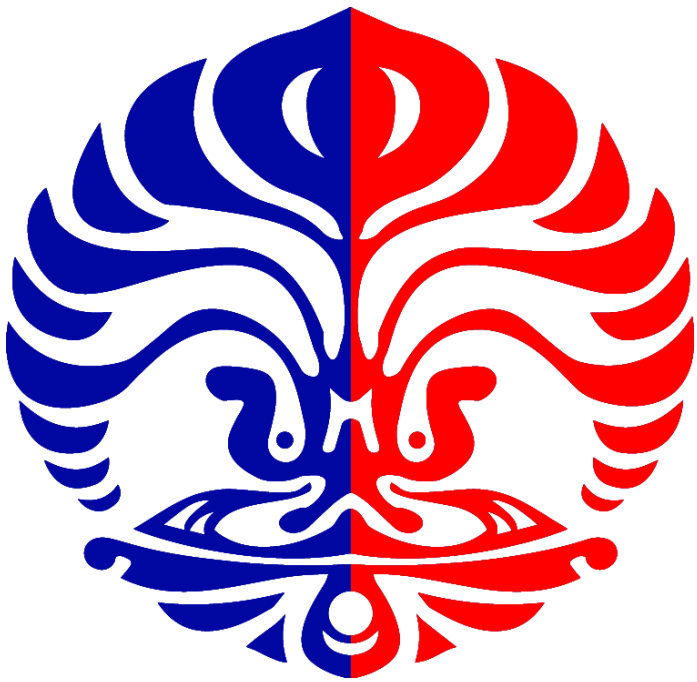
LAPORAN TUGAS

Natural Language Processing

Named Entity Recognition



Fithriannisa Augustianti - 1306381950
Septiviana Savitri - 1306381742
Valdi Rachman - 1306381862

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Named Entity merupakan suatu frasa berupa nama orang, lokasi, organisasi, dan sebagainya[1]. *Named Entity Recognition* (NER) merupakan sebuah tugas untuk mengenali dan menentukan sebuah *class* sebagai sebutan untuk nama yang spesifik untuk seseorang atau suatu instansi dalam sebuah teks[2]. NER dikenal sebagai komponen inti dari sistem *question answering* dan telah dikembangkan sebagai komponen untuk sistem *information extraction*[3].

Menurut Kominfo[4], pada tahun 2013 Indonesia berada di peringkat ke 5 sebagai negara dengan jumlah pengguna Twitter terbesar di dunia dengan jumlah pengguna sebesar 19,5 juta pengguna. Dengan banyaknya jumlah pengguna, tentu menyebabkan banyak *tweet* yang dihasilkan oleh pengguna di Indonesia setiap harinya. Setiap *tweet* tersebut terkadang mengandung informasi-informasi penting seperti kabar arus lalu lintas di musim pulang kampus, kejadian di berbagai tempat, suasana tempat wisata di hari libur dan sebagainya yang mana informasi tersebut berguna untuk menjawab pertanyaan-pertanyaan yang belum terjawab maupun pertanyaan-pertanyaan yang akan muncul di masa depan. NER *tagger* untuk *tweet* Bahasa Indonesia dikembangkan untuk mengekstrak informasi-informasi penting yang dapat menjawab beberapa pertanyaan yang muncul.

Deep learning terbukti dapat menyelesaikan task NLP dengan baik, sebagaimana pada paper Collobert et al. (20XX). Keunggulan dari *Deep Learning* adalah dapat mengekstrak fitur secara otomatis, tanpa harus secara eksplisit ditulis. Fitur yang kerap digunakan dalam *deep learning* adalah *word embedding.*

1.2 Studi Literatur

Pada tugas kali ini, selain NER penulis juga menggunakan metode *Deep Learning*. Bahan acuan yang digunakan oleh penulis dalam mengerjakan tugas ini berasal dari beberapa sumber sebagai berikut.

1. NER

   Permasalahan Named Entity Recognition adalah bagian dari bidang Information Extraction (IE). Menurut (Ghrisman dan Sundheim, 1996), masalah mengenai name entity pertama kali dibahas pada konferensi MUC-6. Pada konferensi tersebut, nama yang menjadi topik pembahasan adalah Person, Place dan Organization. Dalam konferensi ini juga diperkenalkan tag-tag yang mengidentifikasi nama pada teks, yaitu tag ENAMEX(“entity name expression”) dan tag NUMEX (“numeric expression”). Pemanfaatan NER di dunia Machine Learning dapat membantu pembuatan mesin lain. Untuk Bahasa Inggris, nilai evaluasi (Precission, Recall maupun F-1 score) untuk permasalahan NER sudah mencapai nilai yang stabil. Banyak tools yang langsung

dapat digunakan untuk NER bahasa Inggris. Namun, untuk bahasa Indonesia, bidang ini masih menjadi bidang yang memberikan peluang kontribusi bagi para peneliti. Dikarenakan sumbernya yang belum sebanyak bahasa Inggris, performa NER Indonesia dapat dikatakan masih rendah.

Beberapa penelitian sebelumnya mengenai NER sudah dilakukan oleh [5] dan [6]

BAB 2
METODOLOGI

Bab ini akan menjelaskan mengenai metodologi yang digunakan oleh penulis dalam penelitian ini. Metodologi yang dilakukan meliputi tahap pengumpulan data, tahap pra-pemrosesan data, tahap pengembangan model, tahap eksperimen, dan tahap evaluasi.

1.1 Gambaran Umum Pengembangan Metodologi

Penelitian ini bertujuan untuk mengembangkan sebuah NER *tagger* untuk *tweet* berbahasa Indonesia[1] dengan menggunakan pendekatan *machine learning*. Dalam pendekatan *machine learning*, penulis melakukan ekstraksi fitur dengan menggunakan tiga teknik yaitu *Word Embedding*, *Word Shape*, dan POS TAG.

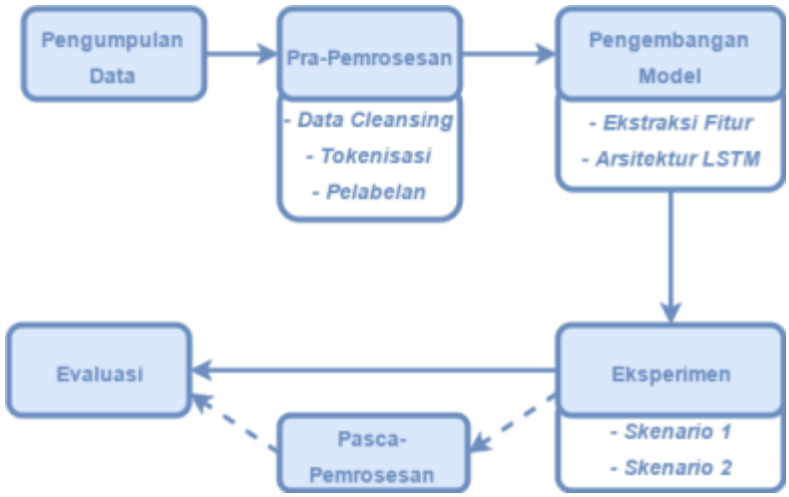
Penelitian ini menggunakan empat buah korpus, yaitu korpus *training* data yang tersedia di SceLe, data yang berasal dari Ibu Ika Alfina, data yang berasal dari Wikipedia, dan data *tweet* dari Twitter yang penulis kumpulkan dengan *crawling tweet*. Selanjutnya penulis melakukan pra-pemrosesan pada data *tweet* yang dikumpulkan dengan cara *crawling tweet*.

Setelah itu dalam pengembangan model, penulis menggunakan ekstraksi fitur dan arsitektur LSTM. Seperti yang telah disebutkan di atas, ekstraksi fitur menggunakan tiga teknik. Model untuk teknik *Word Embedding* dibangun dengan menggunakan empat korpus yang telah disebutkan sebelumnya sedangkan model untuk teknik *Word Shape* dan POS TAG dibangun dengan menggunakan korpus *training* data yang tersedia di SceLe .

Setelah pengembangan model, penulis melakukan eksperimen yang terdiri dari dua skenario. Pada skenario pertama penulis akan membandingkan beberapa kombinasi teknik ekstraksi fitur pada *testing* data korpus *training* data yang berasal dari SceLe. Pada skenario kedua penulis akan menggunakan kombinasi ekstraksi fitur yang terbaik dari hasil skenario satu dan melakukan eksperimen dengan dua jenis *testing* data yang akan dijelaskan lebih lanjut pada subbab selanjutnya.

Untuk file output berupa format IOB, selanjutnya akan dilakukan evaluasi parsial pada hasil yang didapatkan dari skenario satu dan dua dengan menghitung nilai *precision*, *recall*, dan *F1-score*. Penulis menggunakan *5-fold cross validation* untuk mendapatkan rata-rata *precision*, *recall*, dan *F1-score* dari setiap eksperimen dengan cara semua data menjadi 5 bagian, 4 bagian menjadi *training* data dan 1 bagian menjadi *testing* data. Proses tersebut lakukan secara berulang sebanyak lima kali sampai masing-masing bagian menjadi *testing* data.

Untuk evaluasi yang dilakukan pada sistem yang disediakan oleh Bapak Rahmad Mahendra, penulis mengubah file output yang berformat IOB ke dalam bentuk *Named Entity* dengan melakukan tahap pasca-pemrosesan. File output yang telah berbentuk *Named Entity* akan di unggah ke sistem untuk dilakukan *testing*.



Gambar 2.1: Diagram Gambaran Umum

1.2 Pengumpulan Data

Pengumpulan data bertujuan untuk mendapatkan *training* data dan *testing* data yang dapat digunakan dalam pembentukan model (*training*) dan evaluasi model NER (*testing*). Pada penelitian ini, penulis memiliki empat buah korpus yang penggunaannya sebagai berikut.

2.2.1 Data *Training* dari SceLe dan data dari Bu Ika Alfina

Data *training* dari SceLe dan data dari Bu Ika Alfina merupakan data yang terdapat dalam bentuk *Named Entity*. Dalam prosesnya, data yang telah berbentuk *Named Entity* akan diubah bentuknya menjadi format IOB. Berikut contoh pengubahan data.

Format *Named Entity*	Format IOB
<ENAMEX TYPE=PERSON>Joko Widodo<\ENAMEX> beristirahat di <ENAMEX TYPE=LOCATION>Bogor<\ENAMEX>.	[{“text”:[“Joko”, “Widodo”, “beristirahat”, “di”, “Bogor”, “.”],”label”:[“B-PERSON”, “I-PERSON”, “O”, “O”, “B-LOCATION”, “O”]}]

Tabel 2.2.1: Contoh Perubahan Format *Named Entity* pada suatu Kalimat menjadi Format IOB

Data *training* dari SceLe digunakan dalam pembangunan model untuk *Word Embedding*, *Word Shape*, POS TAG, dan *Context* serta digunakan sebagai data untuk skenario 1. Sedangkan data dari Bu Ika Alfina digunakan dalam membangun model untuk *Word Embedding*.

2.2.2 Data *Tweet* hasil *Crawling Tweet* dari Twitter

Data *tweet* hasil *crawling tweet* dari Twitter ini berbentuk json yang mana setiap

instance merepresentasikan data mengenai suatu *tweet* seperti isi *tweet*, jumlah *likes*, jumlah *reply*, jumlah *retweet* dan lain sebagainya. Data hasil *crawling tweet* ini akan dilakukan *cleaning* pada tahap pra-pemrosesan. Data hasil *crawling tweet* ini akan digunakan dalam mengembangan model *Word Embedding*.

2.2.3 Data dari Wikipedia.org

Data yang berasal dari Wikipedia ini berbentuk teks yang terdiri dari beberapa buah kalimat. Data ini akan digunakan pada pengembangan model *Word Embedding*.

1.3 Pra-Pemrosesan

Korpus data yang penulis dapatkan dalam tahap pengumpulan data selanjutnya akan dilakukan pra-pemrosesan sebelum nantinya akan menjadi input dalam tahap pengembangan model. Dalam tahap ini, penulis menggunakan teknik *data cleaning*, tokenisasi, dan pelabelan dalam data.

2.3.1 *Data Cleaning*

Teknik *data cleaning* ini digunakan sebagai tahap pra-pemrosesan data *tweet* hasil *crawling tweet* di Twitter. Terdapat tiga tahap *data cleaning* pada data *tweet* hasil *crawling tweet* yaitu tahap hapus *instance*, tahap ekstraksi *tweet*, dan tahap peunikan *tweet*.

Gambar 2.3.1: Diagram Tahapan *Data Cleaning*

Tahap hapus *instance* itu melakukan penghapusan *instance* pada data berbentuk json hasil *crawling tweet* yang mana *instance* tersebut mengandung kata yang sama dengan *query* pada *crawling tweet* sebelum-sebelumnya. Input pada tahap ini berupa hasil *crawling tweet* ke i yang berbentuk json, sedangkan output berbentuk data json yang sudah tidak mengandung kata yang sama dengan *query* pada *crawling tweet* ke 1 sampai i-1. Berikut contoh input dan output untuk tahap ini.

Input	Output
{data *instance* 1}, {data *instance* 2}, {data *instance* 3}, {data *instance* 4}	{data *instance* 1}, {data *instance* 3}, {data *instance* 4}

Tabel 2.3.1.1: Contoh Input dan Output untuk Tahap Hapus *Instance* apabila diketahui {data *instance* 2} Mengandung Kata yang Sama dengan *query* dari *Crawling Tweet* Sebelumnya

File output dari tahap hapus *instance* akan dimasukkan ke dalam tahap ekstraksi *tweet*. Seperti yang telah dijelaskan pada subbab sebelumnya, tahap ini akan mengekstrak kumpulan *tweet* dari kumpulan *instance* (informasi mengenai suatu *tweet*). Berikut contoh input dan output untuk tahap ini.

Input	Output
{"coordinates": null, "text": "@FavaRustam Iyaaaa eon, papi tolong disadarin please \ud83d\ude02\ud83d\ude02\ud83d\ude02 mungkin dia lagi khilaf? \ud83d\ude05", "is_quote_status": false, "favorited": false, …..}	@FavaRustam Iyaaaa eon, papi tolong disadarin please mungkin dia lagi khilaf?

Tabel 2.3.1.2: Contoh Input dan Output untuk Tahap Ekstrak *Tweet*

File berupa kumpulan *tweet* yang merupakan output pada tahap tersebut akan menjadi input untuk tahap pengunikan *tweet*. Tahap pengunikan *tweet* ini akan memastikan bahwa setiap *tweet* yang telah diekstraksi dari hasil *crawling tweet* itu unik antara *tweet-tweet* lain dari hasil *crawling tweet* yang sama. Berikut contoh input dan output untuk tahap ini.

Input	Output
@LIGUANLN Jomblo teriak jomblo @IJNU_satu Sesama jomblo masa membenci??? @BADELKIE98 @BADKrystal94 Jomblo aja enak @kimbonakr Ini kenapa pabonya gak pernah hilang sih??!! Kenapa?!! Pantes jomblo terus!!! @IJNU_satu Sesama jomblo masa membenci??? Sepi banget kayak jomblo @imbaeirenex TAH JOMBLO KAN	@LIGUANLN Jomblo teriak jomblo @IJNU_satu Sesama jomblo masa membenci??? @BADELKIE98 @BADKrystal94 Jomblo aja enak @kimbonakr Ini kenapa pabonya gak pernah hilang sih??!! Kenapa?!! Pantes jomblo terus!!! Sepi banget kayak jomblo @imbaeirenex TAH JOMBLO KAN SIA

SIA @BADELKIE98 @BADKrystal94 Jomblo aja enak	

Tabel 2.3.1.2: Contoh Input dan Output untuk Tahap Ekstrak *Tweet*

2.3.2 Tokenisasi

Tokenisasi dilakukan menjadi dua tahap yaitu tokenisasi berdasarkan spasi dan berdasarkan simbol. Tokenisasi berdasarkan spasi yaitu memisahkan kata-kata pada suatu kalimat berdasarkan spasi yang muncul pada kalimat tersebut, sedangkan tokenisasi berdasarkan simbol yaitu simbol-simbol yang tidak terletak ditengah kata akan dipisahkan (berdiri sendiri).

2.3.3 Pelabelan

Pelabelan yang dilakukan yaitu pada data *training* dari SceLe yang berformat *Named Entity* diubah menjadi format IOB yang telah dijelaskan pada subbab sebelumnya.

1.4 Pengembangan Model

Pada subbab ini, penulis akan menjelaskan mengenai perancangan dan pengembangan model yang selanjutnya akan digunakan oleh penulis untuk melakukan evaluasi pada tahap eksperimen. Terdapat dua tahap yang penulis lakukan dalam perancangan dan pengembangan model, yaitu:

2.4.1 Ekstraksi Fitur

Di tahap ini penulis melakukan ekstraksi fitur dari kata-kata kata-kata dalam data. Terdapat empat teknik ekstraksi fitur yang nantinya akan penulis kombinasikan agar mendapatkan evaluasi terbaik dari kombinasi teknik ekstraksi fitur (skenario 1). Berikut empat teknik ekstraksi fitur yang dimaksud.

2.4.1.1 *Word Embedding*

Teknik *Word Embedding* memetakan kata-kata dalam korpus ke dalam bentuk one-hot-vector. Panjang suatu vector dalam merepresentasikan one-hot-vector untuk teknik ini sebanyak jumlah kata yang unik dalam korpus. Berikut contoh input dan output berupa one-hot-vector (dengan asumsi terdapat 10 kata unik dalam korpus).

Teks	*Word Embedding* one-hot-vector
Saya ingin pergi bersama ayah	{[0,0,1,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,1,0], [0,0,0,0,0,1,0,0,0,0], [0,0,0,0,0,0,0,1,0,0],

	[0,0,0,0,0,0,0,0,0,1]}

Tabel 2.4.1.1: Contoh Teks yang diubah Dalam Bentuk One-hot-vector untuk *Word Embedding*

Dari contoh di atas dapat dipahami bahwa setiap kata yang unik direpresentasikan atau dipetakan oleh sebuah one-hot-vector yang unik.

2.4.1.2 POS TAG

POS TAG yang penulis gunakan menggunakan tiga tag yaitu NNP (*Proper Noun*), NN (*Noun*), dan O (*Other*). Input berupa data yang berasal dari SceLe akan diubah ke dalam bentuk POS TAG secara manual oleh penulis (hand anotated).Berikut contoh perubahan format data dari *Named Entity* ke format POS TAG.

Format *Named Entity*	Format POS TAG
<ENAMEX TYPE=PERSON>Windy<\ENAME X> membeli buku di <ENAMEX TYPE=LOCATION>PIM<\ENAME X> <ENAMEX TYPE=PERSON>Aditya<\ENAME X> pergi bersama temannya	[{“text”:[“Windy”, “membeli”, “buku”, “di”, “PIM”], “label”:[“NNP”, ”O”, ”NN”, “O”, “NNP”]}, {“text”:[“Aditya”, “pergi”, “bersama”,”temannya”], “label”:[“NNP”, ”O”, “O”, “NN”]}]

Tabel 2.4.1.2.1: Contoh Perubahan Format *Named Entity* pada suatu Kalimat menjadi Format POS TAG

Dalam implementasi kodenya, format POS TAG belum cukup baik dalam pemrosesan *machine learning*, kemudian penulis mengubah format POS TAG menjadi format POS TAG one-hot-vector. Dalam format POS TAG one-hot vector, masing-masing tag dipetakan ke dalam suatu vektor. Tag NN akan dipetakan dalam [1,0,0], tag NNP akan dipetakan dalam [0,1,0], dan tag O akan dipetakan dalam [0,0,1]. Berikut contoh perubahan format data.

Format POS TAG	Format POS TAG one-hot-vector
[{“text”:[“Windy”, “membeli”, “buku”, “di”, “PIM”], “label”:[“NNP”, ”O”, ”NN”, “O”, “NNP”]}, {“text”:[“Aditya”, “pergi”, “bersama”,”temannya”],	{“0”:[[0,1,0],[0,0,1],[1,0,0],[0,0,1],[0,1,0]], “1”: [[0,1,0],[0,0,1],[0,0,1],[1,0,0]] }

“label”:[“NNP”, ”O”, “O”, “NN”]}]	

Tabel 2.4.1.2.2: Contoh Perubahan Format POS TAG pada suatu Kalimat menjadi Format POS TAG one-hot-vector

2.4.1.3 *Word Shape*

Teknik *Word Shape* yang penulis gunakan ini memerlukan input berupa kalimat yang kata-katanya sudah terpisah satu sama lain, sehingga penulis menggunakan format POS TAG sebagai input dari teknik ini. Teknik *Word Shape* memetakan setiap karakter dalam suatu kata ke dalam bentuk baru. Apabila karakter tersebut berupa huruf alfabet kapital maka dipetakan menjadi ‘X’, sedangkan untuk huruf alfabet non kapital dipetakan menjadi ‘x’, untuk angka dipetakan menjadi ‘d’, dan untuk karakter selain alfabet dan angka maka akan dipetakan ke dirinya sendiri. Karakter akan berhenti dipetakan pada suatu kata apabila terdapat kategori karakter yang berurutan sepanjang 4 karakter, seperti kata “Aditya” akan dipetakan menjadi “Xxxxx” karena karakter “x” telah sepanjang 4 karakter. Berikut contoh perubahan format data.

Format POS TAG	Format *Word Shape*
[{“text”:[“Windy”, “membeli”, “buku”, “di”, “PIM”], “label”:[“NNP”, ”O”, ”NN”, “O”, “NNP”]}, {“text”:[“Aditya”, “pergi”, “bersama”,”temannya”], “label”:[“NNP”, ”O”, “O”, “NN”]}]	{ “0”:[“Xxxxx”, “xxxx”, “xxxx”, “xx”, “XXX”], “1”:[“Xxxxx”, “xxxx”, ”xxxx’, “xxxx”] }

Tabel 2.4.1.3.1: Contoh Perubahan Format POS TAG pada suatu Kalimat menjadi Format *Word Shape*

Berdasarkan format *Word Shape* di atas, angka diawal sebelum list kata-kata dalam bentuk baru merupakan indeks yang merepresentasikan suatu kalimat yang mengandung kata-kata tersebut. Setelah pemetaan setiap karakter dalam suatu kata, maka jenis kata hasil pemetaan yang telah penulis dapat akan dipetakan kedalam suatu *integer*, misal “Xxxxx” dipetakan ke 39. Berikut contoh perubahan format data.

Format *Word Shape*	Format *vector Word Shape*
{“0”:[“Xxxxx”, “xxxx”, “xxxx”, “xx”, “XXX”], “1”:[“Xxxxx”, “xxxx”, ”xxxx”, “xxxx”]}	{ “0”:[39, 7, 7, 18, 2], “1”:[39, 7, 7, 7] }

Tabel 2.4.1.3.2: Contoh Perubahan Format *Word Shape* pada suatu Kalimat menjadi Format *vector Word Shape*

2.4.1.4 *Context*

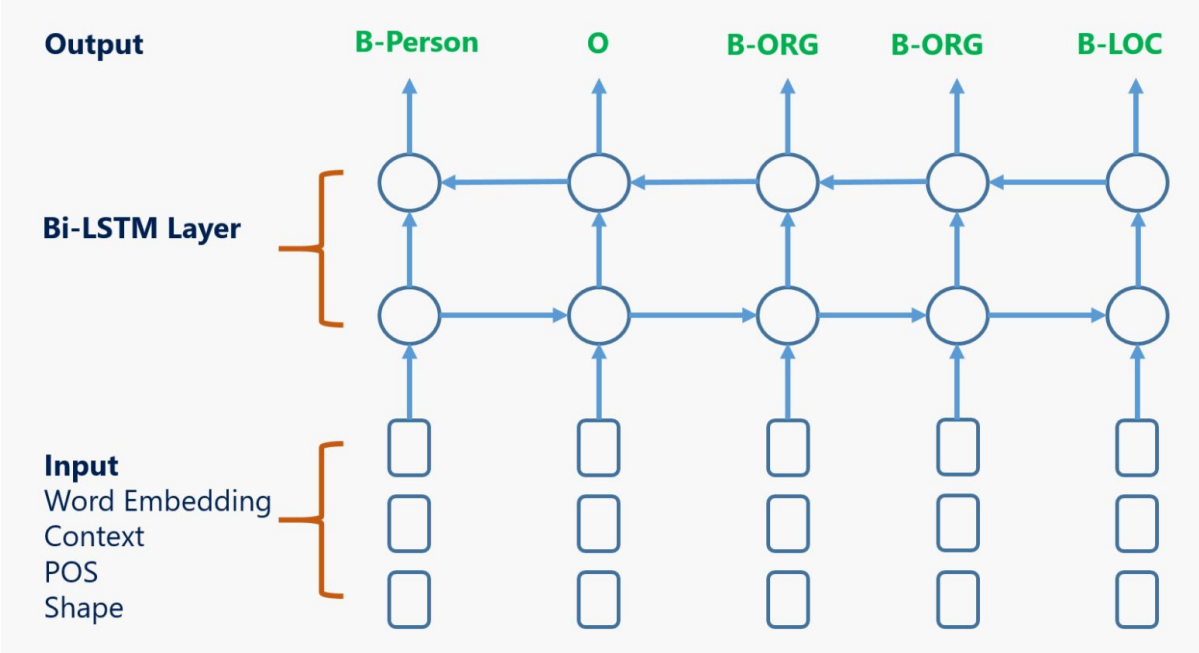
Teknik *Context* merupakan teknik yang memanfaatkan teknik lainnya yaitu *Word Embedding*. Hasil dari teknik *Word Embedding* digunakan sebagai input untuk teknik *Context*. Teknik *Context* ini menggunakan representasi one-hot-vector hasil dari *Word Embedding* untuk mendapatkan one-hot-vector yang merepresentasikan kata di kiri dan di kanan dari suatu kata dalam sebuah kalimat. Berikut contoh input berupa one-hot-vector hasil *Word Embedding* dan output (dengan asumsi terdapat 10 kata unik dalam korpus)

Word Embedding one-hot-vector	*Context* one-hot-vector
{[0,0,1,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,1,0], [0,0,0,0,0,1,0,0,0,0], [0,0,0,0,0,0,0,1,0,0], [0,0,0,0,0,0,0,0,0,1]}	{[[0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,1,0]], [[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,1,0,0,0,0]], [[0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,1,0,0]], [[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,1]], [[0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0]]}

Tabel 2.4.1.1: Contoh One-hot-vector untuk *Word Embedding* diubah dalam Bentuk One-hot-vector untuk *Context*

Dari contoh di atas dapat dipahami bahwa dalam bentuk one-hot-vector untuk *Context* terdiri dari 2 *vector* yaitu *vector* dari kiri yang berada di sebelah kata itu dan *vector* dari kanan yang berada di sebelah kata itu. Apabila kata tersebut terletak di paling awal kalimat maka kata tersebut tidak memiliki kata di sebelah kiri sehingga *vector* untuk kata sebelah kiri merupakan *vector* 0. Sebaliknya apabila kata tersebut terletak di paling akhir kalimat maka kata tersebut tidak memiliki kata di sebelah kanan sehingga *vector* untuk kata sebelah kanan yaitu *vector* 0.

2.4.2 Arsitektur LSTM



1.5 Eksperimen

Dalam melakukan eksperimen, arsitektur *deep learning* yang penulis gunakan adalah Long Short Term Memories. LSTM terbukti baik dalam menyelesaikan *sequence labeling problem*, sebagaimana kita memodelkan problem NER pada penulisan ini. Eksperimen yang penulis lakukan menggunakan dua jenis evaluasi yaitu rata-rata *precision*, *recall*, dan *F1-score* dari setiap iterasi pada *5-cross fold validation* dan sistem evaluasi yang disediakan oleh Bapak Rahmad Mahendra. Eksperimen yang menggunakan *5-cross fold validation* dilakukan dengan awalnya penulis membagi data *training* menjadi 5 bagian kemudian lakukan 5 kali iterasi. Setiap iterasi ke i, bagian data ke i akan menjadi data *testing* dan sisa bagian yang lainnya akan digabungkan kembali dan menjadi data *training*.

Setelah melakukan pembagian data tersebut, penulis membuat model dari data *training* tersebut. Setelah model didapatkan maka penulis melakukan *testing* terhadap masing-masing model dengan sistem evaluasi yang telah disediakan. Selain menggunakan sistem evaluasi, penulis menggunakan *precision*, *recall*, dan *F1-score* pada setiap iterasi di *5-cross fold validation* untuk dihitung rata-rata *precision*, *recall*, dan *F1-score*.

2.5.1 Skenario 1

Pada skenario ini, penulis ingin membandingkan model dari kombinasi beberapa teknik ekstraksi fitur. Data yang digunakan pada skenario ini adalah data *training* dari SceLe. Berikut eksperimen-eksperimen dengan kombinasi teknik ekstraksi fitur yang akan dilakukan pada skenario ini.

2.5.1.1 Menggunakan *Word Embedding*

Pada eksperimen ini, penulis melakukan percobaan dengan model *Word Embedding*. Formatnya sudah dijelaskan pada subbab ekstraksi fitur bagian *Word Embedding*, adapun format modelnya sebagai berikut.

Teks	*Word Embedding* one-hot-vector
Saya ingin pergi bersama ayah	{[0,0,1,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,1,0], [0,0,0,0,0,1,0,0,0,0], [0,0,0,0,0,0,0,1,0,0], [0,0,0,0,0,0,0,0,0,1]}

Tabel 2.5.1.1: Contoh Bentuk One-hot-vector untuk model *Word Embedding*

2.5.1.2 Menggunakan *Word Embedding* dan POS TAG

Pada eksperimen ini, penulis melakukan percobaan dengan model *Word Embedding* digabung dengan model POS TAG. Adapun format model untuk eksperimen ini sebagai berikut.

Teks	*Word Embedding* one-hot-vector	Format POS TAG one-hot-vector	Format one-hot-vector *Word Embedding* dan POS TAG
Saya ingin pergi bersama ayah	{[0,0,1,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,1,0], [0,0,0,0,0,1,0,0,0,0], [0,0,0,0,0,0,0,1,0,0], [0,0,0,0,0,0,0,0,0,1]}	{“0”:[[0,0,1],[0,0,1],[0,0,1],[0,0,1],[1,0,0]] }	{“0”: [[[0,0,1,0,0,0,0,0,0,0], [0,0,1]], [[0,0,0,0,0,0,0,0,1,0],[0,0,1]], [[0,0,0,0,0,1,0,0,0,0],[0,0,1]], [[0,0,0,0,0,0,0,1,0,0],[0,0,1]], [[0,0,0,0,0,0,0,0,0,1],[1,0,0]]]}

Tabel 2.5.1.2: Contoh Format One-hot-vector untuk model *Word Embedding* dengan POS TAG

2.5.1.3 Menggunakan *Word Embedding* dan *Word Shape*

Pada eksperimen ini, penulis melakukan percobaan dengan model *Word Embedding* digabung dengan model *Word Shape*. Adapun format model untuk eksperimen ini sebagai berikut.

Teks	*Word Embedding* one-hot-vector	Format *Word Shape*	Format one-hot-vector *Word Embedding* dan

			Word Shape

Saya ingin pergi bersama ayah	{[0,0,1,0,0,0,0,0,0,0],
	[0,0,0,0,0,0,0,0,1,0],
	[0,0,0,0,0,1,0,0,0,0],
	[0,0,0,0,0,0,0,1,0,0],
	[0,0,0,0,0,0,0,0,0,1]}
		{
		"0":[31, 7, 7, 7, 7]
		}
			{"0":[[[0,0,1,0,0,0,0,0,0,0], 31],
			[[0,0,0,0,0,0,0,0,1,0],7],
			[[0,0,0,0,0,1,0,0,0,0],7],
			[[0,0,0,0,0,0,0,1,0,0],7],
			[[0,0,0,0,0,0,0,0,0,1],7]]}

Tabel 2.5.1.3: Contoh Format One-hot-vector untuk model *Word Embedding* dengan *Word Shape*

2.5.1.4 Menggunakan *Word Embedding* dan *Context*

Pada eksperimen ini, penulis melakukan percobaan dengan model *Word Embedding* digabung dengan model *Context*. Adapun format model untuk eksperimen ini sebagai berikut.

Teks	*Word Embedding* one-hot-vector	Format *Context* one-hot-vector	Format one-hot-vector *Word Embedding* dan *Context*
Saya ingin pergi bersama ayah	{[0,0,1,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,1,0], [0,0,0,0,0,1,0,0,0,0], [0,0,0,0,0,0,0,1,0,0], [0,0,0,0,0,0,0,0,0,1]}	{[[0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,1,0]], [[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,1,0,0,0,0]], [[0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,1]], [[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,1]], [[0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,0]]}	{[[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0]],[0,0,0,0,0,0,0,0,1,0]], [[0,0,0,0,0,0,0,0,1,0],[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,1,0,0,0,0]], [[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,1]], [[0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,1]], [[0,0,0,0,0,0,0,0,0,1][0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0]]}

Tabel 2.5.1.4: Contoh Format One-hot-vector untuk model *Word Embedding* dengan *Context*

2.5.1.5 Menggunakan *Word Embedding*, POS TAG, *Word Shape*, dan *Context*

Pada eksperimen ini, penulis melakukan percobaan dengan model *Word Embedding* digabung dengan model POS TAG, *Word Shape*, dan *Context*.

Adapun format model untuk eksperimen ini sebagai berikut.

Teks	*Word Embedding* one-hot-vector	Format *Context* one-hot-vector	Format POS TAG one-hot-vector	Format *Word Shape*	Format one-hot-vector *Word Embedding* dan *Context*
Saya bersama ayah	{[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,1]}	{[[0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,1,0]],[[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,1]],[[0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0]]}	{"0":[[0,0,1],[0,0,1],[1,0,0]]}	{ "0":[31, 7, 7] }	{"0":[[[0,0,1,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,1,0],[0,0,1],31],[[0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,1],[0,0,1],7],[[0,0,0,0,0,0,0,0,0,1][0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0],[1,0,0],7]]}

Tabel 2.5.1.2: Contoh Format One-hot-vector untuk model *Word Embedding* dengan *Context*

1.6 Pasca-Pemrosesan

Tahap ini hanya dilakukan pada hasil *testing* yang akan di unggah pada sistem evaluasi yang telah disediakan oleh Bapak Rahmad Mahendra.

1.7 Evaluasi

Evaluasi yang penulis gunakan adalah evaluasi parsial untuk cross validation. Pada evaluator yang disediakan oleh Pak Mahendra, evaluasi yang digunakan adalah parsial dan exact match.

BAB 3
IMPLEMENTASI

Pada bab ini penulis akan membahas mengenai implementasi yang penulis gunakan dalam penelitian ini. Penjelasanya telah dijelaskan pada bab 2.

3.1 Pengumpulan Data

Bab ini menjelaskan bagaimana penulis mengumpulkan data

3.1.1 Data *training* dari SceLe

Data training dari SceLe di-download dari SceLe

3.1.2 Data *tweet* hasil *crawling twitter*

```
import tweepy,sys,jsonpickle

consumer_key = '1IKIIYOUkgh909PGfGep6cNBc'
consumer_secret = 'U8uiEeISNlVfzzI3FrD9iPMP46bW4JWjAB2QlynnneOdf6L7q6'

qry= 'warung'
maxTweets = 200000 # Isi sembarang nilai sesuai kebutuhan anda
tweetsPerQry = 100  # Jangan isi lebih dari 100, ndak boleh oleh Twitter
fName='hasil crawl\hc055.json' # Nama File hasil Crawling

auth = tweepy.AppAuthHandler(consumer_key,consumer_secret)
api = tweepy.API(auth,
wait_on_rate_limit=True,wait_on_rate_limit_notify=True)
if (not api):
    sys.exit('Autentikasi gagal, mohon cek "Consumer Key" & "Consumer
Secret" Twitter anda')

sinceId=None;max_id=-1;tweetCount=0
print("Mulai mengunduh maksimum {0} tweets".format(maxTweets))
with open(fName,'w') as f:
    while tweetCount < maxTweets:
        try:
            if (max_id <= 0):
                if (not sinceId):
                    new_tweets=api.search(q=qry,count=tweetsPerQry)
                else:

new_tweets=api.search(q=qry,count=tweetsPerQry,since_id=sinceId)
            else:
                if (not sinceId):

new_tweets=api.search(q=qry,count=tweetsPerQry,max_id=str(max_id - 1))
                else:
```

```
new_tweets=api.search(q=qry,count=tweetsPerQry,max_id=str(max_id -
1),since_id=sinceId)
            if not new_tweets:
                print('Tidak ada lagi Tweet ditemukan dengan
Query="{0}"'.format(qry));break
            for tweet in new_tweets:

f.write(jsonpickle.encode(tweet._json,unpicklable=False)+'\n')
            tweetCount+=len(new_tweets)
            sys.stdout.write("\r");sys.stdout.write("Jumlah Tweets telah
tersimpan: %.0f" %tweetCount);sys.stdout.flush()
            max_id=new_tweets[-1].id
        except tweepy.TweepError as e:
            print("some error : " + str(e));break # Aya error, keluar
print ('\nSelesai! {0} tweets tersimpan di
"{1}"'.format(tweetCount,fName))
```

3.1.3 Data dari Wikipedia.org

____.

3.2 Pra-Pemrosesan

Subbab ini berisi kode dalam tahap pra-pemrosesan.

3.2.1 *Data Cleaning*

Tahap hapus *instance*:

```
def writeListofStringToFile(aList, filename):

    thefile = open(filename, "w")

    for k in aList:
        thefile.write(k.strip())
        thefile.write("\n")

    thefile.close()

    return

namafileInput = "hasil crawl\hc055.json"
fileOutput = "hasil crawl\hc055f.json"

inputFile = open(namafileInput, "r")
flines = inputFile.readlines()
result = []

lines_len = len(flines)

for row in flines:
      if ("#JanganLihatLuarnyaAja" not in row) and ("mantan" not in
row) and ("php" not in row) and ("jalan" not in row) and ("air" not
in row) and ("bersih" not in row) and ("hp" not in row) and
```

```
("samsung" not in row) and ("rumah" not in row) and ("berenang" not
in row) and ("luar negeri" not in row) and ("makan" not in row) and
("bola" not in row) and ("sedih" not in row) and ("kartini" not in
row) and ("#Waisak2017" not in row) and ("mendaki" not in row) and
("pegunungan" not in row) and ("mimpi" not in row) and ("ekspetasi"
not in row) and ("anak" not in row) and ("libur" not in row) and
("konser" not in row) and ("jadwal" not in row) and ("bete" not in
row) and ("nonton" not in row) and ("#diTrans7" not in row) and
("buang" not in row) and ("cinta" not in row) and ("dokter" not in
row) and ("misterius" not in row) and ("gaya" not in row) and
("korea" not in row) and ("capek" not in row) and ("skripsi" not in
row) and ("bosan" not in row) and ("semester" not in row) and
("Kecelakaan" not in row) and ("curang" not in row) and ("sendiri"
not in row) and ("Kicauan" not in row) and ("grup" not in row) and
("labil" not in row) and ("begadang" not in row) and ("jomblo" not
in row) and ("cobaan" not in row) and ("senang" not in row) and
("kerja" not in row) and ("puncak" not in row) and ("kosong" not in
row) and ("senin" not in row) and ("tolong" not in row) and
("jawaban" not in row) and ("menjadi" not in row) and ("bawa" not
in row)  and ("warna" not in row):
                  result.append(row)

print(len(result))
writeListofStringToFile(result, fileOutput)
```

Tahap ekstraksi *tweet*:

```
import json
fileInput = open('hasil crawl\hc055.json', 'r')
fileOutput = open('hasil crawl txt\hc055.txt', 'w')

count = 0
for row in fileInput:
      # Check apakah row tersebut json
      if(len(row) > 4) :
            # load row dalam bentuk json
            rowJson = json.loads(row)
            isiTweet = str(rowJson['text'].encode('ascii',
'ignore')).replace("b\'", "").replace("\\n"," ")
            isiTweet = isiTweet[:-1]
            fileOutput.write(isiTweet)
            fileOutput.write("\n")
            count += 1

print(count)
```

Tahap pengunikan *tweet*:

```
def writeListofStringToFile(aList, filename):

    thefile = open(filename, "w")

    for k in aList:
        thefile.write(k.strip())
```

```
        thefile.write("\n")

    thefile.close()

    return



namafileInput = "hasil crawl txt\hc055.txt"
fileOutput = "hasil crawl txt\hc055f.txt"

inputFile = open(namafileInput, "r")
flines = inputFile.readlines()
result = []

lines_len = len(flines)

for row in flines:
      if row not in result:
            result.append(row)

print(len(result))
writeListofStringToFile(result, fileOutput)
```

3.2.2 Tokenisasi

____.

3.2.3 Pelabelan

____.

3.3 Pengembangan Model

Subbab ini berisi kode dalam tahap pengembangan model.

3.3.1 Ekstraksi Fitur

Sub-subbab ini berisi kode dalam tahap ekstraksi fitur.

3.3.1.1 *Word Embedding + Context*

```
import os.path
import numpy as np
import json
from gensim.models import Word2Vec
import logging
from keras.utils.np_utils import to_categorical
from word_embedding import WordEmbedding

import configparser
Config = configparser.ConfigParser()
Config._interpolation = configparser.ExtendedInterpolation()
Config.read('config.ini')
```

```
scenario = Config.get('general', 'scenario')
project = Config.get(scenario, 'project')
'''Extracting Features'''

logging.basicConfig(
    format='%(asctime)s [%(process)d] [%(levelname)s] %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S',
    level=logging.INFO)


def to_int(sents, dictionary_src):
      int_sents = []
      for sent in sents:
            int_sents.append(dictionary_src[sent])
      return int_sents

def extractData(filename):
      x = []
      y = []
      with open(filename) as f:
            data = json.load(f)
            for sentence in data:
                  x.append(sentence["text"])
                  y.append(sentence["label"])
      return x, y

file1 = './data/' + project + '/data.json'
x_train, y_train = extractData(file1)

we = WordEmbedding(model_name=Config.get(scenario, 'we_model'),
dirpath='word-embedding/model')
x_train_we = [we.transform(sentence) for sentence in x_train]
x_train_context = [we.context_window(sentence, nb_contexts=3) for
sentence in x_train_we] # bikin context window di sini. w1 w2 w3 -->
w1<start>w2  w2w1w3 w3w2<end>

label_dict = {"B-ORGANIZATION": 0,
                  "I-ORGANIZATION": 1,
                  "B-LOCATION": 2,
                  "I-LOCATION": 3,
                  "B-PERSON": 4,
                  "I-PERSON": 5,
                  "O": 6
                  }


# Transform label into integer based on label_dict
y_train_int = [to_int(labels, label_dict) for labels in y_train]
y_train_keras = [to_categorical(yy, len(label_dict)).tolist() for yy in
y_train_int]

y_train_out = y_train_keras

dict_x_train = {}
for i, sentence in enumerate(x_train_we):
      dict_x_train[i] = sentence
```

```
dict_x_context = {}
for i, sentence in enumerate(x_train_context):
      dict_x_context[i] = sentence

dict_y_train = {}
for i, sentence in enumerate(y_train_out):
      dict_y_train[i] = sentence

outfile_x_train = open('./features/' + project + '/x_train.json', 'w')
outfile_x_train.write(json.dumps(dict_x_train))
outfile_x_train.close()

outfile_x_context = open('./features/' + project +
'/x_train_context.json', 'w')
outfile_x_context.write(json.dumps(dict_x_context))
outfile_x_context.close()

outfile_y_train = open('./features/' + project + '/y_train.json', 'w')
outfile_y_train.write(json.dumps(dict_y_train))
outfile_y_train.close()
```

3.3.1.2 POS TAG

```
import json
from pprint import pprint

fileOutput = 'twitter_POS_vector.json'
count = 0

def pos_tag(tag):
      if(tag == "NN"):
            shapeAkhir = '[1,0,0]'
      elif tag == "NNP":
            shapeAkhir = '[0,1,0]'
      else:
            shapeAkhir = '[0,0,1]'
      return shapeAkhir

inputFile = 'twitter_POS.json'
with open(inputFile) as data_file:
      data = json.load(data_file)

splitText = []
wordText = ""
wordLabel = ""
shape = ""
label = ""
dictShape = {}
kataKe = 0
hasil = '{'
for i in range(0, len(data)):
      splitText = data[i]['text']

      hasil += '"' + str(kataKe) + '": ['
      for j in range(0, len(splitText)):
            wordText = data[i]['text'][j]
```

```
            wordLabel = data[i]['label'][j]
            label = pos_tag(wordLabel)
            hasil +=  str(label) + ','
      hasil = hasil[:len(hasil)-1] + '],'
      kataKe += 1
hasil = hasil[:len(hasil)-1]
hasil += '}'
thefile = open(fileOutput, "w")
thefile.write(hasil)
thefile.close()
```

3.3.1.3 *Word Shape*

```
import json
from pprint import pprint

fileOutput = 'twitter_wordS.json'
dictOutput = "dict_training1.json"

count = 0

def word_shape(word, dictS):
      dictOut = {}
      if len(word) >= 100:
            return '__LONG__'
      shape, last, shape_char, seq = [], '', '', 0
      for c in word:
            if c.isalpha():
                  if c.isupper():
                        shape_char = 'X'
                  else:
                        shape_char = 'x'
            elif c.isdigit():
                  shape_char = 'd'
            else:
                  shape_char = c


            if shape_char is last:
                  seq += 1
            else:
                  seq = 0
                  last = shape_char


            if seq < 4:
                  shape.append(shape_char)
      shapeAkhir = ''.join(shape)
      #print(shapeAkhir in dictS.keys())
      if(shapeAkhir in dictS.keys()):
            dictOut = dictS
      else:
            if dictS == {}:
                  dictS["<UNKNOWN>"] = 0
                  dictS[shapeAkhir] = 1
                  dictOut = dictS
```

```
            else:
                  #print("MASUK")
                  inverse = [(value, key) for key, value in
dictS.items()]
                  maxVal = int(max(inverse)[0])
                  dictS[shapeAkhir] = maxVal + 1
                  dictOut = dictS

      return shapeAkhir, dictOut

inputFile = 'tw.json'
with open(inputFile) as data_file:
      data = json.load(data_file)

splitText = []
word = ""
shape = ""
label = ""
dictShape = {}
kataKe = 0
hasil = '{'
for i in range(0, len(data)):
      splitText = data[i]['text']

      hasil += '"'+ str(kataKe) + '": ['
      for j in range(0, len(splitText)):
            word = data[i]['text'][j]
            (shape,dictShape) = word_shape(word, dictShape)
            #print(dictShape)
            label = dictShape[shape]
            hasil +=  str(label) + ','
      hasil = hasil[:len(hasil)-1] + '],'
      kataKe += 1
hasil = hasil[:len(hasil)-1]
hasil += '}'
print(len(dictShape))

thefile = open(fileOutput, "w")
thefile.write(hasil)
thefile.close()

dictWS =  json.dumps([{k : v} for k,v in dictShape.items()], indent=4)
thefile = open(dictOutput, "w")
thefile.write(dictWS)
thefile.close()
```

3.3.2 Arsitektur LSTM

```
def create_model(label_dict, timesteps, features, lstm_depth=1,
kernel_size=3, filters=128, strides=1):
    input_layer = Input(shape=(timesteps, features))
    #masked_layer = Masking(mask_value=0.)(input_layer)
    cnn_layer = Conv1D(filters=filters, kernel_size=kernel_size,
padding='same',
                       activation='relu', strides=strides)(input_layer)
    forward_layer = LSTM(units=128, return_sequences=True)(cnn_layer)
```

```
    backward_layer = LSTM(units=128, return_sequences=True,
go_backwards=True)(cnn_layer)
    merged_layer = merge([forward_layer, backward_layer], mode='concat',
concat_axis=-1)
    dropout_layer = Dropout(0.2)(merged_layer)

    output_layer = TimeDistributed(Dense(units=len(label_dict),
activation='softmax'))(dropout_layer)

    model = Model(inputs=[input_layer], outputs=[output_layer])
    print 'compiling model..'
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'],
                  sample_weight_mode='temporal')

    print model.summary()
    return model
```

3.4 Eksperimen

Subbab ini berisi kode dalam tahap eksperimen.

3.4.1 Skenario 1

Sub-subbab ini berisi kode dalam skenario 1.

3.5 Evaluasi

```
import numpy as np

#predicted = [[0,1,2,1,0,3,0,3],[0,1,2,0,0,2,0,0,3,4]]

#expected = [[0,1,1,0,3,3,3,4],[0,1,2,0,0,0,0,0,3,4]]


def to_tagged_dict(array, tag_b, tag_i):
	result = {}
	index = -1
	for i, sentence in enumerate(array):
		kode = i
		status = 0
		for j, tag in enumerate(sentence):
			if tag == tag_b:
				status = 1
				index+=1
				kode_new = (kode, j)
				result[index] = [kode_new]
			elif tag == tag_i:
				if status == 1:
					kode_new = kode, j
					result[index].append(kode_new)
				else:
					status = 1
					index+=1
					kode_new = (kode, j)
					result[index] = [kode_new]
			else:
				status = 0

	return result

def intersection(list_a, list_b):
	return [val for val in list_a if val in list_b]

def count_confussion(dict_predicted, dict_expected):
	flag_predicted = [False] * len(dict_predicted)
	flag_expected = [False] * len(dict_expected)

	tp = 0
	for i in dict_expected:
		item_expected = dict_expected[i]
		for j in dict_predicted:
			item_predicted = dict_predicted[j]
			if intersection(item_expected, item_predicted) and (not
flag_expected[i]) and (not flag_predicted[j]):
				tp+=1
```

```
                                flag_predicted[j] = True
                                flag_expected[i] = True
        fp = flag_predicted.count(False)
        fn = flag_expected.count(False)
        return tp, fp, fn

def evaluate(list_predicted, list_expected, tag_a, tag_b):
        dict_predicted = to_tagged_dict(list_predicted, tag_a, tag_b)
        dict_expected = to_tagged_dict(list_expected, tag_a, tag_b)
        tp, fp, fn = count_confussion(dict_predicted, dict_expected)
        precision = tp * 1.0 / (tp + fp)
        recall = tp * 1.0 / (tp + fn)
        f1 = (2.0 * precision * recall) / (precision + recall)
        return precision, recall, f1
```

BAB 4
EKSPERIMEN DAN ANALISIS

_____.

4.1 Skenario 1

4.1.1 Menggunakan *Word Embedding*

Eksperimen ini dilakukan dengan menggunakan fitur Word Embedding dengan menggunakan metode 5-folds cross validation. Rasion antara data training dan testing adalah 80 : 20..

Tabel 4.1.1 Hasil Eksperimen dengan menggunakan *Word Embedding*

	Precision	***Recall***	***F-Measure***
ORGANIZATION	0,5264	0,51418	0,5184
PERSON	0,6654	0,5754	0,6151
LOCATION	0,7721	0,7867	0,5184
TOTAL	0,6546	0,6254	0,5506

4.1.2 Menggunakan *Word Embedding* dan Context

Menggunakan fitur Word Embedding dengan melihat context window 1 kata sebelum dan 1 kata sesudah..

Tabel 4.1.2 Hasil Eksperimen dengan menggunakan *Word Embedding* dan context

	Precision	***Recall***	***F-Measure***
ORGANIZATION	0.5414872484	0.469237514	0.5000898881
PERSON	0.6840694732	0.5824703101	0.6260746724
LOCATION	0.7566572558	0.7306726177	0.7369568316
TOTAL	0,6607	0,5941	0,6209

4.1.3 Menggunakan *Word Embedding* dan *Word POS TAG*

Fitur yang digunakan adalah Word Embedding dengan POS TAG, dimana tag yang kami jadikan fokus adalah NN, NNP dan O.

Tabel 4.1.3 Hasil Eksperimen dengan menggunakan *Word Embedding* dan *POS TAG*

	Precision	***Recall***	***F-Measure***
ORGANIZATION	0.6719259582	0.6814543945	0.6746323672
PERSON	0.8444843084	0.814197779	0.8259152417
LOCATION	0.8333767144	0.8011997123	0.8118124172
TOTAL			

4.1.4 Menggunakan *Word Embedding*, POS TAG, dan *Word Shape*

_____.

Tabel 4.1.4 Hasil Eksperimen dengan menggunakan *Word Embedding*, POS TAG, dan *Word Shape*

	Precision	***Recall***	***F-Measure***
ORGANIZATION			
PERSON			
LOCATION			
TOTAL			

4.1 Skenario 1

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kami

5.2 Saran

_____.

DAFTAR PUSTAKA

FORMAT
[1] NAMA BELAKANG, INISIAL DEPAN. (TAHUN, BULAN TANGGAL). *JUDUL*. Diambil kembali dari <NAMA WEB>: <URL>

[1] Mahendra, R. (April 2017). *Deskripsi Tugas NER Twitter NLP Genap 2017*. Diambil kembali dari SCELE Fasilkom UI: https://scele.cs.ui.ac.id/pluginfile.php/20321/mod_resource/content/1/Deskripsi%20Tugas%20NER%20Twitter%20NLP%20Genap%202017.pdf

[2] Augenstein, I., Derczynski, L., Bontcheva, K. (1 Januari 2017) *Generalisation in named entity recognition: A quantitative analysis*. Diambil kembali dari Science Direct: http://remote-lib.ui.ac.id:2057/science/article/pii/S088523081630002X

[3] Wongso, R. Meiliana. Suhartono, D. (19 Oktober 2016). *A Literature Review of Question Answering Syste using Named Entity Recognition*. Diambil kembali dari IEEEXplore: http://remote-lib.ui.ac.id:2087/stamp/stamp.jsp?arnumber=7892454

[4] Kominfo. (7 November 2013). *Kominfo: Pengguna Internet di Indonesia 3 Juta Orang*. Diambil kembali dari Kominfo.go.id: https://www.kominfo.go.id/content/detail/3415/kominfo-pengguna-internet-di-indonesia-63-juta-orang/0/berita_satker

[5] I. Alfina, R. Manurung dan M. I. Fanany, "DBpedia entities expansion in automatically building dataset for Indonesian NER," *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Malang, 2016, pp. 335-340. Diambil kembali dari IEEEXplore: http://remote-lib.ui.ac.id:2087/stamp/stamp.jsp?arnumber=7872784

[6]