dan LSTM. Penulis menggunakan arsitektur model yang sama, yang digunakan oleh (Mandal & Singh, 2019). Penulis juga menggunakan *hyperparameter* yang sama dengan yang digunakan pada paper tersebut. Model dibangun dengan menggunakan *framework* Tensorflow dan Keras. Model di awali dengan *character embedding* dengan panjang dimensi sebesar 15. Kemudian *character embedding* diteruskan ke dalam 4 kanal jaringan syaraf. Pada bagian ini terdapat 3 channel *Convolutional 1D* yang digunakan untuk menangkap representasi *N-gram* dari suatu kata, dengan masing-masing memiliki *kernel* berjumlah 2, 3, 4 dan menggunakan fungsi aktivasi berupa relu. Ketiga lapisan ini diikuti oleh *dropout layer* sebesar 0.2, yang berfungsi untuk mengontrol *overfitting*.

Pada kanal ke 4, di sana terdapat 2 tumpukan LSTM dengan masing-masing memiliki jumlah *units* sebesar 15 dan 25. Selanjutnya, di masing-masing kanal ini dilakukan flatten, yang berfungsi untuk mengubah matriks multi dimensi menjadi vektor lalu dilakukan penggabungan. Terakhir, vektor tersebut dikirimkan ke 2 *dense layer* (*FC Layer* dan *Output*

Layer) dengan *FC Layer* berjumlah 15 *units* (fungsi aktivasi relu), dan *output layer* berjumlah 5 *units* (fungsi aktivasi softmax) karena penulis mengharapkan 5 prediksi label.

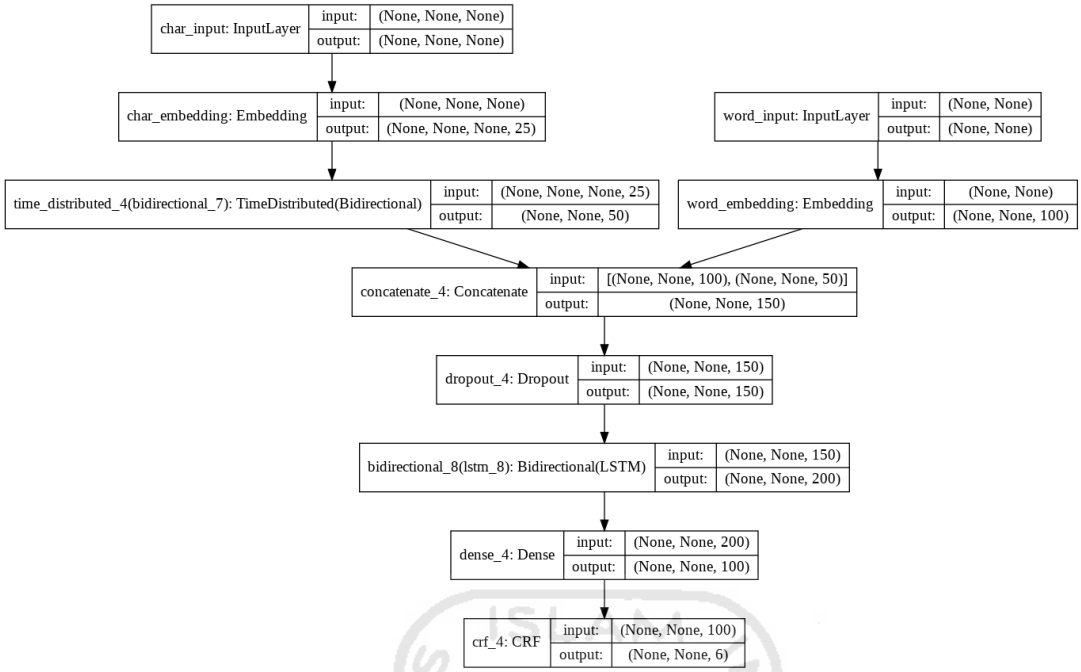


Gambar 3.2 Arsitektur MNN untuk pelatihan level kata

3.3.6 Penandaan Bahasa (Bi-LSTM-CRF)

Pada tahap ini, akan dilakukan pelatihan model pada level kalimat dengan menggunakan metode Bi-LSTM-CRF. Model ini akan menggunakan nilai keluaran berupa *softmax* dari model MNN, yang selanjutnya akan digabungkan dengan *character encoding.* Arsitektur dari model ini dapat dilihat pada Gambar 3.3. Model ini dibangun dengan menggunakan arsitektur yang dimiliki oleh Anago (https://github.com/Hironsan/anago), berbeda dengan model yang digunakan oleh (Mandal & Singh, 2019), di sana mereka menggunakan NCRF++ untuk membangun model. Sehingga, terdapat perbedaan juga pada sisi *hyperparameter* yang digunakan (penulis menggunakan *default hyperparameter* yang digunakan oleh Anago).

Hanya saja secara garis besar sama saja, arsitektur Bi-LSTM-CRF ini terdapat 2 *embedding*, yaitu *word embedding* dan *character embedding*. Setelah *character embedding*, diberikan Bi-LSTM *layer* untuk melakukan pembelajaran konteks pada level kata. Kemudian *character embedding* dan *word embedding* dilakukan penggabungan, di tambahkan *dropout* sebesar 0.5. Setelah itu, diberikan Bi-LSTM untuk melakukan pembelajaran pada level kalimat. Lalu ditambahkan *dense layer* dengan fungsi aktifasi tanh sebelum memasuki CRF *layer*.



Gambar 3.3 Arsitektur Bi-LSTM-CRF untuk pelatihan level kalimat

3.3.7 Evaluasi

Pada tahap ini, akan dilakukan proses evaluasi pada model. Untuk mengetahui seberapa bagus kinerja model ini, evaluasi akan dilakukan dengan menggunakan *confusion matrix*. Pada Tabel 3.3 dapat dilihat disana merupakan contoh *confusion matrix* pada data 2 kelas.

Tabel 3.3 Contoh *Confusion Matrix*

		Prediksi	
		Positif	Negatif
Target	Positif	TP	FP
	Negatif	FN	TN

Sedangkan persamaan yang digunakan untuk menghitung nilai pada *confusion matrix* dapat dilihat pada rumus-rumus di bawah ini

$$\text{Akurasi} = \frac{TP - TN}{TP + FP + TN + FP} \times 100\% \qquad (3.1)$$

$$\text{P}resisi = \frac{\text{TP}}{\text{T}P \ + \ FP} \times 100\% \qquad (3.2)$$

$$\text{R}ecall = \frac{\text{TP}}{\text{T}P \ + \ FN} \times 100\% \qquad (3.3)$$

$$\text{F1} = 2 \times \frac{Presisi \ * \ Recall}{Presisi + \ Recall} \times 100\% \qquad (3.4)$$

Keterangan:

False Positif *(FP)*	Jumlah data dengan kelas positif yang terklasifikasi salah oleh model
False Negatif *(FN)*	Jumlah data dengan kelas negatif yang terklasifikasi salah oleh model
True Positif *(TP)*	Jumlah data dengan kelas positif yang terklasifikasi benar oleh model
True Negatif *(TN)*	Jumlah data dengan kelas negatif yang terklasifikasi benar oleh model

BAB IV
HASIL DAN PEMBAHASAN

4.1 Penyelesaian Masalah

Dari permasalahan yang penulis sebutkan pada subbab 3.1, penulis kemudian membuat sebuah sistem yang dapat digunakan untuk melakukan penandaan bahasa dengan benar pada sebuah kalimat *code-mixing* dan *code-switching*. Secara ringkas yang dilakukan oleh penulis yaitu melakukan pelatihan sistem agar dapat membedakan mana kata yang termasuk ke dalam bahasa Indonesia, mana kata yang termasuk ke dalam bahasa Inggris dan mana kata yang tidak termasuk keduanya. Sebagai contoh, dari kalimat yang penulis sebutkan pada subbab 3.1, sistem yang dikembangkan oleh penulis akan berusaha untuk memberikan penandaan bahasa sebagai berikut

"Knp**id** nama**id** org**id** Bali**id** sering**id** diawali**id** dengan**id** kata**id** I**id** ?**univ** cth**id** *simple*nya**id-en** I**id** Gusti**id** Ngurah**id** Rai**id** .**univ**"

Sistem yang dikembangkan oleh penulis akan memberikan penandaan bahasa (label) pada masing-masing kata dan juga termasuk tanda baca. Untuk mengetahui penjelasan mengenai label tersebut, dapat dilihat pada Tabel 3.2. Namun, kalimat di atas juga dapat diberikan penandaan seperti berikut

"Knp**id** nama**id** org**en** Bali**id** sering**id** diawali**id** dengan**id** kata**id** I**en** ?**univ** cth**id** *simple*nya**en** I**en** Gusti**id** Ngurah**unk** Rai**id** .**univ**"

Hasil penandaan tersebut jika kita lihat secara kata per kata tidak bisa dikatakan jika penandaan tersebut salah. Sebagai contoh, kata *I* juga terdapat pada bahasa Inggris sehingga jika sistem berasumsi bahwa *I* merupakan kata bahasa Inggris tentu tidak salah. Hanya saja, jika penandaan tersebut dilihat dari konteksnya, maka penandaan di atas tentu salah. Karena *I* yang dimaksud pada I Gusti Ngurah Rai merupakan nama orang yang diawali dengan I, sehingga I merupakan bahasa Indonesia, bukan bahasa Inggris. Agar dapat mengetahui kata sesuai dengan konteksnya, digunakanlah metode bernama CRF (*Conditional Random Field*). Secara ringkas, penulis menggunakan metode MNN (LSTM dan CNN) untuk memberikan

prediksi probabilitas pada masing-masing kata (dan juga tanda baca). Kemudian, hasil dari prediksi tersebut, diberikan kepada metode CRF. Selanjutnya metode CRF akan melakukan pemberian bobot pada masing-masing kata di setiap *sequence* (kata yang terletak sebelum dan sesudahnya). Jika kata tersebut lebih banyak muncul pada *sequence* bahasa tertentu (sesuai dengan bahasa yang telah dipelajari dari *dataset*), maka CRF akan memasukkannya ke bahasa tersebut.

4.2 Hasil Pengumpulan Data

Data twit yang dikumpulkan untuk penelitian ini diperoleh sebesar 125.429 twit. Data twit tersebut masih merupakan data mentah yang perlu dilakukan *preprocessing*, dikarenakan tidak semua data pada twit tersebut mengandung *code-mixing* dan *code-switching*. Selanjutnya, penulis melakukan pengumpulan data bahasa dari berbagai sumber. Detail pengumpulan data bahasa yang dilakukan oleh penulis adalah sebagai berikut

a. Penulis mengumpulkan data bahasa Indonesia dari *dump* artikel Wikipedia sampai tanggal 17 Juli 2020. Kemudian penulis memanfaatkan algoritma TF-IDF (pada scikit-learn) untuk melakukan ekstraksi fitur berupa kata dengan minimum DF sebesar 0,01 (ini merupakan nilai yang di pilih oleh penulis setelah melalui beberapa rangkaian uji coba, karena menghasilkan lebih sedikit *noise* namun tetap memperoleh banyak data).
b. Mengumpulkan data bahasa Indonesia dari daftar akar kata.
c. Mengumpulkan data bahasa Indonesia dan Inggris dari daftar *slang words* yang dihimpun dari berbagai sumber.
d. Mengumpulkan data bahasa Inggris dari https://github.com/dwyl/english-words. Data ini sebenarnya terlalu banyak *noise*, sayangnya penulis hanya dapat memperoleh data tersebut.
e. Mengumpulkan data bahasa lain (jawa dan sunda) dari dataset yang dimiliki oleh (Dwi, 2020).

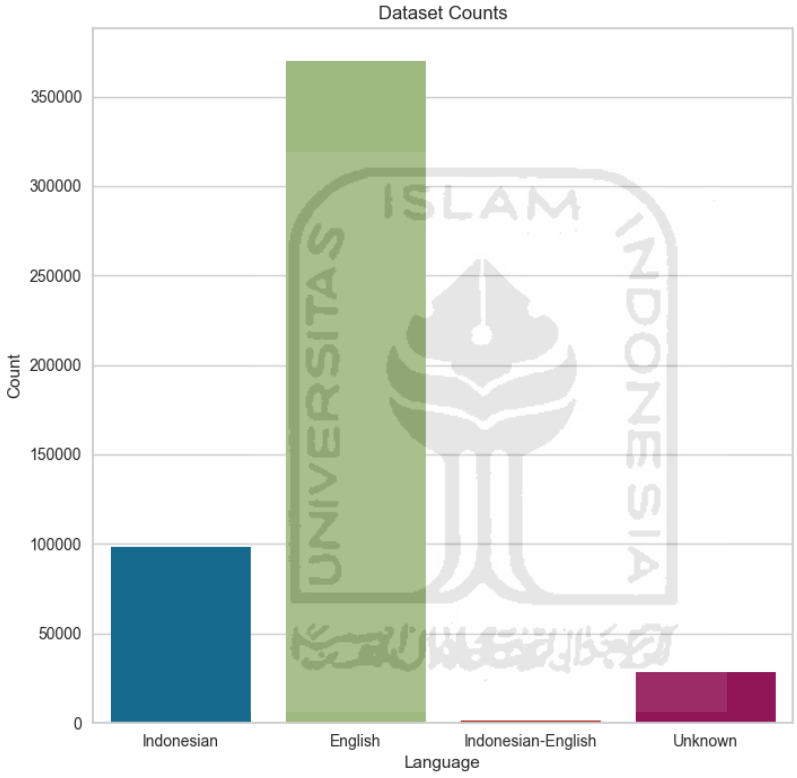
4.3 *Preprocessing*

Pada tahap ini, penulis melakukan *preprocessing* terhadap data yang telah diperoleh pada subbab 4.1. Tahapan *preprocessing* yang dilakukan penulis yaitu

a. Penulis melakukan pembersihan kata secara manual pada data yang diperoleh dari Wikipedia, mengingat TF-IDF akan banyak mengambil seperti kata saltik (salah ketik), karena kata saltik pasti mempunyai nilai IDF yang cukup besar.

b. Penulis menyeleksi satu persatu dan membuang kata yang tidak termasuk bahasa Indonesia dan bahasa Inggris dari masing-masing data *slang words*.

c. Penulis melakukan proses penyaringan data twit yang telah diperoleh, dengan menggunakan kriteria yang telah dijelaskan pada subbab 3.3.2. Data hasil penyaringan inilah yang nantinya akan digunakan untuk pelatihan dan pengujian pada metode MNN dan Bi-LSTM-CRF. Hasil penyaringan diperoleh 13.474 twit, kemudian penulis membagi sebesar 12.684 untuk data latih, dan 790 sebagai data uji.



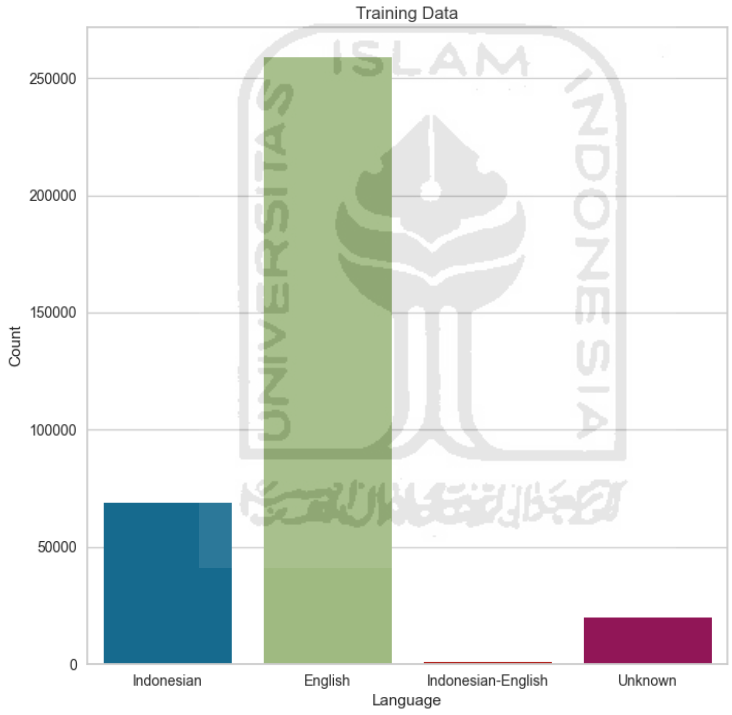
Gambar 4.1 Grafik distribusi keseluruhan kata

Di atas merupakan grafik distribusi kata yang diperoleh untuk masing-masing kelas. Dapat kita lihat jika terdapat kata bahasa Indonesia sebesar 98.343 kata, kata bahasa inggris sebesar 369.742 kata, kata bahasa Indonesia-Inggris sebesar 1.020 kata, dan kata *unknown* sebesar 28.266 kata.

4.4 Pelabelan *Dataset* (Mesin)

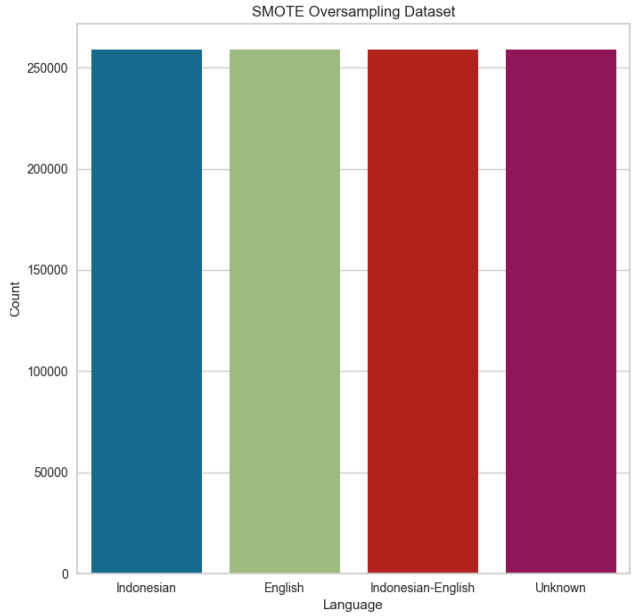
Pada tahap ini, dilakukan proses pelatihan model dengan menggunakan algoritma SVM. Algoritma SVM dilatih dengan menggunakan fitur N-gram (2, 3) yang didapat dari daftar data

latih. Data latih diperoleh dengan mengambil sebanyak 70% dari keseluruhan data, sedangkan sisanya merupakan data uji. Detail data latih dapat dilihat pada Gambar 4.2 di bawah. Untuk membuat fitur N-gram (2, 3) penulis menggunakan algoritma TF-IDF pada *scikit-learn* dengan minimum DF sebesar 0.000001. Fitur N-gram dan nilai TF tersebut penulis dapatkan dari hasil uji coba pada 3 kali percobaan, yaitu dengan nilai N-gram (1, 3), (2, 3), (2, 4) serta DF dengan nilai 0.01, 0.00001, 0.000001. Penulis mendapatkan akurasi model terbaik ketika menggunakan N-gram sebesar (2, 3) dan DF sebesar 0.000001. Penulis tidak menggunakan N-gram yang lebih banyak seperti (2, 8) karena menurut hemat penulis, penulis hanya akan mendapatkan kumpulan sub kata yang tidak begitu penting (bukan sub kata umum yang terdapat pada masing-masing bahasa).



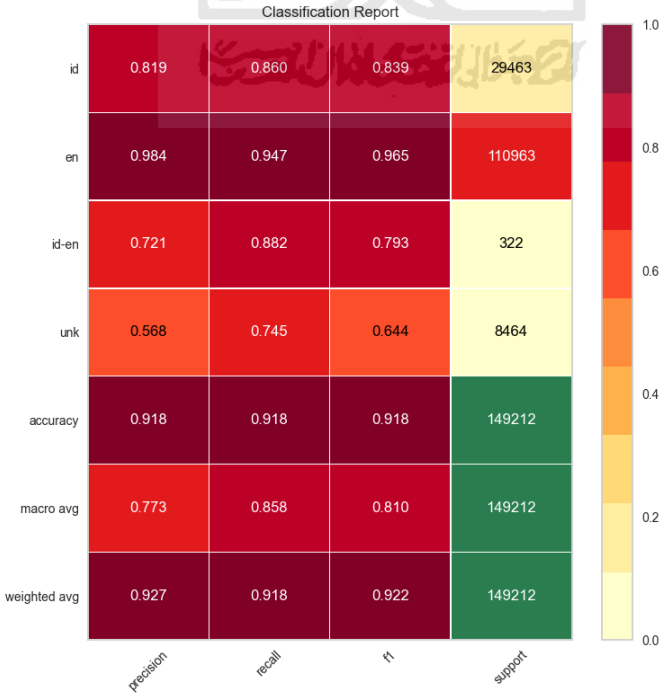
Gambar 4.2 Grafik distribusi data latih SVM

Dari data tersebut, terlihat data latih sangat tidak seimbang. Oleh karena itu, penulis menerapkan metode SMOTE *Oversampling* pada data latih tersebut. Sehingga, hasilnya data latih menjadi seperti pada Gambar 4.3.

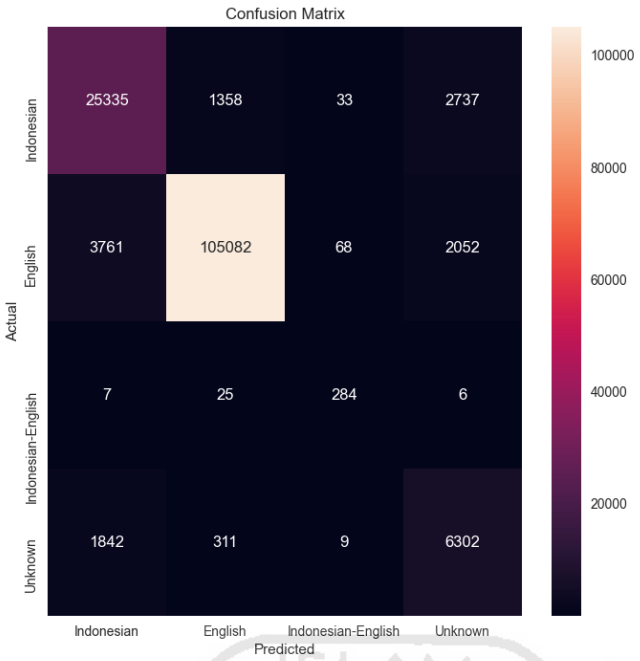


Gambar 4.3 Grafik distribusi data latih SVM setelah *oversampling*

Setelah dilakukan pelatihan model, SVM memperoleh hasil yang cukup baik, yaitu sebesar 91.80% akurasi dan 92.20% f1 *score*. Itu artinya, model SVM sudah cukup baik untuk membantu penulis melakukan pelabelan *dataset*. Detail hasil dari proses pelatihan tersebut dapat dilihat pada gambar *classification report* dan *confusion matrix* di bawah.



Gambar 4.4 *Classification Report* SVM



Gambar 4.5 *Confusion Matrix* SVM

4.5 Pelabelan *Dataset* (Manual)

Pada tahap ini, penulis melakukan koreksi pelabelan yang telah dilakukan oleh SVM secara manual. Penulis menyesuaikan beberapa ribu kata agar sesuai dengan konteksnya. Proses koreksi dengan memperhatikan panduan yang telah dibuat pada subbab 3.2.4. Tujuannya, ketika data digunakan untuk pelatihan model, model dapat memperoleh hasil yang baik, sehingga mampu melakukan prediksi yang tepat sesuai dengan konteksnya.

4.6 Penandaan Bahasa (MNN)

Pada tahap ini, penulis melakukan pelatihan pada model yang sudah dibangun. Pelatihan dilakukan dengan menggunakan fungsi fit() yang dimiliki oleh *framework* Keras. Proses pelatihan dilakukan dengan menggunakan *validation_split* sebesar 0.33%, *epoch* sebesar 50, jumlah *batch* sebanyak 64 dan menggunakan *callback* bernama *EarlyStopping* yang berfungsi untuk memonitor val_loss agar model tidak *overfit*. Sehingga, ketika penulis menggunakan nilai epoch yang cukup besar pun (misalnya 200), maka model tidak akan *overfit* karena *callback* tersebut akan menghentikan proses pelatihan secara otomatis ketika model menunjukkan tanda-tanda *overfit*.

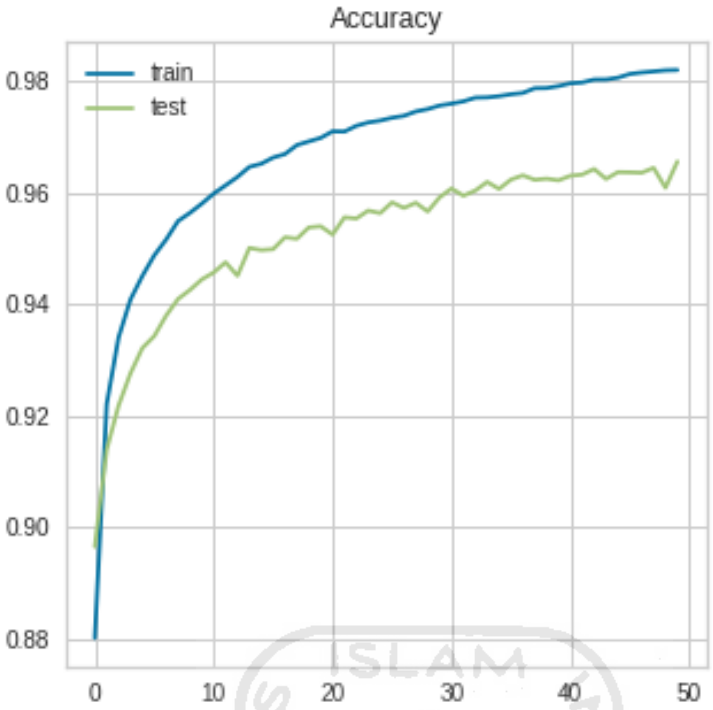
Jumlah *batch* yang penulis gunakan pada pelatihan ini, penulis dapatkan dari 5 pengujian yang telah dilakukan (dengan jumlah *batch* yang diujikan yaitu 32, 48, 64, 72, 128) dan penulis

mendapatkan hasil terbaik ketika menggunakan *batch* sebesar 64. Sedangkan nilai dari *validation_split*, merupakan nilai yang dipilih oleh penulis (yang dirasa cukup untuk memverifikasi bahwa model telah terlatih dengan baik), dengan melihat jumlah *dataset* yang dimiliki oleh penulis. Umumnya, penentuan nilai *validation_split* yang digunakan akan semakin kecil jika *dataset* yang dimiliki hanya sedikit, begitu juga sebaliknya. Sedangkan nilai *epoch* juga dipilih secara acak oleh penulis, dengan memilih nilai yang menurut hemat penulis cukup untuk melakukan pelatihan model. Selain itu, dapat dilihat pada proses pelatihan yang ditunjukkan oleh Gambar 4.6, nilai *validation_accuracy* menunjukkan jika sudah tidak ada peningkatan, sehingga nilai *epoch* yang dipilih oleh penulis dirasa sudah tepat. Berikut merupakan riwayat dari proses pelatihan.

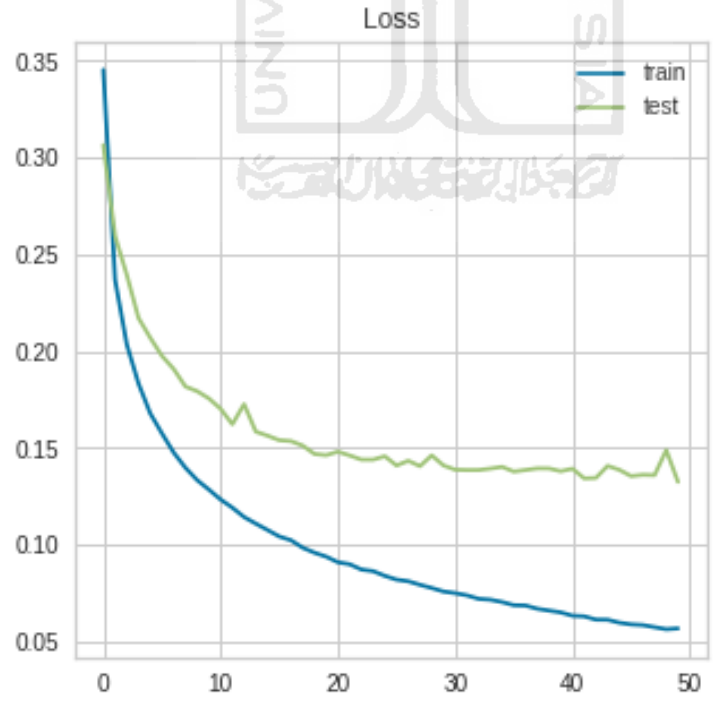
```
Epoch 35/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0704 - accuracy: 0.9773 - val_loss: 0.1402 - val_accuracy: 0.9607
Epoch 36/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0688 - accuracy: 0.9777 - val_loss: 0.1378 - val_accuracy: 0.9623
Epoch 37/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0686 - accuracy: 0.9779 - val_loss: 0.1386 - val_accuracy: 0.9631
Epoch 38/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0669 - accuracy: 0.9788 - val_loss: 0.1394 - val_accuracy: 0.9623
Epoch 39/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0660 - accuracy: 0.9788 - val_loss: 0.1394 - val_accuracy: 0.9625
Epoch 40/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0650 - accuracy: 0.9791 - val_loss: 0.1380 - val_accuracy: 0.9622
Epoch 41/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0633 - accuracy: 0.9796 - val_loss: 0.1393 - val_accuracy: 0.9630
Epoch 42/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0630 - accuracy: 0.9798 - val_loss: 0.1343 - val_accuracy: 0.9633
Epoch 43/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0613 - accuracy: 0.9803 - val_loss: 0.1345 - val_accuracy: 0.9643
Epoch 44/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0612 - accuracy: 0.9803 - val_loss: 0.1408 - val_accuracy: 0.9625
Epoch 45/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0597 - accuracy: 0.9806 - val_loss: 0.1387 - val_accuracy: 0.9637
Epoch 46/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0588 - accuracy: 0.9813 - val_loss: 0.1354 - val_accuracy: 0.9637
Epoch 47/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0585 - accuracy: 0.9816 - val_loss: 0.1361 - val_accuracy: 0.9636
Epoch 48/50
2698/2698 [==============================] - 38s 14ms/step - loss: 0.0574 - accuracy: 0.9818 - val_loss: 0.1359 - val_accuracy: 0.9645
Epoch 49/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0563 - accuracy: 0.9820 - val_loss: 0.1489 - val_accuracy: 0.9609
Epoch 50/50
2698/2698 [==============================] - 37s 14ms/step - loss: 0.0567 - accuracy: 0.9820 - val_loss: 0.1325 - val_accuracy: 0.9656
```

Gambar 4.6 Riwayat pelatihan MNN

Pada pelatihan model ini, digunakan *loss* berupa *categorical crossentropy*, *optimizer* adam, dan metrics berupa *accuracy*. Dapat dilihat pada riwayat akurasi berupa grafik di bawah, terlihat dengan jelas jika nilai *validation accuracy* yang berwarna hijau, cenderung tidak mengalami peningkatan. Begitu juga yang terjadi pada nilai *validation loss*. Sehingga, nilai *epoch* sebesar 50 merupakan nilai yang tepat untuk digunakan pada pelatihan model ini, karena jika proses pelatihan masih terus dilanjutkan, terdapat kemungkinan jika model akan *overfit*.

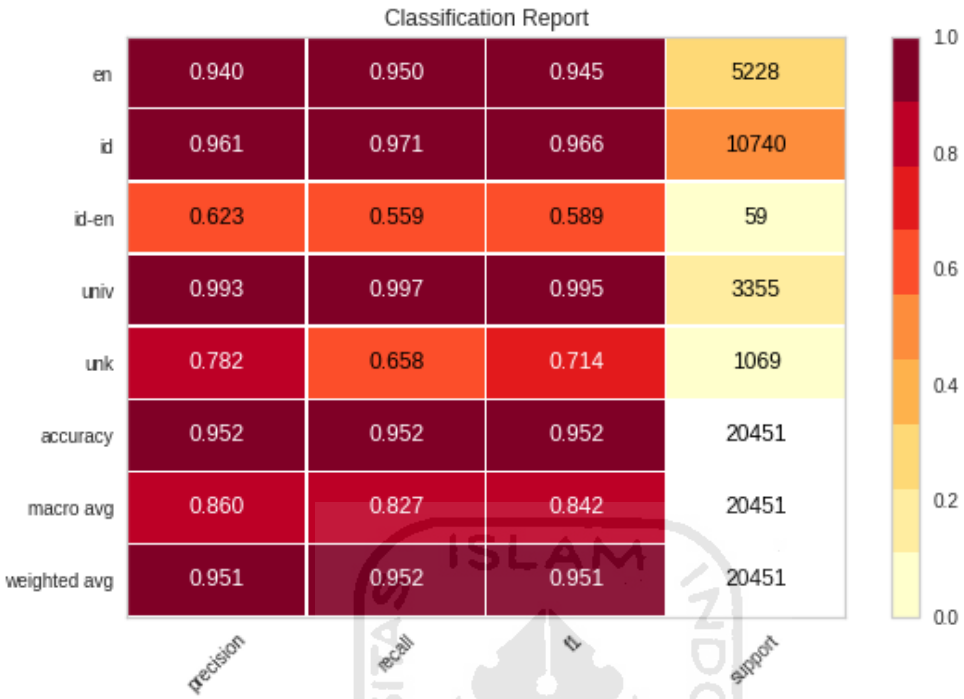


Gambar 4.7 Riwayat akurasi MNN

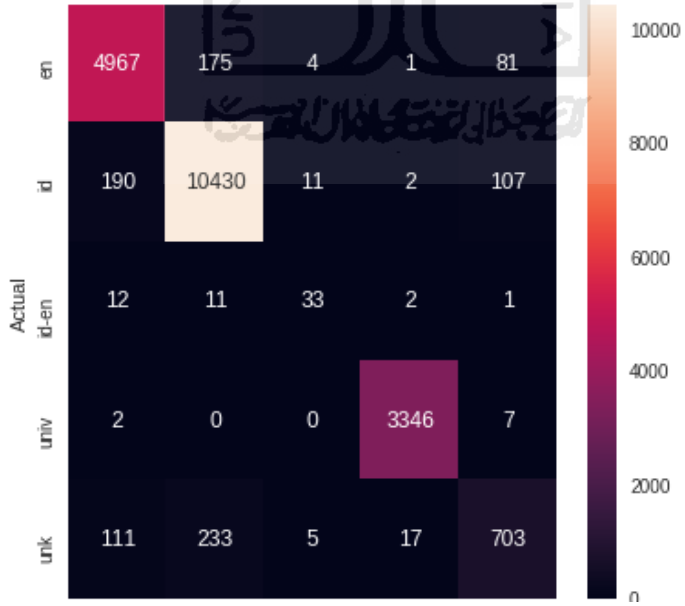


Gambar 4.8 Riwayat *loss* MNN

Untuk mengetahui lebih lanjut performa model, dapat dilihat pada *classification report* dan *confusion matrix* di bawah ini:



Gambar 4.9 *Classification Report* MNN



Gambar 4.10 *Confusion Matrix* MNN

Pada Gambar 4.9, dapat dilihat jika akurasi model mencapai 95.2% dan f1 mencapai 95.1%. Dapat disimpulkan jika model berjalan dengan baik untuk melatih data pada level kata.

Begitu juga dengan *confusion matrix*nya, terlihat bahwa jumlah kesalahan prediksi tidak begitu signifikan.

Selanjutnya, penulis akan mencoba untuk melakukan prediksi sebuah kalimat menggunakan MNN. Lalu nantinya penulis akan membandingkan hasilnya dengan hasil prediksi dari Bi-LSTM-CRF.

```
[11] tagged_txt = []
     for idx, token in enumerate(results):
       tagged_txt.append('%s\%s' % (token, label_encoder.classes_[predictions[idx]]))

     print(' '.join(tagged_txt))

     rasain\id lu\id masuk\id got\en .\univ enak\id kan\id ?\univ
```

Gambar 4.11 Hasil prediksi model MNN

Penulis melakukan prediksi menggunakan MNN untuk kalimat "rasain lu masuk got. enak kan?". Sesuai dengan yang seharusnya, MNN menghasilkan label en pada kata "*got*", karena memang secara probabilitas, "*got*" (dapat) lebih layak masuk ke dalam bahasa inggris. Sedangkan jika dilihat secara konteks kalimat, kata "got" disana yang benar seharusnya mendapatkan label "id", karena "got" yang dimaksud disitu yaitu tempat pembuangan air.

4.7 Penandaan Bahasa (Bi-LSTM-CRF)

Pada tahap ini, merupakan tahap terakhir dari pelatihan model, yaitu data akan dilakukan pelatihan pada level kalimat, sehingga nantinya bisa diperoleh konteks dari suatu kata. Model dibuat dengan menggunakan anago. Anago merupakan *tools open source* yang dikembangkan pada framework tensorflow 1.8.0, sedangkan model MNN yang penulis kembangkan menggunakan framework tensorflow 2.3.0. Sehingga pada penerapannya terdapat sedikit kesulitan, itu juga tidak terlepas dari belum jelasnya penerapan CRF pada tensorflow (mungkin ini sebatas kekurangan pengetahuan penulis).

Sebelum model dilakukan pelatihan, data dilakukan konversi ke dalam bentuk index (*word and character encoding*). Kemudian data juga dimasukkan ke dalam MNN, untuk mendapatkan prediksi berupa *softmax*. Hasil prediksi berupa softmax ini kemudian ditambahkan ke dalam *character embedding*. Sehingga, fitur pada *character embedding* merupakan keluaran *softmax* MNN + *Character encoding*.

Selanjutnya pelatihan dilakukan dengan menggunakan fungsi fit_generator() yang dimiliki oleh *framework* Keras versi 2.2.4. Proses pelatihan dilakukan dengan menggunakan *epoch* sebesar 10 (model tidak mengalami kenaikan selama 10 *epoch*, sehingga dicukupkan 10 agar tidak *overfit*), jumlah *batch* sebanyak 64 (dipilih secara acak) dan menggunakan *callback* bernama *EarlyStopping (*yang berfungsi untuk memonitor *val_loss* agar model tidak *overfit)* dan *F1Score* (digunakan untuk menghitung *f1 score* pada setiap *epoch*). Berikut merupakan riwayat dari proses pelatihan.

```
       univ       0.99      0.99      0.99      2886
         en       0.81      0.82      0.81      2674
        unk       0.44      0.68      0.53       983

avg / total       0.80      0.83      0.81     10495

Epoch 7/10
199/199 [==============================] - 71s 357ms/step - loss: 1.4593
 - f1: 80.29
             precision    recall  f1-score   support

         id       0.74      0.76      0.75      3952
       univ       0.98      0.99      0.99      2886
         en       0.70      0.82      0.76      2674
        unk       0.61      0.58      0.60       983

avg / total       0.79      0.82      0.80     10495

Epoch 8/10
199/199 [==============================] - 72s 360ms/step - loss: 1.4582
 - f1: 81.94
             precision    recall  f1-score   support

         id       0.76      0.78      0.77      3952
       univ       0.99      0.99      0.99      2886
         en       0.85      0.81      0.83      2674
        unk       0.47      0.66      0.55       983

avg / total       0.82      0.83      0.82     10495

Epoch 9/10
199/199 [==============================] - 71s 357ms/step - loss: 1.4572
 - f1: 80.88
             precision    recall  f1-score   support

         id       0.75      0.77      0.76      3952
       univ       0.99      0.99      0.99      2886
         en       0.80      0.81      0.81      2674
        unk       0.46      0.67      0.54       983

avg / total       0.80      0.83      0.81     10495

Epoch 10/10
199/199 [==============================] - 70s 354ms/step - loss: 1.4561
 - f1: 81.59
             precision    recall  f1-score   support

         id       0.75      0.77      0.76      3952
       univ       0.99      0.98      0.99      2886
         en       0.85      0.80      0.83      2674
        unk       0.47      0.65      0.55       983

avg / total       0.82      0.83      0.82     10495
```

Gambar 4.12 Riwayat Pelatihan Bi-LSTM-CRF

Setelah dilakukan proses pelatihan, didapatkan nilai *f1 score* model Bi-LSTM-CRF yaitu 81.59%. Hasil tersebut terlihat kurang baik, mengingat kita telah menggunakan CRF *layer* agar bisa mendapatkan konteks kata, tapi ternyata hasil yang diperoleh kurang baik. Hal ini kemungkinan terjadi karena *dataset* yang dimiliki oleh penulis tidak cukup bagus jika ingin digunakan untuk mengambil konteks kata.

Hasil tersebut penulis memandang kurang baik, mengingat penelitian yang dijadikan rujukan oleh penulis mendapatkan f1 *score* sebesar 93.32%. Hasil tersebut memang didapatkan dengan data yang berbeda dan *hyperparameter* yang berbeda. Hanya saja secara model, model yang digunakan penulis mempunyai kemiripan yang sama dengan model yang digunakan pada rujukan penulis. Oleh sebab itu, penulis menganggap hasil f1 *score* yang penulis dapatkan terbilang tidak terlalu baik.

Namun, kita dapat mencoba untuk melakukan pengujian yang sama seperti yang dilakukan dengan metode MNN, yaitu melakukan uji coba pada sebuah kalimat dan di sini kita akan menggunakan kalimat yang sama. Kita akan menguji apakah model dengan 81.59% f1 *score* ini bisa mendapatkan konteks dari kata "got" pada kalimat tadi.

```
[56] for i, sentence in enumerate(sentences):
       result = []
       for j, word in enumerate(sentence):
         result.append('%s\\\%s' % (word, tags[i][j]))
       print(' '.join(result))

     rasain\\id lu\\id masuk\\id got\\id .\\univ enak\\id kan\\id ?\\univ
```

Gambar 4.13 Hasil prediksi model Bi-LSTM-CRF

Ternyata setelah dilakukan pengujian, model Bi-LSTM-CRF dengan f1 score 81.59% ini dapat mengetahui konteks dari kata "got" pada kalimat tersebut. Sehingga, walaupun model Bi-LSTM-CRF hanya mempunyai nilai f1 *score* sebesar 81.59%, tetapi dalam kasus ini model Bi-LSTM-CRF dapat melakukan penandaan bahasa dengan benar.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian mengenai Penandaan Bahasa Indonesia-Inggris dari data *code-switching* dan *code-mixing* yang telah dilakukan dengan menggunakan metode MNN (CNN dan LSTM) memperoleh hasil akurasi yang sangat baik yaitu sebesar 95.20%, sedangkan untuk Bi-LSTM-CRF hanya memperoleh f1 *score* sebesar 81.59%. Hal ini menunjukkan jika model MNN dapat melakukan identifikasi bahasa dengan sangat baik, sedangkan untuk Bi-LSTM-CRF tidak terlalu baik dalam melakukan identifikasi bahasa sesuai konteks. Hal ini menurut hemat penulis, karena kurangnya pengetahuan penulis tentang penggunaan CRF *layer* yang tepat pada *framework* Keras / Tensorflow. Namun, walaupun model Bi-LSTM-CRF hanya mempunyai f1 score sebesar 85.59%, model tersebut dapat mendapatkan konteks dari suatu kata dengan benar pada kalimat yang diujikan pada subbab 4.7.

5.2 Saran

Dari penelitian yang telah dilakukan ini, terdapat beberapa saran yang dapat dilakukan untuk penengembangan selanjutnya:

a. Perlunya membuat dataset yang lebih baik lagi. Dataset yang dimiliki penulis belum cukup baik, mengingat data tersebut hanya dilakukan verifikasi seorang diri penulis, sehingga belum sempurna.
b. Menambah data latih lebih banyak lagi. Tujuannya agar model dapat dengan baik melakukan prediksi.
c. Mencoba menggunakan CRF layer pada tensorflow terbaru. Harapannya bisa diperoleh hasil f1 *score* yang lebih baik.

DAFTAR PUSTAKA

Aribowo, A. S. (2018). Analisis Sentimen Publik pada Program Kesehatan Masyarakat menggunakan Twitter Opinion Mining, 17–23.

Asror, M. (2009). Teachers Explaining Techniques: Code-Switching and Code-Mixing in the Classroom, 59.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique Nitesh. *Journal of Artificial Intelligence Research*. https://doi.org/10.1613/jair.953

Chollet, F. (2017). *Deep Learning with Python*.

Dwi, W. A. (2020). Penerapan Deep Learning untuk Deteksi Teks Bahasa Indonesia Jawa dan Sunda pada Media Sosial Twitter.

Fachri, M. (2014). Pengenalan Entitas Bernama Pada Teks Bahasa Indonesia Menggunakan Hidden Markov Model.

Firmansyah, D. (2019). Pengaruh Bahasa Indonesia dan Bahasa Inggris di Era Globalisasi. https://doi.org/10.31227/osf.io/uch36

Gumperz, J. (1982). Discourse Strategies. *NM: Cambridge University Press*.

Jaariyah, N., & Rainarli, E. (2017). Conditional Random Fields Untuk Pengenalan Entitas Bernama Pada Teks Bahasa Indonesia. *Komputa : Jurnal Ilmiah Komputer dan Informatika*, *6*(1), 29–34. https://doi.org/10.34010/komputa.v6i1.2474

Mahootian, S. (2006). Code Switching and Mixing. In K. Brown (Ed.), *Encyclopedia of Language and Linguistics* (Second). Oxford: Elsevier.

Manaswi, N. K. (2018). *Deep Learning with Applications Using Python*. Apress. https://doi.org/10.1007/978-1-4842-3516-4

Mandal, S., & Singh, A. K. (2019). Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture, (August), 116–120. https://doi.org/10.18653/v1/w18-6116

Marasigan, E. (1983). Code-Switching and Code-Mixing in Multilingual Societies. *Singapore: Singapore University Press for SEAMO Regional Language Center*.

Nayoan, R. A. N. (2019). Analisis Sentimen Berbasis Fitur Pada Ulasan Tempat Wisata Menggunakan Metode Convolutional Neural Network (CNN).

Nugroho, A. S., Witarto, A. B., & Handoko, D. (2011). Support Vector Machine: Teori dan Aplikasinya dalam Bioinformatika. *Proceedings of the 2011 Chinese Control and*

Decision Conference, CCDC 2011, 842–847.

Yuliana, N., Luziana, A. R., & Sarwendah, P. (2015). Code-Mixing and Code-Switching of Indonesian Celebrities: A Comparative Study. *Lingua Cultura*, *9*(1), 47. https://doi.org/10.21512/lc.v9i1.761

LAMPIRAN