LAPORAN HASIL
RISET KOMPETITIF MAHASISWA

PENGGUNAAN DATA MINING UNTUK TRANSLITERASI
BAHASA ARAB KE BAHASA INDONESIA

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2021

LAPORAN HASIL
RISET KOMPETITIF MAHASISWA

PENGGUNAAN DATA MINING UNTUK TRANSLITERASI BAHASA ARAB KE BAHASA INDONESIA

OLEH
ALVIAN BURHANUDDIN / 19841009
RIZQI AMALIYA / 19841008
AHMAD LATIF QOSIM / 19841007

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG

2021

LEMBAR PENGESAHAN
LAPORAN HASIL RISET KOMPETITIF MAHASISWA

Judul	: Penggunaan data mining untuk Transliterasi bahasa arab ke bahasa Indonesia
Nama Ketua Peneliti	: Alvian Burhanuddin NIM 19841009
Nama Anggota Peneliti	: Rizqi Amaliya NIM 19841008
	Ahmad Latif Qosim NIM 19841008
Jurusan	: Teknik Informatika
Lama Kegiatan	: 6 Bulan
Biaya	: Rp. 5.779.761,-

Malang, Mei 2021
Ketua Peneliti

Alvian Burhanuddin
19841009

Mengetahui,	Menyetujui
Ketua Jurusan	Dosen Pembimbing
Dr. Cahyo Crysdian	Dr. Muhammad Faisal, S.Kom, M.T
NIP. 19740424 200901 1 008	NIP. 197405102005011007

Mengetahui,
Wakil dekan bidang kemahasiswaan dan kerjasama

Dwi Suheriyanto, M.P
NIP. 19740325 200312 1 001

LEMBAR PERNYATAAN ORISINALITAS RISET

Dengan Ini,

Nama Ketua Peneliti	: Alvian Burhanuddin NIM 11650049
Jurusan	: Teknik Informatika
Angkatan Tahun/Semester	: 2020/3

Menyatakan bahwa Riset yang berjudul:

Penggunaan data mining untuk Transliterasi bahasa arab ke bahasa Indonesia, Merupakan karya riset yang dapat dipertanggung jawabkan orisinalitasnya. Apabila di kemudian hari ditemukan kecurangan maka saya bersedia riset ini dibatalkan, mengembalikan dana bantuan penghargaan riset dan menerima sanksi yang telah ditetapkan.

Malang, Mei 2021

Alvian Burhanuddin
19841009

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Berdasarkan sensus yang dilakukan oleh badan pusat statistik, hampir 87 persen penduduk indonesia memeluk agama islam[1]. Hampir keseluruhan dari aktifitas keagamaan umat islam menggunakan bahasa arab. Ironisnya, berdasarkan survey yang dilakukan oleh IIQ mencatat bahwa 65% warga muslim buta pada alquran[1][2]. Melihat hal ini, maka salah satu solusi adalah melakukan transliterasi pada bahasa arab menjadi bahasa indonesia.

Adanya proses transliterasi ini merupakan suatu hal yang harusnya disegerakan. Mengingat kebutuhan ini merupakan hal sebagai perantara dalam menjalankan ibadah dalam islam itu sendiri. Mencakup dari dzikir, wirid, dan kegiatan keagamaan lainnya. Namun, hendaknya ketika sudah ada transliterasi ini tidak kemudian belajar sendiri tanpa ada yang mengarahkan. Karena Terdapat beberapa bagian dari transliterasi tidak dapat sesuai dengan lafad atau suara asli bacaan dari bahasa arab.

Al-Faruqi[2], mencatat tujuan utama dari transliterasi bahasa arab ini bukan transliterasi itu sendiri, melainkan mempermudah untuk mempelajarinya saja. Dan Jika transliterasi tersebut adalah alquran, maka transliterasi tersebut tidak bisa dikatakan sebagai alquran. Selain itu ketika melihat pada surat yusuf ayat 2, yarng artinya adalah *"Sesungguhnya telah kami turunkan al-Qur'an dalam bahasa Arab, mudah-mudahan kamu memikirkannya."*.

Dalam perkembangannya, masalah utama transliterasi manual bahasa arab ke bahasa indonesia adalah banyaknya referensi[3][4]. Hal ini menyebabkan inkonsistensi antara transliterasi satu dengan lainnya. Pada penulisan huruf خ yang memiliki beberapa penulisan, diantaranya kh, ḳ, dan k̲h. Hal ini menjadikan pembaca kesulitan memahami beberapa ejaan tersebut.

Sehingga melihat urgensi dan masalah tersebut, peneliti ingin melakukan pendekatan data mining untuk melakukan transliterasi bahasa arab ke bahasa indonesia. Hal ini memiliki tujuan untuk menemukan model terbaik dalam melakukan transliterasi. Selain itu, hal ini akan memiliki manfaat ke banyak pihak baik kepada civitas akademik maupun kepada masyarakat.

[1] https://www.jpnn.com/news/65-persen-muslim-buta-al-quran

[2] https://www.republika.co.id/berita/p2oodi396/65-persen-masyarakat-indonesia-buta-huruf-alquran

1.2 Pertanyaan Penelitian

Berdasarkan paparan latar belakang yang telah disampaikan. Penulis merumuskan pertanyaan penelitian yang nantinya akan terjawab lewat penelitian ini. Yakni:

1. Mengukur efektifitas penggunaan *machine translation* dalam transliterasi bahasa arab ke bahasa indonesia.
2. Menemukan model terbaik dari *word alignment*, *phrase based machine translation*, dan *neuro machine translation*.

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengetahui efektifitas penggunaan *machine translation* dalam transliterasi bahasa arab ke bahasa indonesia.
2. menemukan model terbaik dari beberapa pilihan metode *machine translation*.

1.4 Manfaat Penelitian

Secara umum, pada penelitian ini terdapat 2 manfaat yang didapatkan yakni untuk akademisi, praktisi, dan masyarakat. Manfaat tersebut antara lain:

1. Manfaat Untuk akademisi
    a. Sebagai referensi akademisi dalam melakukan pengembangan model bahasa indonesia berbasis mesin.
    b. Dapat dijadikan perbandingan pada penelitian selanjutnya, untuk dilakukan penyempurnaan dari sisi limitasi.
2. Manfaat untuk praktisi
    a. Mengurangi kesulitan dalam melakukan tagging data berdasarkan inkonsistensi penulisan ejaan.
3. Manfaat untuk masyarakat
    a. Kemudahan dalam melakukan pembelajaran karena seluruh huruf menggunakan bahasa yang seragam. Sehingga tidak terlalu sulit ketika berpindah dari satu buku ke buku lainnya.

BAB II
KAJIAN PUSTAKA

2.1 Transliterasi

Transliterasi diartikan sebagai penyalinan atau penggantian huruf dari abjad yang satu ke abjad lainnya[5]. Keputusan menteri tahun 1987[6], menerjemahkan transliterasi sebagai pengalih hurufan dari abjad yang satu ke abjad yang lain beserta perangkatnya. Sehingga bisa disimpulkan bahwa transliterasi merupakan sebuah cara untuk mengubah huruf dari abjad satu ke abjad lainnya dengan hal yang berkaitan dengan huruf tersebut. Keputusan menteri tersebut, mencakup beberapa hal yang dirumuskan yakni:

1. Konsonan
2. Vokal
3. Maddah
4. Ta'marbutah
5. Syaddah
6. Kata Sandang
7. Hamzah
8. Penulisan Kata
9. Huruf Kapital
10. Tajwid

Tabel utama sebagai dasar transliterasi dalam peraturan pemerintah[6] dapat ditemukan pada tabel 2.1. Penelitian lanjutan menemukan banyak transliterasi lain bermunculan seiring dengan adaptasi dari masyarakat.

Table 2.1. Tabel transliterasi dalam bahasa indonesia

Huruf Arab	Nama	Huruf Latin	Nama
ا	alif	tidak dilambangkan	tidak dilambangkan
ب	ba	b	be
ت	ta	t	te
ث	sa	ṡ	es

Table 2.1

Huruf Arab	Nama	Huruf Latin	Nama
ج	jim	j	je
ح	ha	ḥ	ha
خ	kha	kh	ka dan ha
د	dal	d	de
ذ	zal	ż	zet
ر	ra	r	er
ز	zai	z	zet
س	sin	s	es
ش	syin	sy	es dan ye
ص	ṣad	ṣ	es
ض	ḍad	ḍ	de
ط	ṭa	ṭ	te
ظ	ẓa	ẓ	zet
ع	’ain	’̲	koma terbalik diatas
غ	gain	g	ge
ف	fa	f	ef
ق	qaf	q	qi
ك	kaf	k	ka
ل	lam	l	el
م	mim	m	em
ن	nun	n	en
و	wau	w	we
ه	ha	ha	ha
ء	hamzah	‘̲	apostrof
ي	ya	y	ye

Temuan transliterasi lain tersebut diungkapkan Ahmad[3] kendati adanya transliterasi bahasa arab ke bahasa indonesia dapat membantu masyarakat yang belum bisa membaca bahasa arab. Utamanya dalam masalah manasik haji. Banyak beredarnya berbagai model translasi di lapangan, memunculkan ketidakteraturan antara satu pedoman transliterasi dengan pedoman transliterasi lain. Hal ini membuat makin bingungnya masyarakat dalam membaca transliterasi tersebut. Beberapa saran untuk perbaikan pada transliterasi telah diberikan. diantaranya adalah memberikan transliterasi yang mengakomodir seperti pada bunyi asli bukan pada karakter. Namun, saran tersebut tentu saja akan membutuhkan pada peninjauan ulang pada peraturan transliterasi saat ini.

2.2 Penelitian Terdahulu

Guellil[7] melakukan transliterasi dari *arabizi* ke bahasa arab menggunakan *machine translation*. *Arabizi* merupakan salah satu bahasa latin yang merupakan transliterasi dari bahasa arab menggunakan dialek aljazair. Penelitian ini, menggunakan dataset dari beberapa media sosial seperti Facebook, Twitter, dan Youtube. *Statistical machine translation* dan *neural machine translation* menghasilkan masing masing akurasi 73% dan 75% pada internal data. Sedangkan pada eksternal data didapati hasil akurasi hanya 45%. Dari penelitian tersebut, ketika dibandingkan *neural machine translation*, mampu meningkatkan akurasi hingga 2%.

Ameur[8] juga melakukan penelitian dengan menggunakan transliterasi untuk bahasa arab menjadi bahasa inggris. Penelitian ini menggunakan *neuro machine translation*, dengan penggunaan pada RNN. Pada penelitian ini lakukan beberapa komparasi pada algoritma machine translation. Dihasilkan *bidirectional attention-based encoder-decoder* memperoleh nilai tertinggi dibandingkan pad metode lainnya. Untuk transliterasi dari bahasa inggris ke bahasa arab didapatkan hasil *word error rate* hingga 5.40 dan *character error rate* mencapai 0.95. Sedangkan untuk bahasa arab ke bahasa inggris diperoleh, nilai *word error rate* 65.16 dan *character error rate* 16.35. Hal yang cukup menarik dari penelitian tersebut, *phrase based machine translation* memperoleh nilai terbaik kedua, dibandingkan dengan algoritma neural network yang lebih modern.

Masmoudi[9] menggunakan *syntatical rules* untuk melakukan pembuatan dataset. Dari dataset tersebut, kemudian dilakukan transliterasi menggunakan CRF. Penelitian yang dilakukan pada dialek tunisia ini menghasilkan skor *word error rate* sebanyak 9.80 dan *character error rate* 10.47. Dibandingkan beberapa metode sebelumnya, pendekatan hybrid ini mampu menghasilkan banyak dataset dalam sebuah inisialisasi penelitian. Sehingga, meskipun penelitian memiliki nilai hasil skor

cukup rendah, namun penelitian ini termasuk salah satu penelitian yang baik dalam proses *unsupervised learning*.

Table 2.2. Perbedaan dan Persamaan dengan penelitian sebelumnya

Nama dan tahun penelitian	Judul penelitian	Persamaan penelitian	Perbedaan penelitian	Orisinalitas penelitian
Guellil, Imane and Azouaou, Faiçal and Abbas, Mourad and Fatiha, Sadat. 2017	Arabizi transliteration of Algerian Arabic dialect into modern standard Arabic	• Penggunaan *Statistical machine translation* dan *neural machine translation* dalam proses data mining	• penelitian tidak mencakup pada karakter. • dataset yang digunakan, pada penggunaan dialek bahasa indonesia.	perbedaan dialek antara bahasa indonesia dengan dialek aljazair akan menjadikan model yang sangat berbeda. Selain itu, perbedaan pada *learning* pada karakter tentu akan berbeda dengan penggunaan kata atau frasa.
Ameur, Mohamed Seghir Hadj and Meziane, Farid and Guessoum, Ahmed. 2017	Arabic machine transliteration using an attention-based encoder-decoder model	• Penggunaan *neural machine translation* sebagai media untuk transliterasi. • penggunaan kata atau frasa dalam melakukan proses transliterasi	• Perbedaan dalam sub algoritma *neural machine translation*, kami menggunakan pendekatan LSTM. • dataset yang digunakan, pada penggunaan dialek bahasa indonesia.	Perbedaan sub algoritma yang berbeda dalam penggunaan transliterasi berbeda dengan penelitian yang telah ada sebelumnya. Selain itu fokus kami ada pada *low-resource language*

BAB III
METODE PENELITIAN

3.1 Jenis dan Pendekatan Penelitian

Dalam melakukan penelitian terdapat 2 metode yakni pendekatan kualitatif dan kuantitatif. Penelitian kali ini mengarah ke penelitian kualitatif. Dimana terdapat Kriteria penelitian kualitatif telah dituliskan sebelumnya oleh creswell[10], dapat terpenuhi dalam penelitian ini.

Selain itu, penelitian ini memiliki langkah yang sama seperti diutarakan oleh Srivastava[11]. Penelitian diawali dengan gagasan dilanjutkan dengan *research question*. Selanjutnya, diteruskan dengan melakukan pengumpulan data dan literatur terkait. Selain itu penelitian ini juga bersifat dinamis, dimana perubahan selalu diterima untuk penelitian yang lebih baik.

3.2 Sumber Data Penelitian

Sumber data pada penelitian ini mengacu kepada data yang digunakan peneliti dalam melakukan percobaan di penelitian laboratorium. Terdapat 2 macam Data primer yang kami gunakan adalah data hasil crawling dari website dan artefak elektronik. Artefak elektronik ini mencakup buku elektronik, atau foto dari buku artikel. Data sekunder merupakan data pelengkap yang digunakan untuk melengkapi data primer. Dalam penelitian ini, kami menggunakan data sekunder berupa transliterasi bahasa arab ke bahasa indonesia[12] sebagai salah satu rujukan.

3.3 Teknik Pengumpulan Data

Penelitian yang akan dilakukan pada tesis ini beberapa metode dalam pengumpulan data, yakni:

1. Penelitian Lapangan

   Pada penelitian lapangan ini, terdapat 2 model yang digunakan peneliti untuk mendapatkan data. Yakni dengan melakukan crawling website dan ekstraksi content artefak elektronik. Dalam melakukan crawling website, terdapat beberapa kategori yang kami tentukan untuk meningkatkan kredibilitas. Diantaranya adalah:

   (a) Kesesuaian keseluruhan konten dalam website.

   (b) Kejelasan penyedia layanan website, hal ini kami ukur berdasarkan ketersediaan beberapa halaman. Semisal laman tentang, contact, dan layanan

privasi.

Selanjutnya, pada bagian dokumen. Kami membatasi pada penerbit cetakan dari kementrian agama, sehingga diharapkan kredibilitas dari konten dari artefak online tersebut telah teruji. Kedua cara mendapatkan data tersebut, dilakukan secara terkomputasi, mencakup:

a. *Crawling* data, Pada tahap ini peneliti melakukan pengambilan konten dari beberapa website. Keseluruhan konten dari website kami ambil, kecuali bagian dari *header*, *sidebar*, dan *footer*. Data yang diambil ini nantinya akan di proses kembali pada tahapan selanjutnya.

b. *Cleaning* data, Hasil dari *crawling* biasanya masih berupa data yang kurang bersih, karena masih memuat bagian bagian yang tidak di perlukan. Pada tahap ini kami melakukan filter pada data sehingga didapatkan kolom berupa

   - *arabic text* merupakan kalimat dalam bahasa arab.
   - *transliteration text* merupakan transliterasi kalimat arab tersebut pada bahasa indonesia.

   Setelah melalui tahap *cleansing*, peniliti melakukan anotasi manual untuk melakukan pengecekan kesesuaian dari konten yang telah didapat. Pengecekan ini meliputi pengecekan harakat dan kesesuaian tanda baca.

2. Penelitian Laboratorium

   Pada penelitian laboratorium, kami melakukan uji empiris pada teori yang ada kepada data yang kami dapatkan selanjutnya. Uji empiris ini meliputi *Training data*, *Model Fitting*, dan *Performance Scoring*.

3.4 Prosedur Penelitian

Prosedur penelitian untuk melakukan uji empiris dapat di temukan pada gambar 3.1. Pada diagram tersebut dimulai dari tahap pengumpulan dataset hingga proses prediksi.

3.4.1 Dataset

Dataset transliterasi kedalam bahasa indonesia, merupakan salah satu dataset yang jarang ditemukan. Untuk menangani hal tersebut, kami menggunakan kalimat sehari hari. Dataset dalam penelitian ini terdiri dari 1000 kalimat, dengan kurang lebih 5000 kata. Selanjutnya, dataset tersebut kami lakukan pelabelan secara manual.

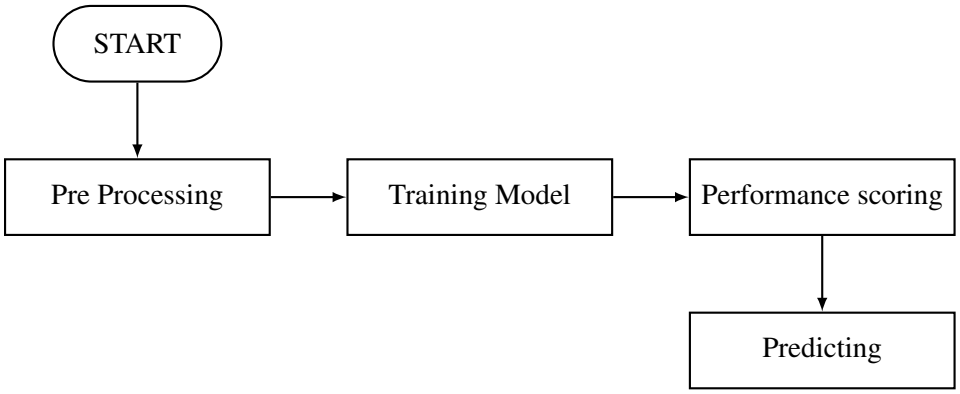


Figure 3.1. Diagram Prosedur Penelitian

3.4.2 Data preprocessing

Pada diagram tersebut diawali dengan proses *preprocessing*. Pada tahapan *preprocessing*, tahapan pembersihan data dilakukan. Mencakup penghilangan bagian yang tidak diperlukan, seperti arti dari bahasa arab, tanda kutip yang tidak diperlukan semacam "(",")", dan angka. Peneliti hanya mengambil teks arab dan transliterasinya.

Ketiadaan dataset pada transliterasi bahasa arab ke bahasa indonesia, menambah salah satu tugas kami untuk membangunnya. Hal ini kami mulai dari melakukan pengecekan manual terhadap data kalimat arab dan transliterasi. Pengecekan ini mencakup pada adanya harakat, dan kesesuaian harakat tersebut. Selanjutnya dilanjutkan dengan melakukan *normalization*.

Pada bagian *normalization*, peneliti melakukan pengecekan pada kemungkinan adanya paragraf dalam data *arabic text*. Ketika ditemukan, maka kalimat tersebut dipisah sesuai dengan titik. Jika tidak ada tanda baca tersedia, maka dipisah berdasarkan makna. Untuk menjaga kredibilitas dari data, kami melakukan bagian ini dengan cara manual.

3.4.3 *Model training*

Dari data yang telah dibersihkan tersebut. *arabic text* dan *transliteration text* dibentuk menjadi token. Token ini digunakan dalam proses *Training model*. Untuk memperoleh hasil terbaik, peneliti membandingkan 3 metode *machine translation*. Yakni dengan menggunakan *dictionary method*, *Phrase-based Method*, dan *Neural network*. Oleh peneliti, setiap data dilatih dengan model tersebut.

3.4.3.1 *Machine Translation*

Lopez[13] menyatakan *Machine translation* dapat didefinisikan sebagai penerjemahan otomatis yang dilakukan oleh komputer dari satu bahasa ke bahasa lain. Metode *machine learning* digunakan Untuk mencapai automasi penerjemahan tersebut. Dimana diterapkan serangkaian algoritma pembelajaran pada mesin untuk mempelajari kumpulan data teks terjemahan atau korpus. Mesin tersebut kemudian mampu untuk memperkirakan data teks lain untuk diterjemahkan berdasarkan nilai dari algoritma tersebut.

Secara teknis, tugas utama dari *machine translation* ialah mengubah rangkaian token dari *source language* dengan *vocabulary* V_S ke dalam rangkaian token *target language* dengan *vocabulary* V_T. Dengan asumsi token dalam rangkaian tersebut adalah kata, sedangkan rangkaiannya adalah kalimat.

Terdapat tiga tahapan dalam penerapan *machine translation*[13]:

1. Tahapan untuk mendeskripsikan bagaimana pengubahan bahasa akan dilakukan. Beberapa algoritma untuk melakukan pengubahan bahasa asal menjadi bahasa lain telah ada dalam penelitian sebelumnya. Diantaranya adalah *word alignment*[14], *Phrase based Machine Translation*[15], dan *Neuro Machine Translation*[16]
2. *Parameterization*. Proses ini merupakan proses untuk melakukan penilaian pada tiap data. Penilaian ini tidak terlepas dari kaidah *machine learning*, yang mana menggunakan serangkaian model statistik dalam pengukurannya.
3. *Decoding*. Proses ini berkenaan dengan input yang diberikan oleh user. Ketika user menginputkan sebuah kalimat, maka komputer akan melakukan proses pencarian pada nilai tertinggi sesuai dengan data pada model. Akurasi dan kecepatan merupakan pembahasan umum pada tahapan ini.

Pembahasan terkait langkah tersebut akan peneliti jelaskan di bagian selanjutnya, utamanya dibagian algoritma. Sesuai dengan deskripsi peneliti pada bagian batasan masalah, bagian yang kami bahas adalah algoritma untuk translasi. Pada bagian *parameterization* dan *decoding*, kami menggunakan penilaian BLEU dan *greedy decoding*.

3.4.3.2 *Phrase Based Machine Translation*

Model *Phrase based Machine Translation* didasarkan pada *noisy channel*. Sehingga dapat dituliskan dalam persamaan:

$$t_{best} = argmax_t\{Pr(t|s)\} \tag{3-1}$$

Koehn[17] memformulasikan ulang persamaan 3-2 menggunakan *bayes rule*. Dimana persamaan:

$$argmax_t p(t|s) = argmax_t p(t|s)p(s) \tag{3-2}$$

Sehingga, dari persamaan tersebut menyebabkan model dapat dihitung secara terpisah, yakni pada probabilitas translasi model$p(t|s)$ dan probabilitas dari model bahasa sumber sendiri $p(t)$.

Pada dasarnya persamaan tersebut digunakan untuk melakukan translasi atau pengubahan dari *sequence* kata yang ada pada kalimat asal s. $s_1...s_j$. Untuk dilakukan translasi pada kalimat target t. $t_1...tj$. Dengan tujuan mencari nilai terbaik dari kandidat kata yang ada pada kalimat. Kandidat kata terbaik yang dimaksud dihasilkan dari proses translasi tersebut, pada beberapa kasus mengalami perubahan urutan. Hal ini dipengaruhi oleh nilai probabilitas antara *sequence* kata dari keseluruhan data yang dijadikan model training. Pengaplikasian *naive rules* ini salah satunya adalah lewat *framework* moses[15], yang juga digunakan pada penelitian ini.

Dalam *Phrase Based Machine Translation*, terdapat setidaknya tiga pendekatan untuk melakukan ekstraksi frasa dalam kalimat[17]. Yakni dengan menggunakan frasa dari *Word Alignment*, frasa dari aturan sintaksis, dan frasa dari *phrase alignments* atau yang biasa disebut sebagai n-grams.

3.4.3.3 *Word Alignment*

Word Alignment dicetuskan pertama kali oleh brown[18]. Ide ini menuliskan secara sederhana bahwa terjemah dari satu kalimat pada kalimat lain digambarkan dengan pasangan kata. Dalam sebuah penerjemahan, urutan kata kata tersebut disamaratakan. Misalkan saja dalam penerjemahan bahasa perancis ke bahasa inggris, *(Le programme a étémis en application | And the(1) program(2) has(3) been(4) implemented(5,6,7))*. Penerjemahan tersebut menggambarkan keseluruhan kata dari bahasa inggris disesuaikan dengan bahasa prancis. Kata dari bahasa perancis tersebut dihubungkan ke dalam bahasa inggris dengan mengabaikan susunannya. Melain-

kan hanya dengan menggunakan padanan kata yang ada di kamus.

Dalam penelitian selanjutnya[14], brown menemukan beberapa kata mungkin tidak akan sesuai karena perbedaan dari susunan tiap bahasa. Untuk mengatasi hal tersebut, brown melakukan pengukuran kemungkinan pada kesamaan kata yang sering muncul. Dari penelitian tersebut, didapatkan bahwa *likelihood* terhadap kata memberikan hasil baik pada dataset penerjemahan dari bahasa inggris ke perancis.

Seiring penelitian yang terus berkembang beberapa penyempurnaan terhadap *word alignment* terus dilakukan. Li[19], dalam penelitian tersebut, menemukan bahwa *word alignment* mengungguli beberapa algoritma lainnya pada beberapa dataset spesifik. Hal ini ditunjukan dengan eksperimen lewat penerapan metode *Prediction Difference* yang menggunakan pendekatan *deterministic*[20] dan sampling[21]

3.4.3.4 *Neuro Machine Translation*

Neuro Machine Translation adalah salah satu metode baru dalam pengembangan teknologi translasi. Metode ini menggunakan serangkaian kalimat sebagai input dalam proses training. Selanjutnya dari input tersebut dijadikan acuan untuk kalkulasi prediksi output. Dalam perancangannya, NMT terdiri dari 2 bagian utama, Encoder, yang melakukan perhitungan representasi nilai kata pada tiap input. Dan decoder, yang menemukan pasangan pasangan kata yang ada pada input dengan melakukan perhitungan probabilitas. Dalam melakukan perhitungan probabilitas tersebut, Arsitektur RNN yang digunakan oleh kalchbrenner[22] menggunakan persamaan.

$$logp(y|x) = \sum_{j=1}^{m} logp(y_j|x_{<j}, s) \tag{3-3}$$

Sedangkan untuk encoder, penelitian tersebut menggunakan convolutional neural network. Disisi lain, bahdanau[23] menggunakan pendekatan *Gate Recurrent unit* (GRU), sebuah pengembangan yang berbeda dari metode umum RNN pada encoder dan decodernya. Penggunaan ini salah satunya didasarkan efektifitas GRU yang lebih baik dibandingkan dengan metode tradisional lainnya[24].

Dalam proses untuk melakukan training dengan model attention, peneliti melakukan encoding pada kalimat input menggunakan GRU. Hasil dari encoding GRU yang berupa vektor dan *hidden states*, kemudian akan dilakukan perhitungan untuk mendapatkan *context vector* c_t yang memiliki persamaan

$$c_t = \sum_s \alpha_{ts} \bar{h}_s \tag{3-4}$$

Dalam persamaan tersebut, α_{ts} merupakan *attention weights* dari sebuah kata yang ada pada sequence kalimat. Nilai ini diperoleh dari persamaan:

$$\alpha_{ts} = \frac{exp(score(h_t, \bar{h}_s))}{\sum_{s'=1}^{S} exp(score(h_t, \bar{h}_s))} \tag{3-5}$$

$score(h_t, \bar{h}_s)$ merupakan kalkulasi dari nilai *decoder hidden state* sebelumnya dengan nilai tiap encoder hidden state pada masing masing token. Persamaan untuk melakukan kalkulasi nilai ini dituliskan dengan

$$score(h_t, \bar{h}_s) = v_\alpha^T tanh(W_a[h_t; \bar{h}_s]) \tag{3-6}$$

Dari nilai kalkulasi tersebut, maka nilai dari *context vector* c_t dihitung berdasarkan dari mean keseluruhan nilai *hidden states*.

3.4.4 *Accuracy measurement*

Model yang dihasilkan dari proses training, oleh kemudian divalidasi dan ditest. Hal ini dilakukan untuk mengetahui hasil terbaik dari penggunaan tiga metode diatas. Dalam memilih variabel nilai, kami menggunakan metrik BLEU[25]. Pada metrik BLEU dilakukan perbandingan *n-grams* dari kandidat dengan referensi. Nilai BLEU didapatkan dari banyaknya kesesuaian kata dalam target dengan referensi.

BAB IV
PEMBAHASAN

Pada bagian ini peneliti menguraikan hasil percobaan sesuai dengan paparan metode yang ada di bagian sebelumnya. Hal ini dilakukan untuk mengetahui akurasi *machine translation* dalam melakukan transliterasi dengan menggunakan model *Phrase based machine learning*, dan *neural network*.

Sebelum dilakukan training pada model, kami melakukan penyelarasan antara dataset kalimat yang akan di transliterasi dengan kalimat transliterasi dalam bahasa indonesia. Dalam penyelarasan ini kami menggunakan MGIZA [26]. Dari penyelarasan tersebut, sebuah kalimat dipisahkan berdasarkan kata. Hasil pasangan dari kata ini akan dijadikan sebagai model training untuk ditentukan nilai bobotnya. Koehn[17] menuliskan terdapat tiga hal yang mempengaruhi nilai bobot pada kata tersebut, yakni kesesuaian heuristik antara frasa kata pada sumber dan target, jumlah frasa dalam sebuah kalimat, dan kesesuaian kata dengan sintaksis kalimat.

Berdasarkan hal tersebut, kami melakukan training dengan menggunakan susunan 1-gram, 2-gram, dan 3-gram pada PBSMT. Dari 3 metode training data yang dilakukan, kami menemukan bahwa model terbaik dihasilkan oleh 3-gram dengan nilai BLEU hingga 24.

Hasil yang didapatkan dari komparasi metode tersebut ditunjukan pada tabel dibawah. Pada tabel tersebut, peneliti memberikan contoh data testing yang terdiri dari *ground truth*, beserta hasil transliterasi dari tiap model.

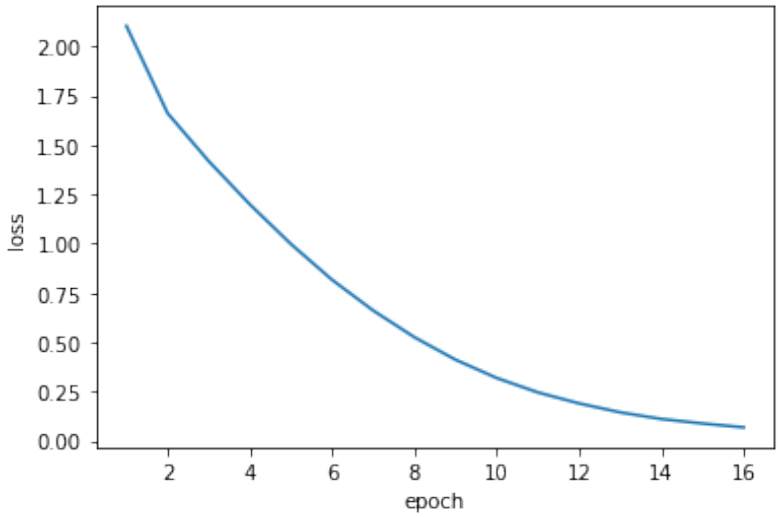
Table 4.1. Hasil uji coba

Teks Asli	من بين ايديهم
ground truth	mim bayna aydihim
Statistical based model	mim bayna aydihim .
attention based model	mim bayna razaqna

Dengan menggunakan dataset yang sama, kami melakukan training pada model berbasis *neural network attention based* dengan pendekatan milik bahdanau. Dalam proses training ini, kami melakukan beberapa pengubahan pada epoch dan batch yang digunakan, terdiri dari 8, 16, dan 24. Model terbaik dihasilkan lewat train dengan menggunakan epoch sebanyak 16. *Attention based* pada penelitian ini memiliki nilai bleu paling rendah dibanding dengan PBSMT. Untuk menemukan titik cerah pada kondisi ini, peneliti melakukan pengamatan manual pada model.

Penurunan nilai bleu ini ditimbulkan karena banyaknya pengubahan susunan kata pada kalimat. Hal ini karena *attention based* tidak melakukan transliterasi sesuai dengan kata pada kalimat.

Figure 4.1. Loss measurement



Sehingga dari hasil percobaan ini kami menemukan model dengan hasil BLEU score tertinggi pada penelitian ini adalah pendekatan dengan *Phrase based machine transliteration* dengan nilai 24. Meskipun begitu, pada percobaan yang kami lakukan menggunakan model ini, terdapat kata yang mengalami perubahan dari nilai dasarnya. Sehingga, salah satu untuk meningkatkan dari nilai tersebut adalah bisa menggunakan lebih banyak dataset. Diharapkan, adanya peningkatan pada akurasi yang didapat.

BAB V
KESIMPULAN

Dalam makalah ini, peneliti telah melakukan eksplorasi pada penggunaan *machine learning* dalam proses transliterasi dari bahasa arab ke bahasa indonesia. Peneliti membandingkan beberapa metode *machine learning* yang telah ada sebelumnya dan telah umum digunakan dalam pengolahan *text mining*. Model tersebut adalah *Dictionary based*, *attention based*, dan *statistical machine based*. Dari ketiga model tersebut, hasil terbaik didapatkan pada penggunaan *statistical based*.

Penelitian ini tidak lepas dari kekurangan, misalkan saja pada bagian susunan kata yang belum sesuai dengan *ground truth*. Selain itu, perlu ditinjau kembali pada eksplorasi penggunaan harakat, yang mana hal tersebut sangat berpengaruh pada proses transliterasi pada bahasa arab ke bahasa indonesia maupun bahasa lainnya.

BIBLIOGRAPHY

[1] S. Indonesia, "badan pusat statistik", *BPS-Statistics Indonesia*, 2018.

[2] I. R. Al-Faruqi, *Toward Islamic English*, 3. International Institute of Islamic Thought (IIIT), 1986.

[3] N. F. Ahmad, "Problematika Transliterasi Aksara Arab-Latin: Studi Kasus Buku Panduan Manasik Haji dan Umrah", *Nusa: Jurnal Ilmu Bahasa dan Sastra*, vol. 12, no. 1, pp. 126–136, 2017.

[4] M. Musadad, "Alquran transliterasi latin dan problematikanya dalam masyarakat muslim denpasar", *SUHUF*, vol. 10, no. 1, pp. 193–209, 2017.

[5] T. R. K. B. Indonesia, "Kamus Bahasa Indonesia", *Jakarta: Pusat Bahasa Departemen Pendidikan Nasional*, vol. 725, 2008.

[6] K. B. M. Agama, M. Pendidikan, and K. R. Indonesia, *Nomor 158 Tahun 1987 dan Nomor 0543 b*, Jakarta, 1987.

[7] I. Guellil, F. Azouaou, M. Abbas, and S. Fatiha, "Arabizi transliteration of Algerian Arabic dialect into modern standard Arabic", in *Social MT 2017/First workshop on social media and user generated content machine translation*, 2017.

[8] M. S. H. Ameur, F. Meziane, and A. Guessoum, "Arabic machine transliteration using an attention-based encoder-decoder model", *Procedia Computer Science*, vol. 117, pp. 287–297, 2017.

[9] A. Masmoudi, M. E. Khmekhem, M. Khrouf, and L. H. Belguith, "Transliteration of arabizi into arabic script for tunisian dialect", *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 19, no. 2, pp. 1–21, 2019.

[10] J. W. Creswell, W. E. Hanson, V. L. Clark Plano, and A. Morales, "Qualitative research designs: Selection and implementation", *The counseling psychologist*, vol. 35, no. 2, pp. 236–264, 2007.

[11] A. Srivastava and S. B. Thomson, "Framework analysis: a qualitative methodology for applied policy research", 2009.

[12] E. Burhanuddin, A. G. Ruskhan, and R. Chrismanto, *Penelitian kosakata bahasa Arab dalam bahasa Indonesia*. Pusat Pembinaan dan Pengembangan Bahasa, Departemen Pendidikan dan Kebudayaan, 1993.

[13] A. Lopez, "Statistical machine translation", *ACM Computing Surveys (CSUR)*, vol. 40, no. 3, pp. 1–49, 2008.

[14] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation", *Computational linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[15] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, *et al.*, "Moses: Open source toolkit for statistical machine translation", in *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, 2007, pp. 177–180.

[16] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation", *preprint arXiv:1701.02810*, 2017.

[17] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation", UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, Tech. Rep., 2003.

[18] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation", *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[19] X. Li, G. Li, L. Liu, M. Meng, and S. Shi, "On the word alignment from neural machine translation", in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1293–1303.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis", *arXiv:1702.04595*, 2017.

[22] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models", in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1700–1709.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling”, *arXiv preprint arXiv:1412.3555*, 2014.

[25] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation”, in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[26] Q. Gao and S. Vogel, “Parallel implementations of word alignment tool”, in *Software engineering, testing, and quality assurance for natural language processing*, 2008, pp. 49–57.

LAMPIRAN

L.1 Data

transliteration_statistical

October 10, 2021

[ ]:
```
import logging
import os
import subprocess
from pathlib import Path

from .util import read_json_file

try:
    import wandb
except:
    pass
from typing import Union

MOSES_IMPLZ = 'moses/moses/bin/lmplz'
MOSES_BUILD_BINARY = 'moses/moses/bin/build_binary'
MOSES_TRAIN_PERL = 'moses/moses/scripts/training/train-model.perl'
EXTERNAL_BIN = 'moses/training-tools'
MOSES_BIN_MOSES = 'moses/moses/bin/moses'
logger = logging.getLogger("moses-rerun")
MOSES_DETOKENIZER = "moses/moses/scripts/tokenizer/detokenizer.perl"
```

[ ]:
```
class MosesSMTModel:

    def __init__(self, config_file:str, use_wandb=True):
        self.config_file = config_file
        self.hparams = read_json_file(config_file)
        self._smooth_variable()
        self.use_wandb = use_wandb
        if self.use_wandb:
            self._init_wandb(self.hparams['wandb_project_run'],
                             self.hparams['wandb_notes'],
                             self.hparams['wandb_name'],
                             self.hparams['wandb_tags'])

    def _init_wandb(self, wandb_project_run, wandb_notes, wandb_name, tags):
        config_wandb = dict(
            default_config=self.config_file,
```

```
            hparams=self.hparams
        )

        self.run = wandb.init(
            project=wandb_project_run,
            notes=wandb_notes,
            name=wandb_name,
            tags=tags,
            config=config_wandb,
            reinit=True
        )

    def _smooth_variable(self):
        self.root_data_pth = Path(self.hparams['data_dir'])
        self.root_output_folder = Path(self.hparams['output_working_dir'])
        self.spv_train_pth = self.root_output_folder / 'train-supervised'
        self.spv_eval = self.root_output_folder / 'eval-supervised'
        self.predicted_file = f"predict{self.hparams['target_file_type']}"
        if "semi-supervised-batch-data" in self.hparams:
            self.semi_supervised_batch_data = self.
↪hparams['semi-supervised-batch-data']
            self.is_semi_supervised = True

    def fit(self, train_data_path: str, out_train_dir: str, absolute_lm_path:␣
↪str):
        try:
            os.makedirs(out_train_dir)
        except FileExistsError:
            logger.warning(out_train_dir + " is exist")
        logger.info("Training time :).. Progressing...")
        subprocess.run([f"nice {MOSES_TRAIN_PERL} \
                        -root-dir { out_train_dir } \
                        -corpus { train_data_path } \
                        -f {self.hparams['source_file_type'].strip('.')} \
                        -e {self.hparams['target_file_type'].strip('.')} \
                        -alignment {self.hparams['moses_args']['alignment']} \
                        -reordering {self.hparams['moses_args']['reordering']} \
                        -lm 0:{self.hparams['moses_args']['moses_ngram']}:{␣
↪absolute_lm_path } \
                        -core {self.hparams['moses_args']['core_cpu']} \
                        -external-bin-dir {EXTERNAL_BIN} --mgiza"
                        ], capture_output=True, shell=True)
        logger.info("Finish Training")

    def predict(self, model_path: str, src_path: str, out_file_dir: str,␣
↪use_custom_file_name=False,
                custom_file_name = 'meong.guk'):
```

```
        try:
            os.makedirs(out_file_dir)
        except FileExistsError:
            logger.warning(out_file_dir + " is exist")
        out_predict = Path(out_file_dir) / self.predicted_file if not␣
↪use_custom_file_name else \
            Path(out_file_dir) / custom_file_name
        subprocess.run(
            [f"{MOSES_BIN_MOSES} -f {model_path} < {src_path} >␣
↪{out_predict}"], shell=True)

    def eval_bleu_moses(self, ref_file: str, evaluation_dir: str, sys_file:␣
↪str):
        import sacrebleu
        try:
            os.makedirs(evaluation_dir)
        except FileExistsError:
            logger.warning(evaluation_dir + " is exist")
        subprocess.run([f"cat {ref_file} | {MOSES_DETOKENIZER} -l en >␣
↪{evaluation_dir}/ref.txt"], shell=True)
        subprocess.run([f"cat {sys_file} | {MOSES_DETOKENIZER} -l en >␣
↪{evaluation_dir}/sys.txt"], shell=True)
        with open(f"{evaluation_dir}/ref.txt",'r+') as file:
            refs = [file.read().split('\n')]
        with open(f"{evaluation_dir}/sys.txt",'r+') as file:
            sys = file.read().split('\n')
        bleu = sacrebleu.corpus_bleu(sys, refs)
        return bleu.score

    def run_experiments(self):
        lm_path = self.root_output_folder / 'lm'
        train_path = self.root_output_folder / 'train'
        logger.info("PERPARE KEN-LM".center(10, '='))
        binary_out = self.prepare_lm(lm_path, Path(self.root_data_pth))
        root_train_data_pth = self.root_data_pth / 'train'
        logger.info("FIT MOSES".center(10, '='))
        self.fit(str(root_train_data_pth),
                 str(train_path),
                 os.path.abspath(str(binary_out)))
        root_test_inf = self.root_data_pth / (self.hparams['data_test'] + self.
↪hparams['source_file_type'])
        dir_out_pred = self.root_output_folder / 'evaluation'
        logger.info("PREDICT MOSES".center(10, '='))
        self.predict(str(train_path / 'model/moses.ini'), str(root_test_inf),␣
↪str(dir_out_pred))
        logger.info("CALCULATE BLEU".center(10, '='))
```

```
        bleu = self.eval_bleu_moses(str(self.root_data_pth / 'test.for'),
                                    str(dir_out_pred),
                                    str(dir_out_pred / self.predicted_file))
        if self.use_wandb:
            self.run.summary['bleu_score'] = bleu
        logger.info(f"BLEU IS {bleu}".center(10, '='))

    def _makedir(self, new_dirs: Union[str, Path]):
        try:
            os.makedirs(new_dirs)
        except FileExistsError:
            logger.warning(str(new_dirs) + " is already exist")

    def run_moses_experiment(self, output_folder: Path, data_path: Path):
        lm_path = output_folder / 'lm'
        train_path = output_folder / 'train'
        logger.info("PERPARE KEN-LM".center(10, '='))
        binary_out = self.prepare_lm(lm_path, Path(data_path))
        root_train_data_pth = data_path / self.hparams['data_train']
        logger.info("FIT MOSES".center(10, '='))
        self.fit(str(root_train_data_pth),
                 str(train_path),
                 os.path.abspath(str(binary_out)))
        root_test_inf = data_path / (self.hparams['data_test'] + self.
↪hparams['source_file_type'])
        dir_out_pred = output_folder / 'evaluation'
        logger.info("PREDICT MOSES".center(10, '='))
        self.predict(str(train_path / 'model/moses.ini'), str(root_test_inf),␣
↪str(dir_out_pred))
        logger.info("CALCULATE BLEU".center(10, '='))
        bleu = self.eval_bleu_moses(str(data_path / 'test.for'),
                                    str(dir_out_pred),
                                    str(dir_out_pred / self.predicted_file))
        return bleu, train_path, dir_out_pred


    def prepare_lm(self, out_lm_dir: Path, train_data_dir: Path) -> str:
        arpa_out = out_lm_dir / ('blm/data.arpa' + self.
↪hparams['target_file_type'])
        binary_out = out_lm_dir / ('blm/data.blm' + self.
↪hparams['target_file_type'])
        import os
        try:
            os.makedirs(str(out_lm_dir / 'blm'))
        except FileExistsError:
            logger.warning(str(out_lm_dir / 'blm') + " is exist")
```

```
        data_train_pth = str(train_data_dir / (self.hparams['data_train'] +␣
↪self.hparams['target_file_type']))

        subprocess.run([f"{MOSES_IMPLZ} -S 20% -o {self.
↪hparams['moses_args']['moses_ngram']}"
                        f" < {data_train_pth} "
                        f" > {arpa_out} "],
                       shell=True)
        subprocess.run([f"{MOSES_BUILD_BINARY} {arpa_out} {binary_out}"],␣
↪shell=True)
        return binary_out
```

[ ]:
```
sup_experiments = 'experiment-config/00001_default_supervised_config.json'
```

[ ]:
```
moses_model = MosesSMTModel(sup_experiments, use_wandb=False)
moses_model.run_experiments()
```

konfigurasi untuk eksperimen berupa json

```
{
    "data_dir" : "data/labelled/",
    "output_working_dir" : "output/supervised/",
    "semi-supervised-count" : 0,
    "moses_args" : {
        "moses_ngram" : 3,
        "core_cpu" : 2,
        "reordering" : "msd-bidirectional-fe",
        "alignment" : "grow-diag-final-and"
    },
    "data_train" : "train",
    "data_development" : "dev",
    "data_test" : "test",
    "source_file_type" : ".inf",
    "target_file_type" : ".for",
    "wandb_project_run": "project-name-here",
    "wandb_notes": "supervised",
    "wandb_name": "supervised",
    "wandb_tags": ["supervised"]
}
```

transliteration_with_attention

October 10, 2021

[1]:
```
#!pip install tensorflow-gpu==2.0.0-alpha0
#!pip install tensorflow==2.0.0-alpha0
```

[2]:
```
from __future__ import absolute_import, division, print_function

# Import TensorFlow >= 1.10 and enable eager execution
import tensorflow as tf

#tf.enable_eager_execution()

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from bidi.algorithm import get_display
import arabic_reshaper
import unicodedata
import re
import numpy as np
import os
import time
import math

print(tf.__version__)
```

```
2.5.0
```

[3]:
```
# Path for the dataset file
path_to_file = "input.txt"
```

[4]:
```
def unicode_to_ascii(s):
    return ''.join(c for c in unicodedata.normalize('NFD', s) if unicodedata.
 ↪category(c) != 'Mn')


def preprocess_sentence(w):
    w = unicode_to_ascii(w.lower().strip())
```

```
    w = re.sub(r"([?.!,¿])", r" \1 ", w)
    w = re.sub(r'[" "]+', " ", w)

    w = re.sub(r"[^a-zA-Z - ?.!,¿]+", " ", w)
    w = w.rstrip().strip()

    w = '<start> %s <end>'%w
    return w
```

[5]:
```
def create_dataset(path, num_examples):
    lines = open(path, encoding='utf-8-sig').read().strip().split('\n')
    word_pairs = [[preprocess_sentence(w) for w in l.split('\t')]  for l in␣
↪lines[:num_examples]]
    print(len(lines))
    print(len(lines[:num_examples]))
    return word_pairs
```

[6]:
```
class LanguageIndex():
    def __init__(self, lang):
        self.lang = lang
        self.word2idx = {}
        self.idx2word = {}
        self.vocab = set()

        self.create_index()

    def create_index(self):
        for phrase in self.lang:
            self.vocab.update(phrase.split(' '))

        self.vocab = sorted(self.vocab)

        self.word2idx['<pad>'] = 0

        for index, word in enumerate(self.vocab):
            self.word2idx[word] = index + 1

        for word, index in self.word2idx.items():
            self.idx2word[index] = word
```

[7]:
```
def max_length(tensor):
    return max(len(t) for t in tensor)


def load_dataset(path, num_examples):
    pairs = create_dataset(path, num_examples)
```

```
    inp_lang = LanguageIndex(sp for en, sp in pairs)
    targ_lang = LanguageIndex(en for en, sp in pairs)


    input_tensor = [[inp_lang.word2idx[s] for s in sp.split(' ')] for en, sp in␣
 ↪pairs]

    target_tensor = [[targ_lang.word2idx[s] for s in en.split(' ')] for en, sp␣
 ↪in pairs]
    max_length_inp, max_length_tar = max_length(input_tensor),␣
 ↪max_length(target_tensor)

    input_tensor = tf.keras.preprocessing.sequence.pad_sequences(input_tensor,
                                                                 ␣
 ↪maxlen=max_length_inp,
                                                                  padding='post')

    target_tensor = tf.keras.preprocessing.sequence.pad_sequences(target_tensor,
                                                                 ␣
 ↪maxlen=max_length_tar,
                                                                 ␣
 ↪padding='post')

    return input_tensor, target_tensor, inp_lang, targ_lang, max_length_inp,␣
 ↪max_length_tar
```

[8]:
```
# Try experimenting with the size of that dataset
num_examples = 10000
# Load the dataset and return Tensor of the input, Tensor for the traget,␣
 ↪Indexed input, Indexed traget, Max length input, Max length target
input_tensor, target_tensor, inp_lang, targ_lang, max_length_inp,␣
 ↪max_length_targ = load_dataset(path_to_file, num_examples)
```

```
24638
10000
```

[9]:
```
# Creating training and validation sets using an 80-20 split
input_tensor_train, input_tensor_val, target_tensor_train, target_tensor_val =␣
 ↪train_test_split(input_tensor, target_tensor, test_size=0.2)

# Show length
len(input_tensor_train), len(target_tensor_train), len(input_tensor_val),␣
 ↪len(target_tensor_val)
```

[9]:
```
(8000, 8000, 2000, 2000)
```

[10]:
```
BUFFER_SIZE = len(input_tensor_train)
BATCH_SIZE = 8
N_BATCH = BUFFER_SIZE//BATCH_SIZE
embedding_dim = 256
units = 300
vocab_inp_size = len(inp_lang.word2idx)
vocab_tar_size = len(targ_lang.word2idx)

dataset = tf.data.Dataset.from_tensor_slices((input_tensor_train,␣
↪target_tensor_train)).shuffle(BUFFER_SIZE)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
```

[11]:
```
def gru(units):
    return tf.keras.layers.GRU(units,
                               return_sequences=True,
                               return_state=True,
                               recurrent_activation='sigmoid',
                               recurrent_initializer='glorot_uniform')
```

Encoder Class

[12]:
```
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.enc_units)

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        return output, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```

Decoder class

[13]:
```
class Decoder(tf.keras.Model):

    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.dec_units)
        self.fc = tf.keras.layers.Dense(vocab_size)
```

```
        # used for attention
        self.W1 = tf.keras.layers.Dense(self.dec_units)
        self.W2 = tf.keras.layers.Dense(self.dec_units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, x, hidden, enc_output):
        hidden_with_time_axis = tf.expand_dims(hidden, 1)
        score = self.V(tf.nn.tanh(self.W1(enc_output) + self.
↪W2(hidden_with_time_axis)))
        attention_weights = tf.nn.softmax(score, axis=1)
        context_vector = attention_weights * enc_output
        context_vector = tf.reduce_sum(context_vector, axis=1)
        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
        output, state = self.gru(x)
        output = tf.reshape(output, (-1, output.shape[2]))
        x = self.fc(output)

        return x, state, attention_weights

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.dec_units))

    def saved_hidden_state(self):
        return self.gru.states
```

[14]:
```
encoder = Encoder(vocab_inp_size, embedding_dim, units, BATCH_SIZE)
decoder = Decoder(vocab_tar_size, embedding_dim, units, BATCH_SIZE)
```

0.1 Define the optimizer and the loss function

[15]:
```
optimizer = tf.keras.optimizers.Adam()


def loss_function(real, pred):
    mask = 1 - np.equal(real, 0)
    loss_ = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=real,␣
↪logits=pred) * mask
    return tf.reduce_mean(loss_)
```

0.2 Save model (Object-based saving)

[16]:
```
save_dir = './model_save'
encoder_prefix = os.path.join(save_dir, "encoder_weights")
decoder_prefix = os.path.join(save_dir, "decoder_weights")
```

[18]:
```
EPOCHS = 16
best_total_loss = 0.0071

for epoch in range(EPOCHS):
    start = time.time()

    hidden = encoder.initialize_hidden_state()
    total_loss = 0

    for (batch, (inp, targ)) in enumerate(dataset):
        loss = 0

        with tf.GradientTape() as tape:
            enc_output, enc_hidden = encoder(inp, hidden)

            dec_hidden = enc_hidden

            dec_input = tf.expand_dims([targ_lang.word2idx['<start>']] *␣
↪BATCH_SIZE, 1)

            # Teacher forcing - feeding the target as the next input
            for t in range(1, targ.shape[1]):
                # passing enc_output to the decoder
                predictions, dec_hidden, _ = decoder(dec_input, dec_hidden,␣
↪enc_output)

                loss += loss_function(targ[:, t], predictions)

                # using teacher forcing
                dec_input = tf.expand_dims(targ[:, t], 1)

        batch_loss = (loss / int(targ.shape[1]))

        total_loss += batch_loss

        variables = encoder.variables + decoder.variables

        gradients = tape.gradient(loss, variables)

        optimizer.apply_gradients(zip(gradients, variables))
```

```
        if batch % 100 == 0:
            print('Epoch {} Batch {} Loss {:.4f}'.format(epoch + 1,
                                                         batch,
                                                         batch_loss.numpy()))

    print('Epoch {} Loss {:.4f}'.format(epoch + 1, total_loss / N_BATCH))

    if best_total_loss > (total_loss / N_BATCH):

        best_total_loss = total_loss

        encoder.save_weights(encoder_prefix)
        decoder.save_weights(decoder_prefix)

        print('Saving weights at epoch {} with Loss {:.4f}'.format(epoch + 1,␣
↪total_loss / N_BATCH))

    print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```

```
Epoch 1 Batch 0 Loss 3.8292
Epoch 1 Batch 100 Loss 2.4311
Epoch 1 Batch 200 Loss 2.3332
Epoch 1 Batch 300 Loss 2.7004
Epoch 1 Batch 400 Loss 2.2621
Epoch 1 Batch 500 Loss 2.6485
Epoch 1 Batch 600 Loss 2.1034
Epoch 1 Batch 700 Loss 1.8764
Epoch 1 Batch 800 Loss 2.0629
Epoch 1 Batch 900 Loss 1.9430
Epoch 1 Loss 2.1032
Time taken for 1 epoch 167.35769724845886 sec

Epoch 2 Batch 0 Loss 1.6459
Epoch 2 Batch 100 Loss 1.2776
Epoch 2 Batch 200 Loss 1.8526
Epoch 2 Batch 300 Loss 1.4784
Epoch 2 Batch 400 Loss 1.7437
Epoch 2 Batch 500 Loss 2.1734
Epoch 2 Batch 600 Loss 1.6157
Epoch 2 Batch 700 Loss 1.5767
Epoch 2 Batch 800 Loss 1.6118
Epoch 2 Batch 900 Loss 1.3642
Epoch 2 Loss 1.6613
Time taken for 1 epoch 170.74187874794006 sec

Epoch 3 Batch 0 Loss 1.3878
Epoch 3 Batch 100 Loss 1.3181
```

```
Epoch 3 Batch 200 Loss 1.3025
Epoch 3 Batch 300 Loss 1.5095
Epoch 3 Batch 400 Loss 1.4300
Epoch 3 Batch 500 Loss 1.5824
Epoch 3 Batch 600 Loss 1.6599
Epoch 3 Batch 700 Loss 1.3756
Epoch 3 Batch 800 Loss 1.3629
Epoch 3 Batch 900 Loss 1.2768
Epoch 3 Loss 1.4181
Time taken for 1 epoch 171.19427514076233 sec

Epoch 4 Batch 0 Loss 1.0119
Epoch 4 Batch 100 Loss 1.3344
Epoch 4 Batch 200 Loss 1.2942
Epoch 4 Batch 300 Loss 1.2126
Epoch 4 Batch 400 Loss 1.1023
Epoch 4 Batch 500 Loss 1.2006
Epoch 4 Batch 600 Loss 1.1799
Epoch 4 Batch 700 Loss 0.9207
Epoch 4 Batch 800 Loss 1.3340
Epoch 4 Batch 900 Loss 1.3081
Epoch 4 Loss 1.1991
Time taken for 1 epoch 171.5133786201477 sec

Epoch 5 Batch 0 Loss 1.2194
Epoch 5 Batch 100 Loss 0.9638
Epoch 5 Batch 200 Loss 0.9456
Epoch 5 Batch 300 Loss 1.1334
Epoch 5 Batch 400 Loss 0.7740
Epoch 5 Batch 500 Loss 1.5060
Epoch 5 Batch 600 Loss 1.0020
Epoch 5 Batch 700 Loss 0.9462
Epoch 5 Batch 800 Loss 0.8262
Epoch 5 Batch 900 Loss 0.8114
Epoch 5 Loss 0.9977
Time taken for 1 epoch 174.11511087417603 sec

Epoch 6 Batch 0 Loss 0.7576
Epoch 6 Batch 100 Loss 0.6871
Epoch 6 Batch 200 Loss 0.7111
Epoch 6 Batch 300 Loss 0.5883
Epoch 6 Batch 400 Loss 0.6637
Epoch 6 Batch 500 Loss 0.9494
Epoch 6 Batch 600 Loss 0.6978
Epoch 6 Batch 700 Loss 0.7529
Epoch 6 Batch 800 Loss 0.7417
Epoch 6 Batch 900 Loss 0.8687
Epoch 6 Loss 0.8166
```

```
Time taken for 1 epoch 172.73238348960876 sec

Epoch 7 Batch 0 Loss 0.3449
Epoch 7 Batch 100 Loss 0.7588
Epoch 7 Batch 200 Loss 0.6575
Epoch 7 Batch 300 Loss 0.6029
Epoch 7 Batch 400 Loss 0.8485
Epoch 7 Batch 500 Loss 0.6065
Epoch 7 Batch 600 Loss 0.5974
Epoch 7 Batch 700 Loss 0.5330
Epoch 7 Batch 800 Loss 0.4942
Epoch 7 Batch 900 Loss 0.5179
Epoch 7 Loss 0.6609
Time taken for 1 epoch 174.34214329719543 sec

Epoch 8 Batch 0 Loss 0.5262
Epoch 8 Batch 100 Loss 0.6588
Epoch 8 Batch 200 Loss 0.4938
Epoch 8 Batch 300 Loss 0.6734
Epoch 8 Batch 400 Loss 0.6879
Epoch 8 Batch 500 Loss 0.4783
Epoch 8 Batch 600 Loss 0.4182
Epoch 8 Batch 700 Loss 0.6679
Epoch 8 Batch 800 Loss 0.5046
Epoch 8 Batch 900 Loss 0.3990
Epoch 8 Loss 0.5258
Time taken for 1 epoch 174.9708857536316 sec

Epoch 9 Batch 0 Loss 0.4108
Epoch 9 Batch 100 Loss 0.4451
Epoch 9 Batch 200 Loss 0.5472
Epoch 9 Batch 300 Loss 0.4824
Epoch 9 Batch 400 Loss 0.3636
Epoch 9 Batch 500 Loss 0.3328
Epoch 9 Batch 600 Loss 0.4902
Epoch 9 Batch 700 Loss 0.5589
Epoch 9 Batch 800 Loss 0.3251
Epoch 9 Batch 900 Loss 0.4653
Epoch 9 Loss 0.4117
Time taken for 1 epoch 174.6313819885254 sec

Epoch 10 Batch 0 Loss 0.2144
Epoch 10 Batch 100 Loss 0.5226
Epoch 10 Batch 200 Loss 0.2864
Epoch 10 Batch 300 Loss 0.2926
Epoch 10 Batch 400 Loss 0.2978
Epoch 10 Batch 500 Loss 0.2073
Epoch 10 Batch 600 Loss 0.3989
```

```
Epoch 10 Batch 700 Loss 0.2055
Epoch 10 Batch 800 Loss 0.3260
Epoch 10 Batch 900 Loss 0.2338
Epoch 10 Loss 0.3192
Time taken for 1 epoch 174.51646971702576 sec

Epoch 11 Batch 0 Loss 0.3327
Epoch 11 Batch 100 Loss 0.1105
Epoch 11 Batch 200 Loss 0.1974
Epoch 11 Batch 300 Loss 0.1537
Epoch 11 Batch 400 Loss 0.3445
Epoch 11 Batch 500 Loss 0.3707
Epoch 11 Batch 600 Loss 0.2339
Epoch 11 Batch 700 Loss 0.1717
Epoch 11 Batch 800 Loss 0.3297
Epoch 11 Batch 900 Loss 0.3024
Epoch 11 Loss 0.2463
Time taken for 1 epoch 174.78922486305237 sec

Epoch 12 Batch 0 Loss 0.1951
Epoch 12 Batch 100 Loss 0.1913
Epoch 12 Batch 200 Loss 0.1870
Epoch 12 Batch 300 Loss 0.1296
Epoch 12 Batch 400 Loss 0.1974
Epoch 12 Batch 500 Loss 0.0976
Epoch 12 Batch 600 Loss 0.2110
Epoch 12 Batch 700 Loss 0.1760
Epoch 12 Batch 800 Loss 0.2806
Epoch 12 Batch 900 Loss 0.1224
Epoch 12 Loss 0.1907
Time taken for 1 epoch 174.82448363304138 sec

Epoch 13 Batch 0 Loss 0.0602
Epoch 13 Batch 100 Loss 0.1334
Epoch 13 Batch 200 Loss 0.1343
Epoch 13 Batch 300 Loss 0.1676
Epoch 13 Batch 400 Loss 0.1948
Epoch 13 Batch 500 Loss 0.0813
Epoch 13 Batch 600 Loss 0.1434
Epoch 13 Batch 700 Loss 0.1424
Epoch 13 Batch 800 Loss 0.0823
Epoch 13 Batch 900 Loss 0.1546
Epoch 13 Loss 0.1454
Time taken for 1 epoch 174.98046278953552 sec

Epoch 14 Batch 0 Loss 0.1150
Epoch 14 Batch 100 Loss 0.0989
Epoch 14 Batch 200 Loss 0.1726
```

```
Epoch 14 Batch 300 Loss 0.1003
Epoch 14 Batch 400 Loss 0.1744
Epoch 14 Batch 500 Loss 0.0730
Epoch 14 Batch 600 Loss 0.0797
Epoch 14 Batch 700 Loss 0.1995
Epoch 14 Batch 800 Loss 0.1625
Epoch 14 Batch 900 Loss 0.1385
Epoch 14 Loss 0.1117
Time taken for 1 epoch 174.85621666908264 sec

Epoch 15 Batch 0 Loss 0.1028
Epoch 15 Batch 100 Loss 0.0538
Epoch 15 Batch 200 Loss 0.0453
Epoch 15 Batch 300 Loss 0.1096
Epoch 15 Batch 400 Loss 0.0402
Epoch 15 Batch 500 Loss 0.0783
Epoch 15 Batch 600 Loss 0.0349
Epoch 15 Batch 700 Loss 0.0874
Epoch 15 Batch 800 Loss 0.1143
Epoch 15 Batch 900 Loss 0.0965
Epoch 15 Loss 0.0889
Time taken for 1 epoch 178.55451107025146 sec

Epoch 16 Batch 0 Loss 0.0244
Epoch 16 Batch 100 Loss 0.1072
Epoch 16 Batch 200 Loss 0.0471
Epoch 16 Batch 300 Loss 0.0426
Epoch 16 Batch 400 Loss 0.0217
Epoch 16 Batch 500 Loss 0.0531
Epoch 16 Batch 600 Loss 0.0657
Epoch 16 Batch 700 Loss 0.1061
Epoch 16 Batch 800 Loss 0.0377
Epoch 16 Batch 900 Loss 0.0913
Epoch 16 Loss 0.0693
Time taken for 1 epoch 181.87508749961853 sec
```

0.3 Transliteration eval

[19]:
```python
def evaluate(sentence, encoder, decoder, inp_lang, targ_lang, max_length_inp,␣
↪max_length_targ):
    attention_plot = np.zeros((max_length_targ, max_length_inp))

    sentence = preprocess_sentence(sentence)

    inputs = [inp_lang.word2idx[i] for i in sentence.split(' ')]
```

```
    inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],␣
↪maxlen=max_length_inp, padding='post')
    inputs = tf.convert_to_tensor(inputs)

    result = ''

    hidden = [tf.zeros((1, units))]
    enc_out, enc_hidden = encoder(inputs, hidden)

    dec_hidden = enc_hidden
    dec_input = tf.expand_dims([targ_lang.word2idx['<start>']], 0)

    for t in range(max_length_targ):
        predictions, dec_hidden, attention_weights = decoder(dec_input,␣
↪dec_hidden, enc_out)

        # storing the attention weigths to plot later on
        attention_weights = tf.reshape(attention_weights, (-1, ))
        attention_plot[t] = attention_weights.numpy()

        predicted_id = tf.argmax(predictions[0]).numpy()

        result += targ_lang.idx2word[predicted_id] + ' '

        if targ_lang.idx2word[predicted_id] == '<end>':
            return result, sentence, attention_plot

        dec_input = tf.expand_dims([predicted_id], 0)

    return result, sentence, attention_plot
```

[20]:
```
# function for plotting the attention weights
def plot_attention(attention, sentence, predicted_sentence):
    fig = plt.figure(figsize=(10,10))
    ax = fig.add_subplot(1, 1, 1)

    heatmap = ax.matshow(attention, cmap='rainbow')

    for y in range(attention.shape[0]):
        for x in range(attention.shape[1]):
            ax.text(x , y, '%.4f' % attention[y, x],
                    horizontalalignment='center',
                    verticalalignment='center', color='black')

    fig.colorbar(heatmap)

    fontdict = {'fontsize': 14}
```

```
    ax.set_xticklabels(['']  + sentence, fontdict=fontdict)
    ax.set_yticklabels(['']  + predicted_sentence, fontdict=fontdict)

    plt.show()
```

[21]:
```
def translate(sentence, encoder, decoder, inp_lang, targ_lang, max_length_inp,␣
↪max_length_targ):
    result, sentence, attention_plot = evaluate(sentence, encoder, decoder,␣
↪inp_lang, targ_lang, max_length_inp, max_length_targ)

    print('Input: {}'.format(sentence))
    print('Predicted translation: {}'.format(result))

    attention_plot = attention_plot[:len(result.split(' ')), :len(sentence.
↪split(' '))]
    plot_attention(attention_plot, get_display(arabic_reshaper.
↪reshape(sentence)).split(' '), result.split(' '))
```

Untuk pakai, tinggal call method `translate`, dan dilengkapi dengan parameternya. Misalkan saja: `translate(u'  ', encoder, decoder, inp_lang, targ_lang, max_length_inp, max_length_targ)`

L.2 *Logbook* kegiatan

Jadwal Pelaksanaan Penelitian:

No	Kegiatan	Waktu Pelaksanaan (Bulan)					
		5	6	7	8	9	10
1	**Pengumpulan data**	v					
	pengembangan *crawler* untuk pengumpulan data						
	Setting server						
	Deployment *crawler*						
2	***preprocessing* data**		v	v			
	pembersihan data						
	normalisasi data						
	anotasi data secara manual						
3	Training data				v		
	Setting server untuk *training*						
	analisa pada model						
4	Pengukuran Hasil					v	
5	Penulisan laporan						v
6	Publikasi hasil Penelitian						v

L.3 *Laporan Keuangan*

No	Nama	Kuantitas	Harga Satuan	Total Biaya
1	IDE Pycharm	1 tahun	3.570.411	3.570.411
2	VPS	12 bulan	53.914	593.347
3	VPS dengan GPU	24 jam	25.666	616.003
4	anotasi data manual	1000 kalimat	1.000	1.000.000
Total				5.779.761

OVH.com

Order Number: **9179743**

Invoice: **WE1779447**
Date: **September 13, 2021**
Payment: **Credit/Debit Card**

Billing contact:

Alvian Burhanuddin
Melati Street, Malang, East Java
Indonesia

VPS Category

INSTALLATION	Domain	Quantity	Unit price	Amount
GPU Service	vps294601.vps.ovh.ca	1	$0.00 USD	$0.00 USD
Ubuntu 20.04 64 bits operating system - english	vps294601.vps.ovh.ca	1	$0.00 USD	$0.00 USD
Localisation canada	vps294601.vps.ovh.ca	1	$0.00 USD	$0.00 USD
			SUB-TOTAL	**$0.00 USD**

SUBSCRIPTION	Domain	Quantity	Unit price	Amount
GPU VPS t1-45 - 24 hour From September 13, 2021 to September 14, 2021	vps294601.vps.ovh.ca	24	$1.79 USD	$42.96 USD
			SUB-TOTAL	**$42.96 USD**

SUBSCRIPTION	**$42.96 USD**
TOTAL	**$42.96 USD**

You can disable the automatic renewal feature at any time in your Control Panel by going to the Billing section, then My Services.
Additionally, any cancellation requests for products/solutions must be made before the 20th of the month

CLOUDCONE

INVOICE

REFERENCE: #464257
BILLING DATE: 11/06/2021
DUE DATE: 11/06/2021

OUR INFORMATION

CloudCone, LLC

30 N. Gould Street,
Suite 6329,
Sheridan. WY 82801
United States of America

BILLING TO

Alvian Burhanuddin

Indonesia
P: +62 81 334 329 292

PRODUCT	AMOUNT	VAT	PRICE	TOTAL
Cloud Servers Add funds to account via PayPal	1	$ 3.76	$ 37.62	$41.38
			Total	**$ 41.38**
			Total due	**$ 41.38**

PAID

PAYMENT INFORMATION

Added funds of $41.38 via PayPal on 11/06/2021

JetBrains s.r.o.
Kavčí Hory Office Park, Na hřebenech II 1718/10
Prague 4 - Nusle, 140 00, Czech Republic
tel. +420 241 722 501 fax +420 241 722 540
e-mail: sales@jetbrains.com
https://www.jetbrains.com

Quote 1803/875508

To:
Alvian Burhanuddin
alvianthelfargy@gmail.com
Melati Street, Malang, 65174, Indonesia

Reference number: 1803/875508 081334329292

Quote Details:
Quote date: June 01, 2021
Valid until: June 02, 2021
Submitted by: Babu Katanga

	Product Description	Price per Item	Qty	Extended Price
All Product Pack APP-A.APP-Y-01	**Personal annual subscription** **Valid June 02, 2021 through June 01, 2022** Purchase of Toolbox Subscription Order / PO Number: R1812586	**US $249.00**	**1**	**US $249.00**

Total:	US $249.00
TAX 0%:	US $0.00
Grand Total:	US $249.00

Payment terms:
Pay online via credit card, PayPal or request proforma invoice. Depending on your locale, additional payment options may be available during checkout.
License delivery is immediate.
An order placed via a purchase order, JetBrains store or otherwise based on this quote is subject to JetBrains Terms and Conditions of Purchase.

L.4 Daftar Riwayat Hidup

1. Nama : Alvian Burhanuddin
2. NIM : 19841009
3. Tempat/Tanggal Lahir : Malang/02 Januari 1994
4. Jenis Kelamin : Laki-Laki
5. Tahun Angkatan : 2019
6. Jurusan : Teknik Informatika
7. Alamat Rumah : Desa Gunung ronggo, Rt. 003 Rw. 001, Kecamatan Tajinan, Kabupaten Malang
8. Telpon/Hp : 081334329292
9. Email : alvianthelfarqy@gmail.com
10. Riwayat Pendidikan

No	Jenjang Pendidikan	Tahun Lulus
1	S1	2016
2	S2	-

11. Pengalaman Organisasi

No	Jenis/Nama organisasi	Jabatan	Tahun
1	-	-	-
2	-	-	-

12. Karya Ilmiah

No	Judul	Tahun	Keterangan
1	Analysis of the Spread of COVID-19 in Local Areas in Indonesia	2021	2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT) (EICONCIT'21)
2	Sistem informasi pengadaan barang dan jasa melalui penyedia di Unit Layanan Pengadaan Universitas Islam Negeri Maulana Malik Ibrahim Malang	2016	skripsi di Universitas Islam Negeri Maulana Malik Ibrahim

Demikian Daftar Riwayat Hidup saya buat dengan sebenarnya, dan apabila dikemudian hari terdapat kesalahan maka saya siap mempertanggungjawabkannya.

Malang, Mei 2021

Yang menyatakan

Alvian Burhanuddin

NIM. 19841009

1. Nama : Rizqi Amaliya

2. NIM : 19841008

3. Tempat/Tanggal Lahir : Malang/27 mei 1993

4. Jenis Kelamin : Perempuan

5. Tahun Angkatan : 2019

6. Jurusan : Teknik Informatika

7. Alamat Rumah : Desa Ganjaran, Rt. 20 Rw. 003,
Kecamatan Gondanglegi, Kabupaten Malang

8. Telpon/Hp : 085730807586

9. Email : 19841008@student.uin-malang.ad.id

10. Riwayat Pendidikan

No	Jenjang Pendidikan	Tahun Lulus
1	S1	2016
2	S2	-

11. Pengalaman Organisasi

No	Jenis/Nama organisasi	Jabatan	Tahun
1	-	-	-
2	-	-	-

12. Karya Ilmiah

No	Judul	Tahun	Keterangan
1	Sistem tracking mahasantri berbasis web: Studi kasus pusat Maḥad Al-Jamiáh Universitas Islam Negeri Maulana Malik Ibrahim Malang	2016	skripsi di Universitas Islam Negeri Maulana Malik Ibrahim

Demikian Daftar Riwayat Hidup saya buat dengan sebenarnya, dan apabila dikemudian hari terdapat kesalahan maka saya siap mempertanggungjawabkannya.

Malang, Mei 2021

Yang menyatakan

Rizqi Amaliya

NIM. 19841008

1. Nama : Ahmad Latif Qosim

2. NIM : 19841007

3. Tempat/Tanggal Lahir : Ngawi/27 Desember 1980

4. Jenis Kelamin : Laki-laki

5. Tahun Angkatan : 2019

6. Jurusan : Teknik Informatika

7. Alamat Rumah : Jalan simpang borobudur utara III No.18, Kecamatan Lowokwaru, Kota Malang

8. Telpon/Hp : 081359801185

9. Email : 19841007@student.uin-malang.ad.id

10. Riwayat Pendidikan

No	Jenjang Pendidikan	Tahun Lulus
1	S1	2010
2	S2	-

11. Pengalaman Organisasi

No	Jenis/Nama organisasi	Jabatan	Tahun
1	-	-	-
2	-	-	-

12. Karya Ilmiah

No	Judul	Tahun	Keterangan
1	*Analysis classification opinion of policy government annouces cabinet reshuffle on youtube comment using 1D convolutional neural network*	2021	2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT) (EICONCIT'21)

Demikian Daftar Riwayat Hidup saya buat dengan sebenarnya, dan apabila dikemudian hari terdapat kesalahan maka saya siap mempertanggungjawabkannya.

Malang, Mei 2021

Yang menyatakan

Ahmad Qosim Latif

NIM. 19841007