

**OPTIMASI MODEL PERCAKAPAN  
BAHASA INDONESIA BERBASIS  
*SEQUENCE TO SEQUENCE (SEQ2SEQ)***

**TUGAS AKHIR**

Diajukan untuk Memenuhi Sebagian Persyaratan  
Mencapai Derajat Sarjana Teknik Informatika



Oleh:

Yohanes Dwiki Witman Gusti Made

14 07 07748

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS ATMA JAYA YOGYAKARTA  
2019**

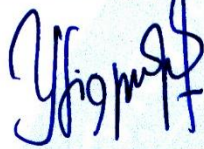
# HALAMAN PENGESAHAN

Tugas Akhir Berjudul  
**Optimasi Model Percakapan Bahasa Indonesia  
Berbasis *Sequence to Sequence* (SEQ2SEQ)**

**Disusun oleh:**  
Yohanes Dwiki Witman Gusti Made  
(NIM: 140707748)

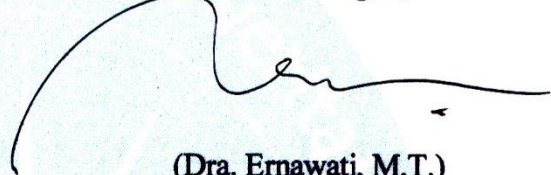
Dinyatakan telah memenuhi syarat  
Pada Tanggal : 28 Februari 2019

Pembimbing I,



(Y. Sigit Purnomo WP., S.T., M.Kom.)

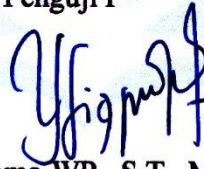
Pembimbing II,



(Dra. Ernawati, M.T.)

**Tim Penguji:**

Penguji I



(Y. Sigit Purnomo WP., S.T., M.Kom.)

Penguji II,



(Yulius Harjoseputro, S.T., M.T.)

Penguji III,



(Joseph Eric Samodra, S.Kom., MIT)

Yogyakarta, 04 Maret 2019  
Universitas Atma Jaya Yogyakarta  
Fakultas Teknologi Industri

  
Dekan,

(Dr. A. Teguh Siswantoro, M.Sc.)

## **HALAMAN PERSEMBAHAN**

*“Non Scholae sed Vitae Discimus”*

## KATA PENGANTAR

Puji syukur kepada Tuhan atas semua berkat, tuntunan dan karunia-Nya yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan tugas akhir “Optimasi Model Percakapan Bahasa Indonesia Berbasis *Sequence to Sequence* (SEQ2SEQ)” ini dengan baik. Tugas akhir adalah tugas yang diwajibkan pada mahasiswa Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta setelah lulus mata kuliah teori, praktikum, dan kerja praktek. Tujuan dari pembuatan tugas akhir ini adalah sebagai salah satu syarat untuk mencapai derajat sarjana Teknik Informatika dari Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta.

Penulis menyadari bahwa dalam pembuatan tugas akhir ini tidak terlepas dari bantuan berbagai pihak yang telah menyumbangkan pikiran, tenaga, dukungan, bimbingan, dan doa kepada penulis baik secara langsung maupun tidak langsung. Oleh sebab itu, penulis mengucapkan terimakasih kepada:

1. Tuhan Yesus Kristus yang telah memberikan petunjuk dan tuntunan, serta melimpahkan berkat, perlindungan, penjaga dan karunia-Nya yang besar kepada penulis.
2. Kedua orang tua penulis serta kakak yang selalu mendukung dan memberikan semangat dalam segala hal untuk penulis sehingga dapat sampai pada titik ini.
3. Bapak Dr. A. Teguh Siswanto selaku Dekan Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta.
4. Bapak Y. Sigit Purnomo WP., S.T., M.Kom., selaku Dosen Pembimbing I yang telah meluangkan waktu dan pikiran untuk memberi bimbingan, petunjuk dan pengarahan kepada penulis sehingga tugas akhir ini dapat diselesaikan dengan baik.
5. Ibu Dra. Ernawati, M.T., selaku Dosen Pembimbing II yang telah meluangkan waktu dan pikiran untuk memberi bimbingan, petunjuk dan pengarahan kepada penulis sehingga tugas akhir ini dapat diselesaikan dengan baik.

6. Seluruh Dosen dan Staf Pengajar Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta yang telah membantu penulis selama masa kuliah di Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta.
7. Seluruh teman-teman yang mendukung penulis yang tidak dapat disebutkan satu-persatu.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki penulis. Oleh karena itu segala kritik dan saran yang bersifat membangun sangat diharapkan. Akhir kata, semoga tugas akhir ini dapat berguna dan bermanfaat bagi semua pihak.

Yogyakarta, 5 Maret 2019

Penulis,

Yohanes Dwiki Witman Gusti Made

# DAFTAR ISI

HALAMAN JUDUL .....	i
HALAMAN PENGESAHAN .....	ii
HALAMAN PERSEMBAHAN .....	iii
KATA PENGANTAR .....	iv
DAFTAR ISI .....	vi
DAFTAR GAMBAR .....	viii
DAFTAR TABEL .....	ix
DAFTAR KODE .....	x
INTISARI .....	xi
BAB I .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	4
1.3. Tujuan Penelitian .....	4
1.4. Batasan Masalah .....	4
1.5. Metodologi Penelitian .....	5
1.6. Alat dan Bahan .....	6
1.7. Sistematika Penulisan Laporan .....	7
BAB II .....	9
BAB III .....	12
3.1. Aplikasi <i>Chatbot</i> .....	12
3.2. <i>Natural Language Processing</i> .....	13
3.3. <i>Deep Learning</i> .....	13
3.4. <i>Recurrent Neural Network</i> (RNN) .....	14
3.5. Mekanisme Gerbang RNN .....	15
3.6. <i>Sequential to Sequential</i> (SEQ2SEQ) .....	17
3.7. Mekanisme Perhatian .....	18
3.8. <i>Gradient Descent</i> .....	20
3.9. <i>Framework Deep Learning</i> .....	20

3.10. Tensorflow .....	21
BAB IV .....	23
4.1. Analisis Algoritma RNN Pada Model SEQ2SEQ .....	23
4.2. Alur Pengerjaan Model Optimasi .....	24
4.3. Perancangan Sampel Data.....	25
4.4. Perancangan Model Optimasi .....	26
4.4.1. Model Gerbang LSTM-LSTM .....	27
4.4.2. Model Gerbang LSTM-GRU.....	28
4.4.3. Model Gerbang GRU-GRU.....	29
4.4.4. Model Gerbang GRU-LSTM.....	30
4.5. Perancangan Algoritma dan <i>Pseudocode</i> .....	30
4.5.1. Pelatihan Model.....	30
4.5.2. Generasi Respon .....	31
4.6. Perancangan Pengujian Model.....	32
4.7. <i>Framework Deep Learning</i> .....	33
BAB V .....	34
5.1. Implementasi Model Optimasi.....	34
5.1.1. Fungsi-Fungsi Umum .....	34
5.1.2. Kode Model Optimasi .....	41
5.1.3. Kode Proses Pelatihan dan Pengujian .....	46
5.1.4. Kode Generasi Respon .....	47
5.2. Pelatihan Model Optimasi.....	48
5.2.1. Lama Waktu Pelatihan.....	48
5.2.2. Nilai Metrik dan <i>Loss</i> Pelatihan .....	50
5.2.3. Grafik Nilai <i>Loss</i> Pelatihan.....	53
5.2.4. Ukuran Bobot Hasil Pelatihan .....	55
5.2.5. Analisis Hasil Pelatihan.....	56
5.3. Evaluasi Model Optimasi.....	58
5.3.1. Lama Waktu Pengujian .....	58
5.3.2. Nilai Metrik dan <i>Loss</i> Pengujian .....	60
5.3.3. Grafik Nilai <i>Loss</i> Pengujian .....	62
5.3.4. Analisis Hasil Pengujian.....	64
5.4. Antarmuka dan Hasil Generasi Respon .....	66
5.5. Kelebihan dan Kekurangan Sistem .....	68
BAB VI.....	69
6.1. Kesimpulan.....	69
6.2. Saran .....	69
DAFTAR PUSTAKA.....	70

## DAFTAR GAMBAR

Gambar 3.1. <i>Timeline</i> dari Aplikasi <i>Chatbot</i> (Wetstein, 2017).....	13
Gambar 3.2. Ruang Lingkup <i>Deep Learning</i> (Sze <i>et al.</i> , 2017).....	14
Gambar 3.3. Arsitektur RNN Sederhana (Christopher Olah via Github, 2015)....	15
Gambar 3.4. Unit LSTM (François Deloche via Wikimedia Commons, 2017)....	16
Gambar 3.5. Unit GRU (François Deloche via Wikimedia Commons, 2017).....	17
Gambar 3.6. Arsitektur SEQ2SEQ (CIS530 University of Pennsylvania, 2018)..	18
Gambar 4.1. Ilustrasi Proses Pelatihan serta Pengujian dan Generasi Respon.....	24
Gambar 4.2. Alur Pengerjaan Model Percakapan .....	25
Gambar 4.3. Alur Pengerjaan Sampel Data.....	26
Gambar 5.1. Contoh Hasil Fungsi <i>Preprocessing Text</i> .....	35
Gambar 5.2. Contoh Hasil Fungsi <i>Load Data Sample</i> .....	36
Gambar 5.3. Contoh Hasil Fungsi <i>Encode Sequences</i> .....	37
Gambar 5.4. Contoh Hasil Fungsi <i>Encode Output</i> .....	38
Gambar 5.5. Contoh Angka Prediksi Dari Fungsi <i>Predict Sequences</i> .....	40
Gambar 5.6. Hasil Visualisasi Model Gerbang LSTM-LSTM .....	42
Gambar 5.7. Hasil Visualisasi Model Gerbang LSTM-GRU .....	43
Gambar 5.8. Hasil Visualisasi Model Gerbang GRU-GRU .....	44
Gambar 5.9. Hasil Visualisasi Model Gerbang GRU-LSTM.....	45
Gambar 5.10. Kesalahan <i>Memory Error</i> .....	48
Gambar 5.11. Grafik Nilai <i>Loss</i> Pelatihan Pada Sampel Data 5 Ribu.....	53
Gambar 5.12. Grafik Nilai <i>Loss</i> Pelatihan Pada Sampel Data 10 Ribu.....	54
Gambar 5.13. Grafik Nilai <i>Loss</i> Pelatihan Pada Sampel Data 15 Ribu.....	54
Gambar 5.14. Grafik Nilai <i>Loss</i> Pelatihan Pada Sampel Data 20 Ribu.....	54
Gambar 5.15. Grafik Nilai <i>Loss</i> Pengujian Pada Sampel Data 5 Ribu .....	63
Gambar 5.16. Grafik Nilai <i>Loss</i> Pengujian Pada Sampel Data 10 Ribu .....	63
Gambar 5.17. Grafik Nilai <i>Loss</i> Pengujian Pada Sampel Data 15 Ribu .....	64
Gambar 5.18. Grafik Nilai <i>Loss</i> Pengujian Pada Sampel Data 20 Ribu .....	64
Gambar 5.19. Antarmuka Program <i>Chatbot</i> .....	68



## DAFTAR TABEL

Tabel 2.1. Perbandingan Teknik Optimasi Model SEQ2SEQ.....	11
Tabel 4.1. Tabel Variasi Optimasi Model SEQ2SEQ .....	27
Tabel 4.2. Tabel Struktur <i>Layer</i> Model LSTM-LSTM .....	28
Tabel 4.3. Tabel Struktur <i>Layer</i> Model LSTM-GRU.....	28
Tabel 4.4. Tabel Struktur <i>Layer</i> Model GRU-GRU .....	29
Tabel 4.5. Tabel Struktur <i>Layer</i> Model GRU-LSTM.....	30
Tabel 5.1. Lama Waktu Pelatihan Model Tanh RNN – Tanh RNN Dengan <i>Gradient Descent</i> SGD .....	49
Tabel 5.2. Lama Waktu Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan <i>Gradient Descent</i> Adam .....	49
Tabel 5.3. Nilai Metrik dan <i>Loss</i> Pelatihan Model Tanh RNN – Tanh RNN Dengan <i>Gradient Descent</i> SGD .....	50
Tabel 5.4. Nilai Metrik dan <i>Loss</i> Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan <i>Gradient Descent</i> Adam.....	51
Tabel 5.5. Ukuran Bobot Hasil Pelatihan Model Tanh RNN – Tanh RNN Dengan <i>Gradient Descent</i> SGD .....	55
Tabel 5.6. Ukuran Bobot Hasil Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan <i>Gradient Descent</i> Adam.....	55
Tabel 5.7. Lama Waktu Pengujian Model Tanh RNN – Tanh RNN Dengan <i>Gradient Descent</i> SGD .....	58
Tabel 5.8. Lama Waktu Pengujian Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan <i>Gradient Descent</i> Adam .....	59
Tabel 5.9. Nilai Metrik dan <i>Loss</i> Pengujian Model Tanh RNN – Tanh RNN Dengan <i>Gradient Descent</i> SGD .....	60
Tabel 5.10. Nilai Metrik dan <i>Loss</i> Pengujian Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan <i>Gradient Descent</i> Adam.....	61

## DAFTAR KODE

Kode 4.1. <i>Pseudocode</i> Proses Pelatihan .....	31
Kode 4.2. <i>Pseudocode</i> Proses Generasi Respon.....	32
Kode 5.1. Fungsi <i>Preprocessing Text</i> .....	35
Kode 5.2. Fungsi <i>Load Data Sample</i> .....	36
Kode 5.3. Fungsi <i>Create Tokenizer</i> .....	36
Kode 5.4. Fungsi <i>Max Length</i> .....	37
Kode 5.5. Fungsi <i>Encode Sequences</i> .....	37
Kode 5.6. Fungsi <i>Encode Output</i> .....	38
Kode 5.7. Fungsi <i>Word For Id</i> .....	39
Kode 5.8. Fungsi <i>Predict Sequences</i> .....	39
Kode 5.9. Fungsi <i>Translate</i> .....	40
Kode 5.10. Fungsi <i>Precision</i> .....	41
Kode 5.11. Fungsi <i>Recall</i> .....	41
Kode 5.12. Fungsi <i>F1</i> .....	41
Kode 5.13. Fungsi Model Gerbang LSTM-LSTM.....	42
Kode 5.14. Fungsi Model Gerbang LSTM-GRU .....	43
Kode 5.15. Fungsi Model Gerbang GRU-GRU .....	44
Kode 5.16. Fungsi Model Gerbang GRU-LSTM .....	45
Kode 5.17. Proses Pelatihan dan Pengujian .....	47
Kode 5.18. Proses Generasi Respon .....	47

## INTISARI

*Chatbot* saat ini berkembang dengan kerangka kerja *sequence-to-sequence* (SEQ2SEQ) yang berbasiskan dua *Recurrent Neural Network* (2-RNN). Seiring waktu, tingkat kesulitan pelatihannya semakin kompleks karena pelatihan jangka panjang dengan RNN sangat rentan pada masalah hilang atau meledaknya gradien. Oleh karena itu, pelatihan dengan RNN akhir-akhir ini lebih fokus kepada upaya untuk mengatasi masalah hilang atau meledaknya gradien melalui langkah optimasi.

Beberapa metode optimasi pernah diteliti sebelumnya pada kerangka kerja SEQ2SEQ, seperti mekanisme gerbang *Long-Short Term Memory* (LSTM) yang baik dalam pemahaman informasi secara lebih lama dan *Gated Recurrent Unit* (GRU) yang konvergensi pelatihannya cepat, mekanisme perhatian (*Attention Mechanism*) yang mengatasi masalah sekuens yang panjang, dan *gradient descent* yang menghasilkan model dengan tingkat kesalahan minimal. Perlu adanya penelitian terbaru yang mengungkapkan hasil gabungan dari ketiga optimasi tersebut untuk mencapai *state-of-the-art* dari model percakapan.

Hasil penelitian ini menunjukkan performa optimasi model percakapan SEQ2SEQ berdasarkan nilai metrik *accuracy*, *precision*, *recall*, *F1*, dan *loss* dengan memakai sampel data dari *OpenSubtitle 2018* Bahasa Indonesia. Penelitian ini juga berhasil menerapkan mekanisme gerbang yang berbeda pada *encoder* dan *decoder* model. Posisi optimal seluruh model terletak pada sampel 15 ribu dan *epoch* ke-50, dengan skor pengujian terbaik dihasilkan oleh model gerbang LSTM-LSTM, yaitu *accuracy* 71,59%, *precision* 86,95%, *recall* 66,37%, *F1* 75,25%, dan *loss* 2,664.

**Kata Kunci:** *Chatbot*, *SEQ2SEQ*, *RNN*, *Optimasi*, *Bahasa Indonesia*

# BAB I

## Pendahuluan

### 1.1.Latar Belakang

*Natural Language Processing* (NLP) atau disebut sebagai pemrosesan bahasa alami adalah teori yang termotivasi dari teknik komputasi untuk analisa otomatis dan representasi bahasa manusia. NLP memungkinkan komputer untuk melakukan tugas bahasa alami, seperti penguraian dan pelabelan kelas kata, terjemahan mesin, hingga yang populer saat ini adalah sistem dialog. Sistem dialog memungkinkan manusia sebagai aktor untuk berinteraksi dengan mesin atau lebih dikenal sebagai *chatbot*.

*Chatbot* adalah sebuah program dari pemodelan percakapan yang mensimulasikan sebuah percakapan interaktif antara mesin dan manusia dengan menggunakan bahasa alami manusia (Shawar *et al.*, 2007). Awalnya *chatbot* berpotensi untuk menggantikan peran *customer care*, karena tingginya biaya dukungan manusia. Sekarang ini *chatbot* mengalami transformasi yang pesat untuk membantu pemesanan, transaksi jual beli, hingga sampai kepada aplikasi asisten virtual yang cerdas. Oleh karena manfaatnya yang besar, maka *chatbot* digadang-gadang sebagai pionir teknologi masa depan.

Saat ini *chatbot* sudah semakin moderen dengan semakin banyak penelitian aktif dan perusahaan-perusahaan besar saling berlomba untuk mengembangkan aplikasi *chatbot* menjadi lebih baik. Teknologi *machine learning* telah membantu agen-agen *chatbot*, misalnya *Facebook Messenger Bot*, untuk lebih adaptif terhadap perbedaan gaya input dan tugas (Radziwill *et al.*, 2017). Selama beberapa dekade, pendekatan *machine learning* telah didasarkan pada model yang dangkal, misalnya SVM dan regresi logistik, yang dilatih dengan fitur berdimensi tinggi dan tersebar (Young *et al.*, 2018). Mengandalkan *machine learning* sebenarnya bagus untuk membuat prediksi yang baik, namun pemahaman bahasa alami, bagaimanapun, membutuhkan lebih dari itu (Young *et al.*, 2018).

Jika mengikuti tren saat ini, penelitian *chatbot* semakin bertambah dan berfokus kepada penggunaan jaringan saraf dengan *deep learning*. *Chatbot* menjadi sangat maju sejak keberhasilan Vinyals (2015) membuat model percakapan dengan *dataset* yang besar pada kerangka kerja *sequence-to-sequence* (SEQ2SEQ). Vinyals (2015) mengatasi kerumitan pemetaan antara kueri dan respon, yang sering dibatasi menjadi domain yang sangat sempit, dengan usaha besar rekayasa dari tangan manusia daripada otomatis mesin.

Arsitektur SEQ2SEQ tersusun atas dua *Recurrent Neural Network* (2-RNN). RNN memberikan manfaat untuk mengolah kata-kata yang sekuens dan memori jangka panjang. Pendekatan itu menghasilkan kinerja yang sangat baik dan menghasilkan balasan percakapan yang akurat dan masuk akal pada domain terbuka (Vinyals *et al.*, 2015). SEQ2SEQ bahkan bisa digunakan untuk tugas terjemahan mesin, *question-answer*, dan lainnya tanpa perlu perubahan yang besar arsitekturnya. Kesederhanaannya itu menjadi kekuatan utama SEQ2SEQ sehingga mudah diterapkan di berbagai tugas NLP.

Meskipun kualitas SEQ2SEQ menarik, RNN dianggap gagal menjadi alat utama pembelajaran, karena sulit melatihnya secara efektif. Hubungan antara parameter jaringan dan dinamika RNN di *hidden state* pada setiap langkah waktu bisa menyebabkan ketidakstabilan jaringan (Salehinejad *et al.*, 2017). Disamping itu, RNN membutuhkan banyak iterasi untuk melatih modelnya dengan tujuan meminimalkan *loss* di sepanjang waktu pembelajaran. Fakta-fakta itu menyebabkan tantangan untuk melatih model RNN, seperti: (1) terdapat kemungkinan gradiennya mengecil saat bobotnya kecil, karena perkalian matriks yang berulang kali saat propagasi balik; (2) gradien bisa meledak karena perkalian bobotnya menjadi lebih besar dari batas norma gradien (Salehinejad *et al.*, 2017). Hal itu terkenal sebagai masalah hilang atau meledaknya gradien (*vanishing / exploding gradient*). Oleh karena itu, fokus utama penelitian RNN akhir-akhir ini lebih mengurangi kompleksitas pelatihan dan mengatasi ketidakstabilan gradien melalui langkah-langkah optimasi.

Optimasi RNN pada model SEQ2SEQ pernah diteliti sebelumnya seperti pemakaian mekanisme gerbang *Long-Short Term Memory* (LSTM) (Sutskever

*et al.*, 2014; Vinyals *et al.*, 2015), mekanisme gerbang *Gated Recurrent Unit* (GRU) (Cho *et al.*, 2014), dan mekanisme perhatian (*Attention Mechanism*) (Bahdanau *et al.*, 2014). Setelah itu, *gradient descent* juga membantu perolehan model yang lebih baik dengan mencari kesalahan (*error*) yang paling rendah. Penjelasan tadi mengungkapkan sebagian penelitian tentang optimasi model SEQ2SEQ dan perlu adanya optimasi gabungan untuk membentuk *state-of-the-art* dari model percakapan.

Penelitian ini memakai sampel data dari *dataset* domain terbuka, yang banyak mengandung *noise*, yaitu *OpenSubtitle 2018*. Penelitian ini merupakan pengembangan dari pekerjaan Vinyals (2015) yang memakai *dataset* Bahasa Inggris sehingga masih ada ruang bagi peneliti untuk mengembangkan model percakapan dengan *dataset* Bahasa Indonesia. Memodelkan percakapan juga mudah: sekuens input dapat menjadi rangkaian dari apa yang telah dibicarakan (konteks) dan sekuens output adalah jawabannya (target) (Vinyals *et al.*, 2015). Hal itu bisa dilihat dari dialog film, dimana terjadi pergantian giliran percakapan, yaitu aktor satu bertindak sebagai inisiasi percakapan dan aktor lain bertindak sebagai respon dari percakapan.

Fokus dari penelitian ini adalah implementasi model percakapan SEQ2SEQ dan kemudian membahas strategi penggabungan optimasi menggunakan mekanisme gerbang, mekanisme perhatian, dan *gradien descent*. *Dataset* yang digunakan adalah *OpenSubtitle 2018* Bahasa Indonesia dan *framework* pemrograman yang digunakan adalah Keras dan Tensorflow. Penelitian ini dilakukan untuk melihat performa model yang sudah dioptimalkan pada domain terbuka. Beberapa indikator pengukuran akan digunakan untuk mengetahui performa model dari penelitian ini adalah hasil *accuracy*, *precision*, *recall*, *F1*, dan *loss*. Hasil dari penelitian ini nantinya dapat digunakan sebagai referensi dalam pemilihan metode-metode optimasi terutama dalam tugas pemodelan percakapan.

## 1.2. Rumusan Masalah

Berdasarkan latar belakang permasalahan, maka dapat diidentifikasi permasalahan yang akan dibahas adalah sebagai berikut.

1. Bagaimana mengembangkan optimasi gabungan untuk model percakapan SEQ2SEQ berbasis algoritma RNN dengan sampel data dari *dataset* percakapan *OpenSubtitle 2018* Bahasa Indonesia ?
2. Bagaimana performa dari hasil optimasi gabungan untuk model percakapan SEQ2SEQ berbasis algoritma RNN dengan sampel data dari *dataset* percakapan *OpenSubtitle 2018* Bahasa Indonesia ?

## 1.3. Tujuan Penelitian

Penelitian ini mempunyai tujuan sebagai berikut.

1. Mengembangkan optimasi gabungan terbaik untuk model percakapan SEQ2SEQ berbasis algoritma RNN dengan sampel data dari *dataset* percakapan *OpenSubtitle 2018* Bahasa Indonesia.
2. Mengetahui performa dari hasil gabungan optimasi untuk model percakapan SEQ2SEQ berbasis algoritma RNN dengan sampel data dari *dataset* percakapan *OpenSubtitle 2018* Bahasa Indonesia.

## 1.4. Batasan Masalah

Berdasarkan permasalahan diatas, maka penelitian ini akan dibatasi pada hal-hal sebagai berikut:

1. Indikator yang akan digunakan dalam pengukuran performa dari optimasi gabungan untuk model percakapan SEQ2SEQ berbasis algoritma RNN ini adalah *accuracy*, *precision*, *recall*, *F1*, dan *loss*.
2. Optimasi gabungan untuk model percakapan adalah kombinasi mekanisme gerbang LSTM dan GRU serta penambahan mekanisme perhatian versi Bahdanau dan *gradient descent* Adam.
3. Sampel data diambil dari *OpenSubtitle 2018* Bahasa Indonesia dengan ukuran 5 ribu, 10 ribu, 15 ribu, dan 20 ribu baris data.

## 1.5. Metodologi Penelitian

Adapun metode yang digunakan adalah sebagai berikut:

### 1. Studi Pustaka

Pada bagian langkah studi pustaka ini dilakukan untuk mencari sumber pustaka yang berkaitan dengan aplikasi yang akan dibuat. Pada penelitian ini dilakukan pencarian sumber pustaka untuk model percakapan SEQ2SEQ berbasis algoritma RNN. Selanjutnya, dilakukan pencarian sumber pustaka tentang optimasi model SEQ2SEQ seperti penambahan mekanisme gerbang, mekanisme perhatian, dan *gradient descent*. Studi pustaka membantu pekerjaan dalam hal pembuatan model dan memberikan langkah-langkah selanjutnya dengan diberikan teori yang sudah ada. Melalui metode penelitian didapatkan data yang merupakan hasil dari sumber pustaka yang sudah tertulis sebelumnya.

### 2. Analisis Algoritma

Tahapan analisis algoritma dilakukan untuk memahami langkah-langkah proses dan cara kerja model percakapan yang akan dibuat dengan metode yang telah ditentukan. Selanjutnya adalah memahami langkah kerja optimasi model SEQ2SEQ berbasis algoritma RNN dan mencari cara untuk menggabungkan teknik-teknik optimasi yang ada yaitu penggabungan mekanisme gerbang, mekanisme perhatian, dan *gradient descent*. Analisis dilakukan dengan melakukan studi literatur dari teori-teori yang didapatkan dari hasil studi pustaka. Tahapan ini dapat menghasilkan gambaran umum tentang model optimasi yang akan dibuat, langkah proses kerjanya, dan referensi arsitektur model optimasi yang akan dibuat.

### 3. Perancangan Model Optimasi

Tahap perancangan model digunakan untuk merancang model optimasi berdasarkan gambaran, langkah proses kerja, dan arsitektur



model optimasi yang telah didefinisikan pada tahap analisis algoritma. Tahap perancangan ini menghasilkan alur dan *pseudocode* dari model yang akan dibuat.

#### 4. Implementasi Model Optimasi

Tahap implementasi dilakukan untuk mewujudkan rancangan yang telah didefinisikan pada tahapan sebelumnya. Hasilnya adalah sebuah model yang telah dioptimalkan sesuai dengan tujuan penelitian.

#### 5. Evaluasi Model

Tahap evaluasi model adalah tahap untuk menguji model yang telah dibuat pada tahap implementasi. Evaluasi model dilakukan untuk menguji indikator-indikator sesuai dengan tujuan penelitian sehingga menghasilkan sebuah model yang optimal dan sesuai dengan tujuan penelitian. Evaluasi model dilakukan berdasarkan parameter yang telah ditentukan, dan hasil akhirnya dapat dijadikan bahan referensi untuk penelitian selanjutnya.

### 1.6. Alat dan Bahan

Pada penelitian ini akan digunakan *dataset OpenSubtitle 2018* Bahasa Indonesia dan bahasa pemrograman utamanya adalah Python versi 3.6 yang berjalan pada sistem operasi berbasis Windows. Perangkat keras untuk melakukan penelitian ini yaitu:

1. *Personal computer* (PC) untuk proses pembangunan dan percobaan model dengan Windows 10 Pro versi 1803 (Redstone 4), *processor* Intel Pentium G4560 @3.50 GHz, RAM 16 GB, dan GPU Nvidia GTX 1070 8 GB dengan 1920 CUDA *cores*.
2. *Personal computer* (PC) untuk proses pembangunan dan percobaan model dengan sistem operasi Windows 7 Ultimate, *processor* Intel

CPU Intel Core i7 @3.50 Ghz, RAM 16 GB, dan GPU Nvidia GeForce GTX 660 Ti 3 GB dengan 1344 CUDA *cores*.

Dalam penelitian ini juga menggunakan beberapa *library* untuk bahasa pemrograman Python. *Library* yang digunakan dalam penelitian ini yaitu.

1. Numpy 1.15.4 untuk pemrosesan *array* dan matriks.
2. Python CSV untuk membaca sampel data dengan ekstensi *Tab Separated Values* (TSV).
3. Tensorflow 1.12.0 untuk *framework deeplearning*.
4. Keras 2.2.4 untuk *framework deep learning* yang bekerja diatas Tensorflow.
5. Tensorboard 1.12.0 untuk visualisasi hasil penelitian.
6. NLTK dan Re untuk proses pembersihan dan tokenisasi sampel data.
7. Spyder 3.3.2 untuk lingkungan pengembangan yang *scientific programming*.

### **1.7. Sistematika Penulisan Laporan**

Laporan tugas akhir ini disusun dengan sistematika sebagai berikut.

#### **BAB I : Pendahuluan**

Pada bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, alat dan bahan, metodologi penelitian, dan sistematika penulisan laporan yang terkait dengan proses penelitian ini.

#### **BAB II : Tinjauan Pustaka**

Bab ini berisi tentang penjelasan singkat dan ringkasan mengenai penelitian terdahulu yang berhubungan atau memiliki kesamaan dengan permasalahan yang akan dibahas oleh penulis di dalam tugas akhir ini.

#### **BAB III : Landasan Teori**

Pada bab ini berisi dasar-dasar teori yang melandasi serta mendukung dalam implementasi teknik optimasi dan proses yang berhubungan dengan pembangunan model yang optimal.

#### **BAB IV : Analisis dan Perancangan Model Optimasi**

Bab ini berisi penjelasan uraian analisis algoritma dan perancangan model optimasi yang akan dibuat.

#### **BAB V : Implementasi dan Evaluasi Model Optimasi**

Bab ini berisi penjelasan mengenai hasil dari penelitian ini yang berupa data-data hasil implementasi dan pengujian serta pembahasannya.

#### **BAB VI : Kesimpulan dan Saran**

Pada bab ini berisi kesimpulan mengenai penelitian yang telah dibuat beserta saran-saran yang berguna bagi pengembangan lebih lanjut.

## BAB II

### Tinjauan Pustaka

Pada tahun 1950-an, ilmuwan Alan Turing mencetuskan tes Turing, yaitu tes kemampuan komputer untuk mengidentifikasi dengan benar apakah seseorang sedang berbicara dengan manusia atau komputer. Joseph Weizenbaum kemudian menciptakan model percakapan pertama dengan nama ELIZA dan punya kapabilitas untuk melakukan tes Turing pada masa itu. Akan tetapi, sampai saat ini belum ada satupun komputer yang lolos tes Turing semenjak tes itu diperkenalkan, sehingga menjadi pemicu bagi peneliti untuk mengembangkan model percakapan yang lebih baik.

Perkembangan model percakapan dalam beberapa tahun terakhir ditandai dengan banyaknya penelitian jaringan saraf. Model percakapan dengan jaringan saraf yang populer saat ini adalah *sequence-to-sequence* (SEQ2SEQ). Model ini dibentuk dari dua jaringan RNN (2-RNN) dan juga dikenal sebagai model *encoder-decoder*. Kegunaannya dipopulerkan oleh Sutskever (2014) untuk tugas terjemahan mesin dalam menerjemahkan Bahasa Inggris ke Bahasa Prancis. Sutskever (2014) memakai *multilayer* LSTM pada model SEQ2SEQ dan membalik urutan kata pada kalimat sumber sehingga terjadi banyak depedensi jangka pendek untuk mengatasi masalah pengoptimalan yang jauh lebih sederhana. Setelah itu, terdapat penelitian SEQ2SEQ dengan *Gated Recurrent Unit* (GRU). GRU diketahui mampu mempelajari representasi frasa bahasa secara semantik dan sintaksis yang diperkenalkan pada tugas terjemahan mesin (Cho *et al.*, 2014).

Mekanisme gerbang LSTM dan GRU telah berhasil menangani masalah gradien yang menghilang atau meledak selama pelatihan jangka panjang. Akan tetapi, butuh beberapa bukti empiris yang menunjukkan unit gerbang mana yang paling baik untuk masalah optimasi model percakapan. Chung (2014) menemukan bahwa GRU menghasilkan model yang setara dengan LSTM, tetapi GRU secara komputasi lebih efisien. Hal itu disebabkan karena GRU disusun atas dua gerbang: *reset* dan *update*; sedangkan LSTM disusun atas tiga gerbang yaitu *input*, *output*, dan *forget* serta terdapat unit memori yang dikontrol oleh gerbang tersebut (Chung

*et al.*, 2014). GRU mempunyai sedikit parameter sehingga pelatihan dan konvergensinya lebih cepat daripada LSTM. Akan tetapi, LSTM pada prinsipnya dapat menggunakan unit memorinya untuk mengingat informasi yang jaraknya jauh dan melacak berbagai atribut yang sedang diproses (Karpathy *et al.*, 2016). LSTM mempertahankan pemahaman informasi di setiap langkah waktu secara lebih lama, sehingga untuk pelatihan jangka panjang dengan data yang besar, secara teori LSTM lebih baik daripada GRU.

Arsitektur SEQ2SEQ itu sendiri juga memiliki keterbatasan. Model SEQ2SEQ memerlukan seluruh input yang dikodekan ke dalam vektor tunggal dengan panjang yang berukuran tetap, sehingga model tidak dapat menggeneralisasi input jauh lebih lama daripada kapasitas tetapnya. Salah satu cara untuk mengatasi masalah ini adalah dengan memakai mekanisme perhatian (Bahdanau *et al.*, 2014). Mekanisme perhatian mampu memperluas vektor ukuran tetap dengan memungkinkan model secara otomatis mencari bagian dari sumber kalimat yang relevan untuk memprediksi kata target, tanpa harus membentuk bagian-bagian itu sebagai segmen padat secara eksplisit.

Metode optimasi yang juga sering digunakan pada *deep learning* adalah *gradient descent* (GD). Ide dasar GD adalah menyesuaikan bobot model dengan mencari turunan dari fungsi biaya (*error*) dan secara iteratif bergerak menuju nilai parameter yang meminimalisasi nilai dari fungsi biaya (*error*) tersebut. Contoh ekstensi GD adalah AdaGrad, RMSProp, dan Adam. AdaGrad bekerja dengan baik pada masalah gradien yang tersebar, sedangkan RMSProp cocok untuk masalah pembaruan bobot secara online dan non-stasioner. Adam lalu menggabungkan keuntungan dari dua ekstensi itu untuk mempertahankan tingkat pembelajaran per-parameter dari perkiraan momen pertama dan kedua (Kingma *et al.*, 2014).

Teknik-teknik optimasi yang sudah dijelaskan tadi telah tersedia secara resmi pada *framework deep learning* dan dirilis secara terbuka (*opensource*). Beberapa penelitian telah menunjukkan detail-detail pekerjaan mereka seperti bahasa pemrograman maupun *framework* yang digunakan pada penelitian mereka. Perbandingan mengenai teknik optimasi model SEQ2SEQ tersebut dapat dilihat pada Tabel 2.1. berikut ini.

Tabel 2.1. Perbandingan Teknik Optimasi Model SEQ2SEQ

Item Pemandangan	Sutskever (2014)	Cho (2014)	Bahdanau (2014)	Vinyals (2015)	Made (2019)*
Mekanisme Gerbang	LSTM	GRU	GRU dengan <i>Attention</i>	LSTM	Kombinasi LSTM dan GRU dengan <i>Attention</i>
<i>Gradient Descent</i>	SGD	AdaDelta, SGD	SGD, AdaDelta	SGD, AdaGrad	Adam
Bahasa Pemrograman	C++	Python	C++	C++	Python
<i>Deep Learning Framework</i>	-	Theano	-	-	Keras dan Tensorflow
Hasil	Terjemahan Mesin Bahasa Inggris - Prancis	Terjemahan Mesin Bahasa Inggris - Prancis	Terjemahan Mesin Bahasa Inggris - Prancis	Model Percakapan Bahasa Inggris	Model Percakapan Bahasa Indonesia

\* Dalam penelitian ini.

Penelitian ini memakai *OpenSubtitle 2018* Bahasa Indonesia selayaknya yang dikerjakan oleh Vinyals (2015) dengan *OpenSubtitle 2016* Bahasa Inggris. Penelitian yang ada sebagian besar memakai korpus Bahasa Inggris seperti yang ditunjukkan oleh Tabel 2.1. Itu artinya masih ada ruang untuk melakukan penelitian dengan *dataset* Bahasa Indonesia. Tabel 2.1 juga menunjukkan optimasi model SEQ2SEQ bisa dilakukan dengan bantuan kerangka kerja *deep learning*. Penulis memakai *framework* Keras dan Tensorflow untuk implementasi kombinasi gerbang LSTM dan GRU dengan penambahan mekanisme perhatian dan *gradient descent*. Diharapkan hasil performa setelah optimasi akan menghasilkan model yang sepadan atau lebih dari implementasi-implementasi sebelumnya.

## **BAB III**

### **Landasan Teori**

Bab ini akan membahas berbagai teori yang melandasi penulis dalam penelitian yang akan dilakukan.

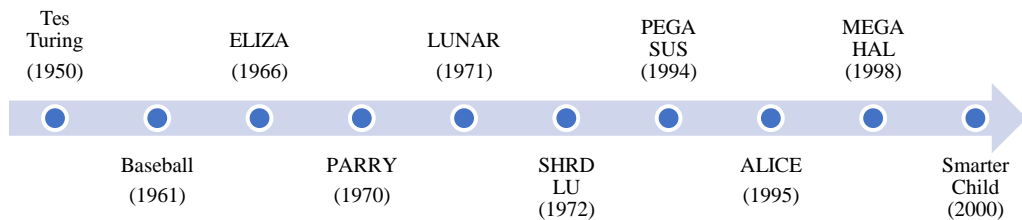
#### **3.1. Aplikasi *Chatbot***

Joseph Weizenbaum menciptakan ELIZA sebagai *chatbot* pertama. Sistem ELIZA dibuat untuk meniru bahasa psikoterapis dan sistemnya didasarkan pada pencocokan kata kunci. Setelah ELIZA, *chatbot* berikutnya adalah PARRY. Berbeda dengan ELIZA, PARRY dimodelkan sebagai pasien paranoid saat dia berbicara ke terapisnya dengan mengandalkan pencocokan pola dan beberapa trik, seperti bercerita, jika tidak ada jawaban yang ditemukan.

Beberapa aplikasi *chatbot* muncul selain bidang kedokteran. BASEBALL dapat menjawab pertanyaan tentang liga bisbol Amerika Serikat dan LUNAR dapat menjawab pertanyaan tentang geologi batuan dan tanah di bulan. Keduanya mirip dengan *chatbot* ELIZA namun didasarkan pada *knowledge base system*. Setelah itu, muncul SHRDLU yang sistemnya dapat memindahkan berbagai balok yang berwarna dan berbentuk. Aplikasi *chatbot* juga diterapkan untuk pemesanan penerbangan yang bernama PEGASUS, di mana pengguna dapat mencari dan memesan penerbangan *American Airlines*.

Ketidakmampuan yang kelihatan jelas dari *chatbot* ELIZA dan PARRY adalah responnya yang dapat diprediksi, berlebihan, dan kurang memiliki kepribadian. Oleh karena itu, pada tahun 1995 *chatbot* ALICE (*Artificial Linguistic Internet Computer Entity*) dibuat untuk mengatasi hal tersebut dan sistemnya berbasiskan pencocokan pola dengan bahasa XML, sehingga sangat mudah untuk menulis program *chatbot* sendiri tanpa pengetahuan pemrograman. Akan tetapi, teknik pencocokan pola tampaknya tidak mengarah pada terobosan dalam kecerdasan buatan. Arah baru penelitian *chatbot* adalah pengenalan sistem MegaHAL yang didasarkan pada teknik pembelajaran mesin. Setelah itu, aplikasi *chatbot* semakin berkembang melalui pesan instan dan jaringan SMS

dengan SmarterChild, yang merupakan dasar dari aplikasi *virtual assistant* saat ini. Berikut adalah *timeline* dari aplikasi *chatbot*.



Gambar 3.1. *Timeline* dari Aplikasi *Chatbot* (Wetstein, 2017)

### 3.2. *Natural Language Processing*

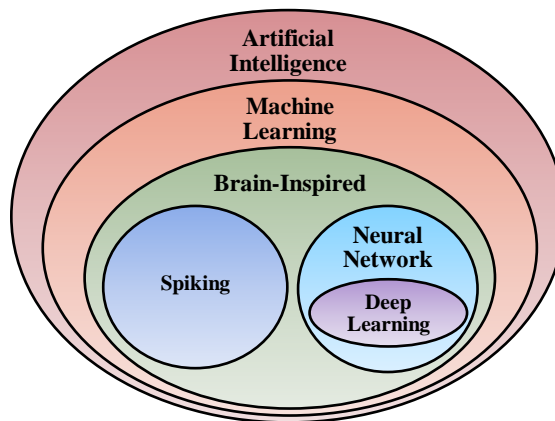
*Natural Language Processing* (NLP) adalah cabang dari kecerdasan buatan yang termotivasi dari teknik komputasi untuk analisa otomatis dan representasi bahasa manusia dengan tujuan memudahkan manusia untuk menjalin komunikasi dengan mesin. Jika mengikuti tren saat ini, penelitian NLP terbaru semakin bertambah dan berfokus pada penggunaan metode baru dari *deep learning* dengan praktik yang paling populer adalah pembelajaran yang diawasi (*supervised learning*). NLP telah menghasilkan beragam aplikasi yang berguna bagi manusia, seperti terjemahan mesin, kategorisasi teks, deteksi spam, ekstraksi informasi, peringkasan dokumen, sistem percakapan, dan aplikasi medis (Khurana *et al.*, 2017).

### 3.3. *Deep Learning*

*Deep learning* adalah kelas dari pembelajaran mesin yang bekerja dalam banyak layer pemrosesan untuk ekstraksi dan transformasi dengan tujuan mempelajari representasi yang hirarkis dari data statistik non-linier. Akhir-akhir ini *deep learning* telah menghasilkan teknologi mutakhir diberbagai bidang pekerjaan, seperti klasifikasi (*supervised*) dan analisis pola (*unsupervised*) terutama untuk memahami data seperti gambar, suara, dan teks.



Alasan penting yang mendasari populernya *deep learning* saat ini adalah akurasi, efisiensi, dan fleksibilitas yang didukung oleh meningkatnya kemampuan perangkat keras untuk membantu proses komputasinya yang berat. *Deep learning* menawarkan cara untuk memanfaatkan sejumlah besar data dengan sedikit rekayasa tangan karena secara otomatis mesin melakukan ekstraksi fitur secara rinci, dimana hal itu memerlukan banyak waktu dan upaya yang intens. *Deep learning* berada pada titik potong yang bisa dilihat pada ilustrasi Gambar 3.2.

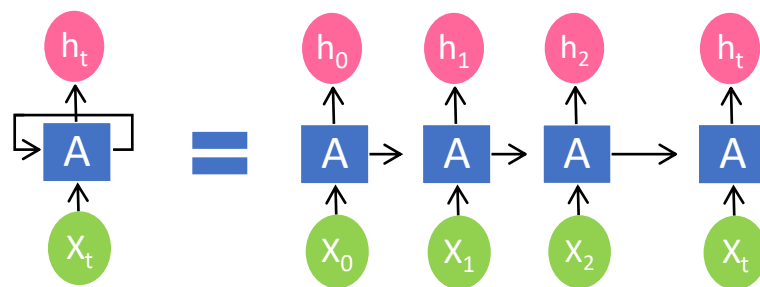


Gambar 3.2. Ruang Lingkup *Deep Learning* (Sze *et al.*, 2017)

### 3.4. *Recurrent Neural Network* (RNN)

*Recurrent Neural Network* (RNN) merupakan salah satu kelas dari *deep neural network* yang diawasi. Pelatihan RNN dalam tingkat yang diawasi membutuhkan *dataset* pelatihan dari pasangan input-target dengan tujuan meminimalkan perbedaan nilai *loss* pasangan itu dengan mengoptimalkan bobot jaringan (Salehinejad *et al.*, 2017). RNN dibentuk dari neuron buatan dengan satu atau lebih umpan balik yang berulang. Pada setiap langkah waktu, neuron akan menerima data, melakukan komputasi, dan menghasilkan keluaran. RNN menangkap dinamika yang kaya dari keadaan tersembunyi untuk konteks jangka panjang, sehingga membentuk model yang ekspresif dan sangat kuat untuk tugas-tugas yang sekuens, seperti pengenalan suara, sintesis ucapan, visi mesin, generasi deskripsi video, dan rangkaian teks.

RNN memiliki tiga lapisan yaitu layer *input*, layer tersembunyi yang berulang, dan layer *output* (Salehinejad *et al.*, 2017). Layer input memiliki unit input, terkoneksi penuh ke unit tersembunyi yang ada di layer tersembunyi. Unit tersembunyi itu terhubung satu sama lain secara berulang. Layer tersembunyi bisa didefinisikan sebagai “memori” atau ruang keadaan yang berdimensi tinggi dengan dinamika non-linier untuk mengingat dan memproses informasi masa lalu. Keadaan tersembunyi akan merangkum semua informasi unik yang diperlukan sebagai keadaan terakhir dari jaringan, melalui serangkaian langkah waktu. Informasi itu lalu terintegrasi, sehingga mampu menentukan perilaku jaringan di masa depan dan melakukan prediksi yang akurat di layer *output*. Arsitektur RNN bisa dilihat pada ilustrasi Gambar 3.3.



Gambar 3.3. Arsitektur RNN Sederhana (Christopher Olah via Github, 2015)

### 3.5. Mekanisme Gerbang RNN

Salah satu kelemahan RNN adalah pembelajaran jangka panjang dengan *gradient descent* bisa menghasilkan masalah menghilang atau meledaknya gradien (Salehinejad *et al.*, 2017). Pada saat pembelajaran, ketika nilai gradien dipropagasikan kembali, nilai itu terus dikalikan dengan bobot yang lebih kecil dari satu ( $<1.0$ ). Oleh karena itu, nilai gradien yang dihasilkan saat iterasi lama kelamaan cenderung mengecil atau mengalami kehilangan nilai. Nilai gradien secara eksponensial juga bisa membesar, karena nilai gradien dikalikan dengan bobot yang lebih besar dari satu ( $>1.0$ ). Maka perlu mencegah nilai gradien tersebut menjadi angka-angka yang terlalu kecil atau terlalu besar.

Salah satu cara mengatasi masalah menghilang atau meledaknya gradien adalah memodifikasi arsitektur model dengan memasukkan unit gerbang yang dirancang khusus untuk menyimpan informasi selama waktu periode yang lama. Mekanisme gerbang yang paling dikenal saat ini adalah *Long-Short Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). LSTM mampu menangani penghafalan dan pengingatan kembali untuk jangka panjang, khususnya data yang sangat besar. LSTM pada prinsipnya dapat menggunakan unit memorinya untuk mengingat informasi yang jaraknya jauh dan melacak berbagai atribut teks yang sedang diproses (Karpathy *et al.*, 2016). Sementara itu, GRU memiliki parameter yang lebih sedikit dari LSTM, sehingga cocok untuk data yang sedikit, agar tidak terjadi *overfitting*. Selain itu, GRU memberikan konvergensi yang lebih cepat dan hasilnya bisa disandingkan dengan LSTM (Chung *et al.*, 2014).

Persamaan variabel dari LSTM yaitu  $x_t$  sebagai input vektor ke LSTM,  $F_t$  sebagai vektor aktivasi *forget gate*,  $I_t$  sebagai vektor aktivasi *input gate*,  $O_t$  sebagai vektor aktivasi *output gate*,  $H_t$  sebagai vektor *hidden state* atau dikenal sebagai vektor keluaran dari unit LSTM,  $C_t$  sebagai vektor *cell state*,  $W$ - $U$ - $b$  sebagai matriks bobot dan parameter vektor bias yang perlu dipelajari selama pelatihan. Bentuk persamaan LSTM yang ringkas adalah sebagai berikut.

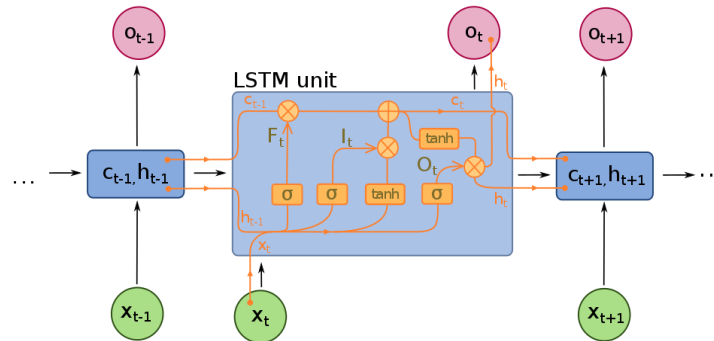
$$F_t = \sigma_g (W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

$$I_t = \sigma_g (W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

$$O_t = \sigma_g (W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$C_t = f_t \circ c_{t-1} + i_t \circ \sigma_g (W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

$$H_t = O_t \circ \sigma_h (c_t) \quad (5)$$



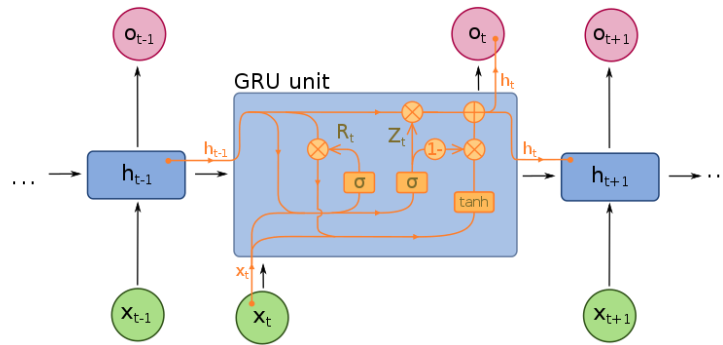
Gambar 3.4. Unit LSTM (François Deloche via Wikimedia Commons, 2017)

Persamaan variabel dari GRU yaitu  $x_t$  sebagai input vektor ke GRU,  $Z_t$  sebagai vektor *update gate*,  $R_t$  sebagai vektor *reset gate*,  $O_t$  sebagai vektor aktivasi *output gate*,  $H_t$  sebagai vektor *hidden state* atau dikenal sebagai vektor keluaran dari unit LSTM,  $W-U-b$  sebagai matriks bobot dan parameter vektor bias yang perlu dipelajari selama pelatihan. Bentuk persamaan GRU yang ringkas adalah sebagai berikut.

$$F_t = \sigma_g (W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$R_t = \sigma_g (W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$C_t = Z_t \circ h_{t-1} + (1-Z_t) \circ \sigma_g (W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \quad (3)$$



Gambar 3.5. Unit GRU (François Deloche via Wikimedia Commons, 2017)

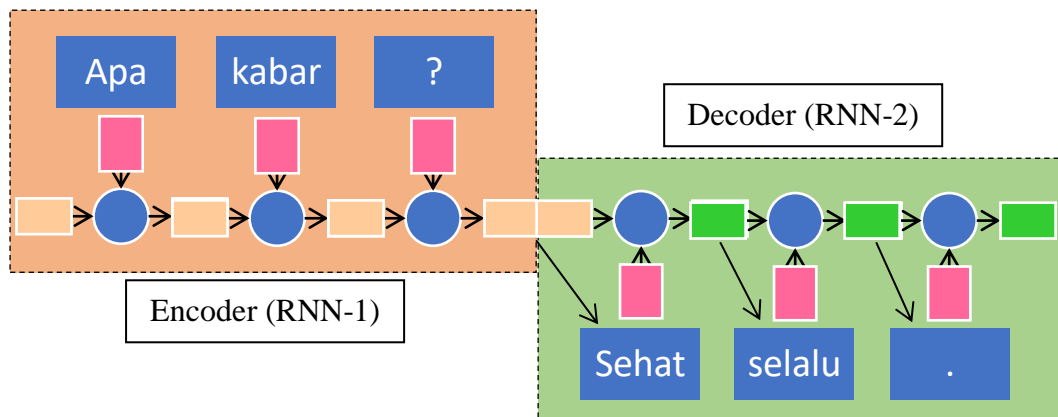
### 3.6. Sequential to Sequential (SEQ2SEQ)

*Recurrent Neural Network* (RNN) sejauh ini berguna untuk pemodelan informasi yang sekuensial seperti pada tugas-tugas bahasa alami. Model RNN yang populer saat ini adalah *sequential-to-sequential* (SEQ2SEQ). Model SEQ2SEQ dibentuk dari dua jaringan RNN, yaitu *encoder* dan *decoder*. Jaringan RNN pada *encoder* melakukan *encoding* untuk sekuens input ke dalam vektor konteks dengan ukuran tetap, sedangkan *decoder* menggunakan vektor konteks tadi sebagai "benih" untuk menghasilkan sekuens target. Oleh karena itu, model SEQ2SEQ sering juga disebut sebagai model *encoder-decoder*.

Aplikasi di bidang bahasa alami mulai berkembang karena kemampuan ingatan RNN untuk konteks jangka panjang. Model percakapan SEQ2SEQ mampu memprediksi kata berikutnya dengan diberikan kata atau kalimat konteks sebelumnya dari suatu percakapan. Hal itu akan menghasilkan model

generatif yang merangkai kata per kata dan membentuk kalimat yang benar-benar baru, dibandingkan dengan model *retrieval* yang hanya mengambil kalimat utuh pada pra-konstruksi repositori.

Penelitian arsitektur SEQ2SEQ mengalami kemajuan untuk menghasilkan model dengan akurasi yang tinggi. Perkembangan model SEQ2SEQ ada yang memakai gabungan dua gerbang yang berbeda (LSTM dan GRU) pada *encoder* model (Bay *et al.*, 2017). Ada juga yang memakai mekanisme gerbang yang *multilayer*, contohnya 2 layer LSTM pada *encoder* dan 2 layer LSTM pada *decoder*. Arsitektur sederhana SEQ2SEQ bisa dilihat pada ilustrasi Gambar 3.6.



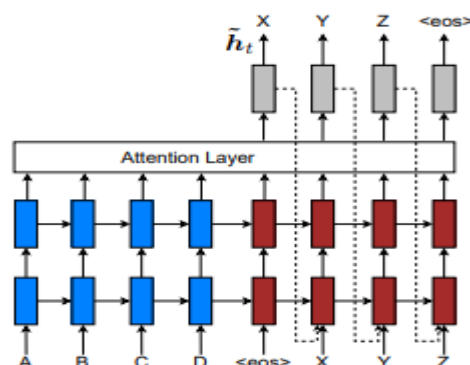
Gambar 3.6. Arsitektur SEQ2SEQ (CIS530 University of Pennsylvania, 2018)

### 3.7. Mekanisme Perhatian

Konsep perhatian telah mendapatkan popularitas baru-baru ini dalam pelatihan jaringan saraf dengan mengambil contoh dari mekanisme perhatian visual manusia. Perhatian visual manusia pada dasarnya memberi fokus pada wilayah tertentu dari gambar dengan "resolusi tinggi" sambil melihat gambar di sekitarnya dalam "resolusi rendah", dan kemudian menyesuaikan titik fokus itu dari waktu ke waktu. Mekanisme perhatian kemudian berkembang dan memungkinkan model untuk belajar keberpihakan antara modalitas yang berbeda (Luong *et al.*, 2015), misalnya, antara visual gambar dan keterangan gambar pada pembuatan *image caption*.

Pada tugas bahasa alami, input dikodekan ke dalam ukuran tetap, sehingga kalimat yang panjang tidak dapat digeneralisasi jauh lebih panjang daripada kapasitas tetapnya. Bahdanau (2015) lalu menerapkan mekanisme perhatian untuk menyelaraskan kata-kata pada tugas terjemahan mesin sehingga mengatasi kinerja buruk pada kalimat yang panjang. Mekanisme perhatian mampu memperluas vektor ukuran tetap itu dengan memungkinkan model secara otomatis mencari bagian dari sumber kalimat yang relevan untuk memprediksi kata target, tanpa harus membentuk bagian-bagian itu sebagai segmen padat secara eksplisit.

Mekanisme perhatian membantu jaringan mengingat aspek-aspek tertentu dari input dengan lebih baik. Mekanisme perhatian digunakan ketika menghasilkan setiap kata dalam *decoder* (Lopyrev, 2015). Untuk setiap kata keluaran, mekanisme perhatian menghitung bobot dari setiap kata input yang menentukan seberapa banyak perhatian yang harus diberikan pada kata input itu. Penjumlahan bobot maksimum adalah satu ( $\leq 1$ ) yang kemudian digunakan untuk menghitung rata-rata tertimbang dari layer tersembunyi terakhir yang dihasilkan setelah memproses setiap kata input. Rata-rata bobot ini, disebut sebagai konteks, kemudian dimasukkan ke layer *softmax* bersama dengan layer tersembunyi terakhir dari langkah saat ini dari proses *decoding*. Lapisan mekanisme perhatian (*attention layer*) pada jaringan SEQ2SEQ bisa dilihat pada ilustrasi Gambar 3.7.



Gambar 3.7. *Attention Layer* Pada Jaringan SEQ2SEQ (Luong *et al.*, 2015)

### 3.8. Gradient Descent

*Gradient Descent* (GD) adalah metode pengoptimalan yang sederhana dan populer dalam *deep learning*. Model *deep learning* biasanya memiliki parameter (bobot dan bias) dan fungsi biaya (*error*) untuk mengevaluasi seberapa baik set dari parameter tersebut. Ide dasar GD adalah menyesuaikan bobot model dengan mencari turunan dari fungsi biaya (*error*) sehubungan dengan masing-masing anggota matriks bobot di dalam model. GD memulai dari nilai awal parameter tersebut dan secara iteratif bergerak menuju nilai parameter yang meminimalisasi nilai dari fungsi biaya (*error*) tersebut.

Ada beberapa ekstensi dari metode pengoptimalan GD. Contoh ekstensi tersebut adalah *Stochastic Gradient Descent* (SGD), *Adaptive Gradient Algorithm* (AdaGrad), *Root Mean Square Propagation* (RMSProp), dan *Adaptive Moment Estimation* (Adam). SGD mempertahankan tingkat pembelajaran tunggal (*alpha*) untuk semua pembaruan bobot dan tingkat pembelajaran tidak berubah selama pelatihan. AdaGrad mempertahankan tingkat pembelajaran per-parameter sehingga meningkatkan kinerja pada masalah dengan gradien yang tersebar. RMSProp juga mempertahankan laju pembelajaran per-parameter yang disesuaikan berdasarkan rata-rata besarnya bobot gradien terbaru, seperti seberapa cepat bobot itu berubah, sehingga RMSProp baik digunakan untuk masalah pembaruan bobot terbaik secara sekuensial (online) dan non-stasioner. Adam lalu menggabungkan keuntungan dari dua ekstensi itu untuk menghitung laju pembelajaran individu yang adaptif untuk parameter yang berbeda dari perkiraan momen pertama seperti RMSProp, dan momen kedua gradien (Kingma *et al.*, 2014). Pada praktiknya, Adam saat ini direkomendasikan sebagai algoritma yang digunakan secara *default*, dan seringkali bekerja dengan lebih baik daripada RMSProp dan AdaGrad.

### 3.9. Framework Deep Learning

Pelatihan RNN membutuhkan kemampuan komputasi yang tinggi dan banyak implementasi *deep learning* memakai *General Purpose Graphics*

*Processing Unit* (GPGPU) untuk memanfaatkan kemampuan komputasi paralel supaya proses komputasi *deep learning* menjadi lebih cepat. Kerangka kerja *deep learning* menawarkan blok-blok pembangunan untuk merancang, melatih, dan memvalidasi jaringan melalui antarmuka pemrograman tingkat tinggi. Hal itu dapat dilakukan dengan memakai sebuah *library* yang dapat mendukung beragam jenis perangkat keras yang berbeda. Misalnya, suatu program yang dibangun dengan Keras diatas kerangka kerja Tensorflow, bisa berjalan pada CPU maupun GPU tanpa perlu mengubah kode program itu.

Saat penelitian ini berlangsung, sudah ada beberapa kerangka kerja *deep learning* yang populer serta didukung oleh komunitas dari pengembang dan perusahaan teknologi, seperti Theano, Tensorflow, Torch, Microsoft Cognitive Toolkit, Neon, Chainer, dan lain sebagainya. Kerangka kerja tersebut menyediakan fitur yang serupa dengan menawarkan dukungan luas untuk algoritma *deep learning*, memiliki banyak panduan dan dokumentasi, dan bahkan menawarkan visualisasi dari proses pelatihan model. Pada akhirnya kerangka kerja tersebut mempunyai tujuan umum, yaitu untuk manajemen komputasi yang intens dimana melibatkan proses iterasi yang cukup banyak dengan jumlah data yang besar.

### **3.10. Tensorflow**

Tensorflow merupakan kerangka kerja *machine learning* yang dibangun secara terbuka (*opensource*) yang didukung oleh perusahaan teknologi Google. Secara umum Tensorflow adalah antarmuka pemrograman untuk perhitungan numerik memakai grafik aliran data. Node-node yang ada di dalam grafik mewakili operasi matematika, sedangkan tepi grafik mewakili array data multidimensi (*tensor*) yang mengalir di antara mereka. Komputasinya dapat dijalankan dengan sedikit atau tanpa perubahan pada beragam sistem yang heterogen, mulai dari perangkat seluler hingga perangkat komputasi seperti kartu GPU. Sistem Tensorflow itu fleksibel dan dapat digunakan untuk mengekspresikan berbagai macam algoritma, termasuk pelatihan dan inferensi untuk model *deep learning* (Abadi *et al.*, 2016) dan telah digunakan untuk



berbagai bidang penelitian, seperti pengenalan suara, visi komputer, robot, pengambilan informasi, pemrosesan bahasa alami, ekstraksi informasi geografis, dan penemuan obat-obatan secara komputasional. Oleh karena itu, Tensorflow termasuk sebagai *library* yang paling populer dengan jumlah pengembang dan dukungan komunitas yang besar.

## BAB IV

### Analisis dan Perancangan Model Optimasi

Pada bab ini akan dibahas mengenai analisis dan perancangan optimasi model yang akan dibuat dalam penelitian ini, yaitu analisis algoritma, alur kerja model secara umum, perancangan model optimasi yaitu kombinasi mekanisme gerbang dengan penambahan mekanisme perhatian dan *gradient descent*, pemilihan kerangka kerja *deep learning*, perancangan algoritma dan *pseudocode*, perancangan sampel data untuk pelatihan dan validasi, serta perancangan pengujian model.

#### 4.1. Analisis Algoritma RNN Pada Model SEQ2SEQ

Model SEQ2SEQ dibentuk dari dua jaringan RNN, yaitu sebagai *encoder* dan *decoder*. RNN memberikan manfaat untuk mengolah data yang sekuens dan memori jangka panjang. Akan tetapi, RNN dianggap gagal menjadi alat utama pembelajaran, karena sulit melatihnya secara efektif. Ada beberapa tantangan untuk melatih model dengan algoritma RNN dan *gradient descent*, seperti: (1) terdapat kemungkinan nilai gradiennya mengecil karena perkalian matriks yang berulang kali dengan bobot yang kecil saat propagasi balik; (2) gradien bisa membesar dan meledak karena hasil perkalian bobotnya menjadi lebih besar dari batas norma gradien (Salehinejad *et al.*, 2017). Hal itu terkenal sebagai masalah hilang atau meledaknya gradien (*vanishing / exploding gradient*). Oleh karena itu, akhir-akhir ini pelatihan dengan algoritma RNN lebih berfokus kepada upaya untuk mengurangi masalah hilang atau meledaknya gradien melalui langkah optimasi.

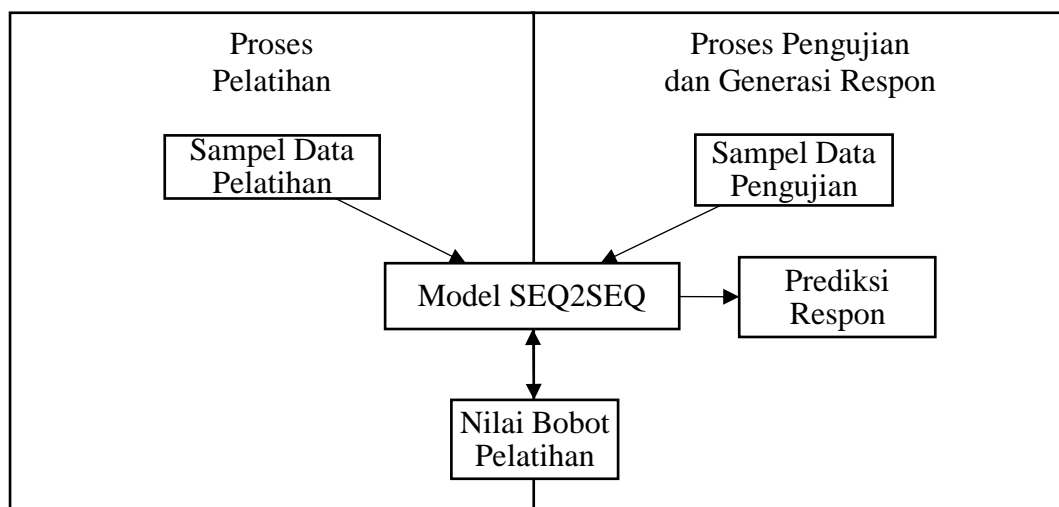
Optimasi algoritma RNN pada model SEQ2SEQ dapat dilakukan dengan memasukkan unit gerbang yang dirancang khusus untuk menyimpan informasi dengan waktu periode yang lama. Unit gerbang tersebut adalah *Long-Short Term Memory* (LSTM) (Sutskever *et al.*, 2014; Vinyals *et al.*, 2015) dan *Gated Recurrent Unit* (GRU) (Cho *et al.*, 2014). Unit gerbang tersebut akan mengubah struktur internal RNN yang hanya memiliki fungsi aktivasi *Tanh*

menjadi struktur internal yang memiliki gerbang. Unit gerbang GRU disusun atas dua gerbang: *reset* dan *update*; sedangkan LSTM disusun atas tiga gerbang yaitu *input*, *output*, dan *forget* serta terdapat unit memori yang dikontrol oleh gerbang tersebut (Chung *et al.*, 2014).

Penelitian ini memakai kombinasi unit gerbang LSTM dan GRU. Unit gerbang GRU diketahui konvergensinya lebih cepat dan LSTM memiliki unit memori untuk mengingat informasi yang jaraknya jauh dan mempertahankan pemahaman informasi secara lebih lama. Penelitian ini juga menambahkan mekanisme perhatian (*Attention Mechanism*) untuk mengatasi masalah sekuens yang panjang dan *gradient descent* untuk menghasilkan model dengan tingkat kesalahan minimal.

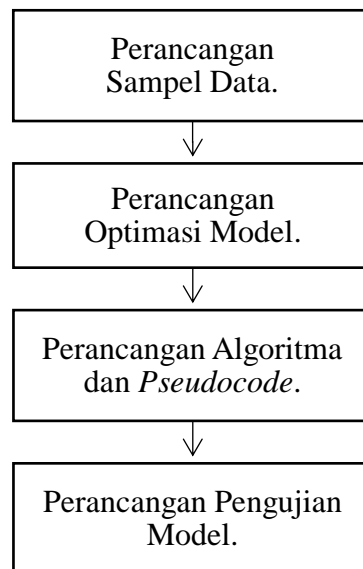
#### 4.2. Alur Pengerjaan Model Optimasi

Model yang akan dibuat memiliki dua tahap pengerjaan, yaitu proses pelatihan serta pengujian dan generasi respon. Proses pelatihan menghasilkan bobot pelatihan dan dipakai untuk memprediksi kalimat respon berdasarkan input kalimat yang dimasukkan. Proses pelatihan serta pengujian dan generasi respon berjalan secara terpisah, namun masih memakai model yang sama, yang mana dapat memproses kedua bagian secara berbeda berdasarkan parameter yang diberikan. Hal tersebut digambarkan pada Gambar 4.1.



Gambar 4.1. Ilustrasi Proses Pelatihan serta Pengujian dan Generasi Respon

Alur pengerjaan model percakapan SEQ2SEQ ini terdiri dari perancangan sampel data, perancangan dan pseudocode optimasi model SEQ2SEQ, dan perancangan pengujian model SEQ2SEQ.



Gambar 4.2. Alur Pengerjaan Model Percakapan

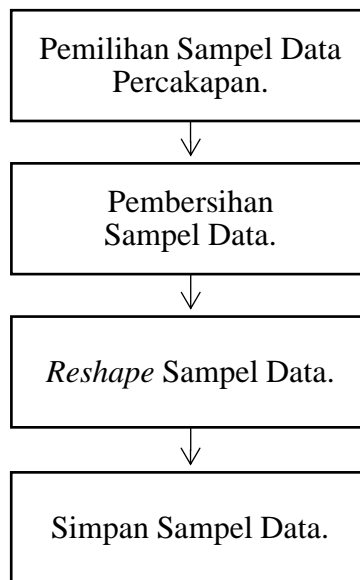
#### 4.3. Perancangan Sampel Data

Penelitian ini memilih *dataset OpenSubtitle 2018* Bahasa Indonesia yang mengandung 137,2 juta token dan 22,8 juta kalimat (Lison *et al.*, 2018). *Dataset OpenSubtitle* berisi kalimat yang diucapkan oleh banyak aktor dan mengingat ruang lingkupnya yang luas dengan berbagai *genre* dan judul film, *OpenSubtitle* termasuk jenis *dataset* percakapan pada domain terbuka (Vinyals *et al.*, 2015).

Untuk memenuhi tujuan penelitian dengan spesifikasi komputer yang ada, penelitian ini memakai 5-20 ribu baris percakapan dari *OpenSubtitle 2018* sebagai sampel data. *OpenSubtitle* adalah *dataset* yang sangat berderau (*noise*) (Vinyals *et al.*, 2015) sehingga pemakaian sampel data akan menghindari *overfitting* karena mencoba memasang data yang berderau itu untuk dilatih ke dalam jaringan.

Penelitian ini menerapkan langkah pemrosesan sederhana pada sampel data dengan menghapus kata yang bukan teks percakapan dan mengubah beberapa kata ke dalam bentuk yang baku. Giliran percakapan aktor pada *OpenSubtitle 2018* tidak ditunjukkan dengan jelas, sehingga setiap kalimat diasumsikan akan diucapkan oleh

aktor yang berbeda. Setiap baris data adalah pasangan kalimat konteks dan target, sehingga perlu mengubah bentuk (*reshape*) sampel data menjadi berpasangan, maksimal 15 kata untuk setiap konteks dan target. Sampel data lalu bisa disimpan untuk proses pelatihan. Perancangan sampel data dapat dilihat pada Gambar 4.3.



Gambar 4.3. Alur Pengerjaan Sampel Data

Model percakapan memprediksi kalimat berikutnya dengan diberikan kalimat sebelumnya, berlaku untuk setiap baris percakapan. Langkah itu membuat setiap baris percakapan akan dipakai sebagai konteks dan target. Total pembagian pelatihan dan pengujian penelitian ini sebesar 80% sampel data sebagai set pelatihan dan 20% sampel data sebagai set pengujian. Pembagian ini dilakukan sedemikian rupa sehingga setiap kalimat dalam pasangan kalimat (konteks-target) muncul bersama di set pelatihan atau di set pengujian tetapi tidak di keduanya.

#### 4.4. Perancangan Model Optimasi

Model *deep learning* yang digunakan dalam penelitian ini adalah SEQ2SEQ dimana *encoder* akan menangkap esensi dari input dan *decoder* menghasilkan output dengan melihat kata dan konteks sebelumnya dalam setiap langkah waktu. Optimasi dilakukan dengan kombinasi mekanisme gerbang LSTM atau GRU pada *encoder* dan *decoder* model. LSTM diketahui bagus untuk mengingat informasi

dalam waktu yang lebih lama, sedangkan GRU lebih baik dalam hal kecepatan konvergensi pelatihan. Mekanisme perhatian digunakan karena terbukti mengatasi masalah sekuens yang panjang dan meningkatkan kinerja model menjadi lebih baik.

Vinyals (2015) mengaplikasikan model percakapan SEQ2SEQ dengan layer LSTM baik pada *encoder* dan *decoder* dengan *gradient descent*, namun tanpa mekanisme perhatian. Pada penelitian ini akan memperlihatkan perbandingan variasi penggunaan unit gerbang pada *encoder* dan *decoder* dengan penambahan mekanisme perhatian dan *gradient descent*. Variasi model gerbang tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1. Tabel Variasi Optimasi Model SEQ2SEQ

<b>Nama Model</b>	<b><i>Encoder</i></b>	<b><i>Decoder</i></b>	<b>Unit Perhatian</b>	<b><i>Gradient Descent</i></b>
LSTM-LSTM	LSTM	LSTM	Bahdanau (2014)	Adam
LSTM-GRU	LSTM	GRU	Bahdanau (2014)	Adam
GRU-GRU	GRU	GRU	Bahdanau (2014)	Adam
GRU-LSTM	GRU	LSTM	Bahdanau (2014)	Adam

#### 4.4.1. Model Gerbang LSTM-LSTM

Model ini memiliki total 6 layer pada jaringannya sesuai dengan Tabel 4.2. Layer *embedding* dipakai untuk mengubah besarnya jumlah kata (*vocabulary*) menjadi vektor padat dengan ukuran tetap. Kombinasi gerbang model ini adalah 1 layer LSTM pada *encoder* dan 1 layer LSTM pada *decoder* serta menerapkan fungsi *dropout* agar menghindari terjadinya *overfitting* pada jaringan. Keduanya dihubungkan oleh layer *Repeat Vector* agar bisa mengulangi input dari *encoder* ke *decoder*. Layer *Batch Normalization* digunakan agar distribusi dari input ke layer tertentu tidak berubah dari waktu ke waktu akibat pembaruan parameter dari setiap *batch*. Layer *Attention* memakai fungsi aktivasi *softmax*, *time distributed*, dan *dense*. Kemudian layer *Attention* dipasang diakhir layer, dekat dengan layer *decoder* sesuai dengan pekerjaan Bahdanau (2014).

Tabel 4.2. Tabel Struktur *Layer* Model LSTM-LSTM

<i>Layer</i> *	Ukuran unit	Jumlah Layer
<i>Embedding</i>	$1 \times N$ <i>Vocabulary</i>	1
<i>Encoder LSTM</i>	$1 \times 256$	1
<i>Repeat Vector</i>	$1 \times 256$	1
<i>Decoder LSTM</i>	$1 \times 256$	1
<i>Batch Normalization</i>	$1 \times 256$	1
<i>Attention</i>	$1 \times 256$	1
<b>Total Layer :</b>		6

\*Menunjukkan urutan hirarki *layer* dari paling awal hingga akhir.

#### 4.4.2. Model Gerbang LSTM-GRU

Model ini memiliki total 6 layer pada jaringannya sesuai dengan Tabel 4.3. Layer *embedding* dipakai untuk mengubah besarnya jumlah kata (*vocabulary*) menjadi vektor padat dengan ukuran tetap. Kombinasi gerbang model ini adalah masing-masing 1 layer LSTM pada *encoder* dan 1 layer GRU pada *decoder* serta menerapkan fungsi *dropout* agar menghindari terjadinya *overfitting* pada jaringan. Kedua arsitektur layer yang berbeda itu dihubungkan oleh layer *Repeat Vector* supaya bisa mengulangi input dari *encoder* ke *decoder*. Layer *Batch Normalization* digunakan agar distribusi dari input ke layer tertentu tidak berubah dari waktu ke waktu akibat pembaruan parameter dari setiap *batch*. Layer *Attention* memakai fungsi aktivasi *softmax*, *time distributed*, dan *dense*. Kemudian layer *Attention* dipasang diakhir layer, dekat dengan layer *decoder* sesuai dengan pekerjaan Bahdanau (2014).

Tabel 4.3. Tabel Struktur *Layer* Model LSTM-GRU

<i>Layer</i> *	Ukuran unit	Jumlah Layer
<i>Embedding</i>	$1 \times N$ <i>Vocabulary</i>	1
<i>Encoder LSTM</i>	$1 \times 256$	1

<i>Repeat Vector</i>	1 x 256	1
<i>Decoder GRU</i>	1 x 256	1
<i>Batch Normalization</i>	1 x 256	1
<i>Attention</i>	1 x 256	1
<b>Total Layer :</b>		6

\*Menunjukkan urutan hirarki *layer* dari paling awal hingga akhir.

#### 4.4.3. Model Gerbang GRU-GRU

Model ini memiliki total 6 layer pada jaringannya sesuai dengan Tabel 4.4. Layer *embedding* dipakai untuk mengubah besarnya jumlah kata (*vocabulary*) menjadi vektor padat dengan ukuran tetap. Kombinasi gerbang model ini adalah masing-masing 1 layer GRU pada *encoder* dan 1 layer GRU pada *decoder* serta menerapkan fungsi *dropout* agar menghindari terjadinya *overfitting* pada jaringan. Keduanya dihubungkan oleh layer *Repeat Vector* agar bisa mengulangi input dari *encoder* ke *decoder*. Layer *Batch Normalization* digunakan agar distribusi dari input ke layer tertentu tidak berubah dari waktu ke waktu akibat pembaruan parameter dari setiap *batch*. Layer *Attention* memakai fungsi aktivasi *softmax*, *time distributed*, dan *dense*. Kemudian layer *Attention* dipasang diakhir layer, dekat dengan layer *decoder* sesuai dengan pekerjaan Bahdanau (2014).

Tabel 4.4. Tabel Struktur *Layer* Model GRU-GRU

<i>Layer</i> *	Ukuran unit	Jumlah Layer
<i>Embedding</i>	1 x $N$ <i>Vocabulary</i>	1
<i>Encoder GRU</i>	1 x 256	1
<i>Repeat Vector</i>	1 x 256	1
<i>Decoder GRU</i>	1 x 256	1
<i>Batch Normalization</i>	1 x 256	1
<i>Attention</i>	1 x 256	1
<b>Total Layer :</b>		6

\*Menunjukkan urutan hirarki *layer* dari paling awal hingga akhir.



#### 4.4.4. Model Gerbang GRU-LSTM

Model ini memiliki total 6 layer pada jaringannya sesuai dengan Tabel 4.5. Layer *embedding* dipakai untuk mengubah besarnya jumlah kata (*vocabulary*) menjadi vektor padat dengan ukuran tetap. Kombinasi gerbang model ini adalah masing-masing 1 layer GRU pada *encoder* dan 1 layer LSTM pada *decoder* serta menerapkan fungsi *dropout* agar menghindari terjadinya *overfitting* pada jaringan. Kedua arsitektur layer yang berbeda itu dihubungkan oleh layer *Repeat Vector* supaya bisa mengulangi input dari *encoder* ke *decoder*. Layer *Batch Normalization* digunakan agar distribusi dari input ke layer tertentu tidak berubah dari waktu ke waktu akibat pembaruan parameter dari setiap *batch*. Layer *Attention* memakai fungsi aktivasi *softmax*, *time distributed*, dan *dense*. Kemudian layer *Attention* dipasang diakhir layer, dekat dengan layer *decoder* sesuai dengan pekerjaan Bahdanau (2014).

Tabel 4.5. Tabel Struktur *Layer* Model GRU-LSTM

<i>Layer</i> *	Ukuran unit	Jumlah Layer
<i>Embedding</i>	$1 \times N$ <i>Vocabulary</i>	1
<i>Encoder GRU</i>	$1 \times 256$	1
<i>Repeat Vector</i>	$1 \times 256$	1
<i>Decoder LSTM</i>	$1 \times 256$	1
<i>Batch Normalization</i>	$1 \times 256$	1
<i>Attention</i>	$1 \times 256$	1
<b>Total Layer :</b>		6

\*Menunjukkan urutan hirarki *layer* dari paling awal hingga akhir.

#### 4.5. Perancangan Algoritma dan *Pseudocode*

##### 4.5.1. Pelatihan Model

Pelatihan model dimulai dari pemuatan sampel data yang sudah dibersihkan, pembagian sampel data, tokenisasi sampel data, penyandian kata

dalam bentuk yang sekuens, pendefinisian model jaringan, lalu sekuens kata dilatih secara optimal untuk menghasilkan bobot jaringan yang dipakai pada saat generasi respon percakapan. Pada saat pelatihan, terjadi perhitungan skor dari metrik *accuracy*, *precision*, *recall*, *F1*, dan nilai *loss* dari bobot jaringan. Nilai itu kemudian disimpan untuk mengetahui seberapa bagus bobot jaringan yang dihasilkan pada saat pelatihan. Berikut adalah *pseudocode* dari proses pelatihan model secara umum.

```
Pemuatan sampel data yang telah dibersihkan.
Pembagian sampel data menjadi variabel pelatihan (80%) dan tes (20%).
Tokenisasi seluruh sampel data menjadi vocabulary. Kemudian hitung panjang total vocabulary dan panjang kata maksimum pada vocabulary.
Proses penyandian variabel pelatihan dan tes ke dalam bentuk yang sekuens (encode sequences).
Pemuatan model jaringan dengan layer embedding, encoder, repeat vector, decoder, batch normalization, dan attention.
Pemuatan model jaringan dengan gradient descent (Adam), tipe loss (categorical crossentropy), dan metrik (accuracy, precision, recall, F1).
Inisialisasi nilai parameter epoch dan batch size.
Persiapan variabel pelatihan dan tes yang diproses (feed) ke dalam jaringan.

For epoch dari 1 sampai N
    Proses feed variabel pelatihan dan tes serta nilai parameter ke dalam jaringan.
    Perolehan nilai metrik dan loss.
    Perbaruan bobot jaringan.
End For

Simpan nilai bobot jaringan yang sudah terbentuk.
Simpan nilai metrik untuk pelatihan dan tes dari bobot jaringan yang sudah terbentuk.
```

Kode 4.1. *Pseudocode* Proses Pelatihan

#### 4.5.2. Generasi Respon

Proses generasi respon percakapan dimulai dari pemuatan sampel data yang sudah dibersihkan, tokenisasi sampel data, pemuatan bobot jaringan dari suatu model, input kalimat percakapan dan penyandian input dalam bentuk yang sekuens, proses translasi untuk menghasilkan indeks kata, lalu indeks itu digunakan untuk menemukan kata yang ada pada hasil

tokenisasi sampel data. Bila indeks kata sesuai dengan indeks yang ada di token, maka kata itu berhasil ditemukan dan akan ditambahkan ke dalam variabel kalimat generasi untuk menampung seluruh kata yang berhasil ditemukan. Proses generasi respon berakhir ketika sudah didapatkan seluruh kata pada indeks. Terakhir, program akan menampilkan kalimat keluaran sesuai dengan isi variabel kalimat generasi. Berikut adalah rancangan kode atau *pseudocode* dari proses generasi itu secara umum.

```
Pemuatan sampel data yang telah dibersihkan.
Tokenisasi seluruh sampel data menjadi vocabulary. Kemudian
hitung panjang total vocabulary dan panjang kata maksimum pada
vocabulary.
Pemuatan bobot jaringan suatu model dari hasil pelatihan.
Input kalimat ke dalam sistem percakapan.
Perolehan input dan penyandian input ke dalam bentuk sekuens
(encode sequences).
Proses translasi yaitu memprediksi sekuens input dengan model
yang telah dimuat. Hasil prediksi berupa kumpulan indeks.

Persiapan mapping indeks ke dalam bentuk kata.
Foreach indeks.
    Pencarian indeks yang sama pada hasil tokenisasi sampel.
    If indeks ditemukan.
        Tambahkan token kata ke variabel kalimat generasi.

Tampilkan kalimat dari variabel kalimat generasi ke sistem
percakapan.
```

Kode 4.2. *Pseudocode* Proses Generasi Respon

#### 4.6. Perancangan Pengujian Model

Model percakapan diuji dengan memakai sampel data tes dari *OpenSubtitle 2018*. Objek penilaian dalam pengujian ini adalah skor dari metrik *accuracy*, *precision*, *recall*, *F1*, dan *loss* untuk tiap kelas model percakapan. Akurasi generasi kalimat itu dilihat dari seberapa benar pemetaan kalimat konteks dengan kalimat target percakapan.

Selain *accuracy*, pengujian juga memperhatikan skor dari *precision* dan *recall*. *Precision* menunjukkan seberapa dekat atau tidak tersebarnya kumpulan konteks setelah dipetakan ke target kalimat. Itu menunjukkan ketepatan antara informasi yang tersedia dengan jawaban yang diberikan. *Recall* menunjukkan

seberapa lengkap sistem dalam menemukan kembali hasil prediksi pada nilai yang positif saja. *Precision* dan *recall* kemudian didefinisikan sebagai berikut.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

<i>TP</i>	=	<i>True Positive</i>
<i>TN</i>	=	<i>True Negative</i>
<i>FP</i>	=	<i>False Positive</i>
<i>FN</i>	=	<i>False Negative</i>

*F-measure (F1)* adalah metrik evaluasi dalam informasi temu kembali yang mengkombinasikan *precision* dan *recall* karena pada suatu waktu nilai keduanya dapat memiliki bobot yang berbeda. Maka dari itu, *F1* adalah rata-rata harmonik dari *precision* dan *recall* dengan rentangan nilai antara 0 sampai 1. *F1* kemudian didefinisikan sebagai berikut.

$$\textbf{F1} = 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$

#### 4.7. Framework Deep Learning

Berbagai macam kerangka kerja (*framework*) *deep learning* telah tersedia untuk menyediakan fungsi-fungsi umum pada proses komputasi *deep learning* seperti fungsi *Sequential*, *LSTM*, *GRU*, *Embedding*, dan fungsi-fungsi lainnya. Kerangka kerja yang tersedia saat ini mempunyai komunitas pengguna dan pengembang serta memiliki keunggulaannya masing-masing. Penelitian ini memilih *framework* Tensorflow dalam pembuatan model percakapan SEQ2SEQ karena dukungan fitur yang cukup lengkap, tutorial dan dokumentasi yang baik, serta pembaharuan program secara berkala dari pengembang resminya atau dari komunitasnya yang besar. Untuk meminimalkan kode program, penelitian ini memakai Keras. Keras mampu berjalan diatas Tensorflow yang berfokus untuk penggunaan kode program yang sedikit, modular, dan dapat diperluas sehingga ramah untuk dipakai oleh *developer*.

## **BAB V**

### **Implementasi dan Evaluasi Model Optimasi**

Pada bab ini akan membahas mengenai hasil optimasi model dan pengujian dari model percakapan yang meliputi implementasi model optimasi, hasil pelatihan, hasil pengujian serta pembahasan mengenai kelebihan dan kekurangan sistem. Pembahasan hasil penelitian ini digunakan untuk mendeskripsikan hasil dari proses pembuatan dan optimasi model, serta untuk menganalisis apakah sistem yang dibuat sudah memenuhi tujuan yang ingin dicapai pada penelitian ini.

#### **5.1. Implementasi Model Optimasi**

Model jaringan dan fungsi-fungsi dalam penelitian ini akan diterapkan dalam bentuk kode program dengan menggunakan bahasa Python dan *framework* Tensorflow dan Keras. Berikut ini adalah implementasi kode program untuk tiap bagian fungsi-fungsi yang ada.

##### **5.1.1. Fungsi-Fungsi Umum**

###### **5.1.1.1. Fungsi *Preprocessing Text***

Fungsi *preprocessing text* digunakan untuk membersihkan teks percakapan pada sampel data yang didalamnya berisi pasangan kalimat konteks dan target. Pertama, fungsi ini mengubah seluruh teks menjadi huruf kecil. Fungsi ini kemudian menghilangkan seluruh huruf yang bukan huruf abjad dan tanda baca, mengubah beberapa kata ke dalam bentuk yang baku, lalu menghapus spasi yang berlebihan antar token kata. Fungsi ini juga melakukan pemotongan kata (*trimming*) jika suatu baris teks diketahui memiliki lebih dari 15 kata. Pemotongan ini dilakukan agar sekuens pada konteks dan target tidak terlalu panjang.

Sesudah langkah *preprocessing* tadi, maka teks yang sudah bersih itu dibalikkan untuk dipakai pada langkah selanjutnya.

```
def preprocessing_text(text):
    text = text.lower()
    text = re.sub(r'^a-zA-Z\s.,?!]', u'', text, flags=re.UNICODE)
    text = kata_baku(text)
    text = ' '.join(text.split())
    maxlen = 15
    if len(text.split()) > maxlen:
        text = (' ').join(text.split()[:maxlen])

    return text
```

Kode 5.1. Fungsi *Preprocessing Text*

Apa yang harus kulakukan ?	apa yang harus kulakukan ?
Aku harus konsentrasi belajar .	aku harus konsentrasi belajar .
Apa ?	apa ?
Su-eun yang disana itu , dia terus memandanguku .	sueun yang disana itu , dia terus memandanguku .
Aku perkiraan , dia akan mengajakku kencan minggu ini .	aku perkiraan , dia akan mengajakku kencan minggu ini .
Aku harus belajar .	aku harus belajar .
Ada apa dengan gadis zaman sekarang ?	ada apa dengan gadis zaman sekarang ?
Mereka punya standar yang tinggi .	mereka punya standar yang tinggi .
- Kau pergi duluan .	kamu pergi duluan .
- Tidak , kau yang duluan .	tidak , kamu yang duluan .
Aku ke kanan , dan kau ke kiri .	aku ke kanan , dan kamu ke kiri .

Gambar 5.1. Contoh Hasil Fungsi *Preprocessing Text*

### 5.1.1.2. Fungsi *Load Data Sample*

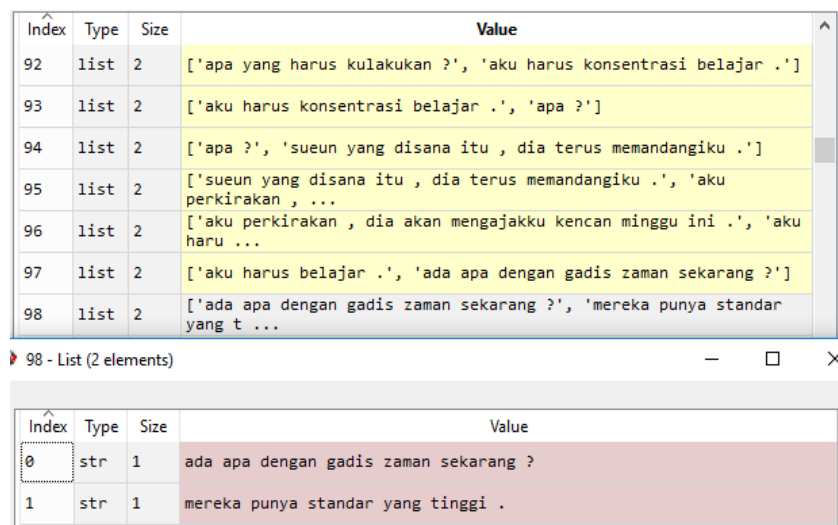
Fungsi *load data sample* digunakan untuk membaca sampel data yang sudah dibersihkan sebelumnya dari *OpenSubtitle 2018*. Sampel data itu disimpan dengan format *tab-separated values* (TSV). TSV memisahkan elemen kalimat konteks dan target dengan karakter ‘\t’ (tabs) sebagai *delimiter*. Di dalam fungsi tersebut terjadi pembacaan data TSV sesuai alamat direktori *default* dari tempat penyimpanan sampel data. Setelah itu, setiap baris pada sampel data itu disimpan ke dalam array penampung. Proses ini berakhir ketika seluruh sampel data berhasil dimuat dan array tersebut dibalikkan untuk dipakai pada langkah selanjutnya. Berikut potongan kode program untuk proses *load data sample*.

```

file = "clean-sample-data.csv"
folder = "OpenSubtitles2018"
def load_clean_sample_data():
    filepath = os.path.join(folder,file)
    with open(filepath, "r", encoding="utf8") as read:
        reader = csv.reader(read,delimiter="\t")
        dataset = []
        for row in reader:
            dataset.append(row)
    read.close()
    return dataset

```

Kode 5.2. Fungsi *Load Data Sample*



Index	Type	Size	Value
92	list	2	['apa yang harus kulakukan ?', 'aku harus konsentrasi belajar .']
93	list	2	['aku harus konsentrasi belajar .', 'apa ?']
94	list	2	['apa ?', 'sueun yang disana itu , dia terus memandangu .']
95	list	2	['sueun yang disana itu , dia terus memandangu .', 'aku perkiraan , ...']
96	list	2	['aku perkiraan , dia akan mengajakku kencan minggu ini .', 'aku haru ...']
97	list	2	['aku harus belajar .', 'ada apa dengan gadis zaman sekarang ?']
98	list	2	['ada apa dengan gadis zaman sekarang ?', 'mereka punya standar yang t ...']

98 - List (2 elements)

Index	Type	Size	Value
0	str	1	ada apa dengan gadis zaman sekarang ?
1	str	1	mereka punya standar yang tinggi .

Gambar 5.2. Contoh Hasil Fungsi *Load Data Sample*

### 5.1.1.3. Fungsi *Create Tokenizer*

Fungsi *create tokenizer* digunakan untuk membentuk token kata pada suatu baris kalimat. Proses tokenisasi kata ini nantinya dapat berguna untuk mendefinisikan kamus kata (*vocabulary*) dan besar panjangnya. Keduanya dipakai untuk proses pelatihan dan generasi respon. Berikut potongan kode program untuk fungsi *create tokenizer*.

```

import networks

def create_tokenizer(lines):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer

```

Kode 5.3. Fungsi *Create Tokenizer*

#### 5.1.1.4. Fungsi *Max Length*

Fungsi *max length* digunakan untuk menghitung panjang maksimal dari suatu baris kalimat. Fungsi ini menghitung jumlah kata yang paling maksimal dari setiap baris kalimat. Berikut potongan kode program untuk fungsi *max length*.

```
def max_length(lines):  
    return max(len(line.split()) for line in lines)
```

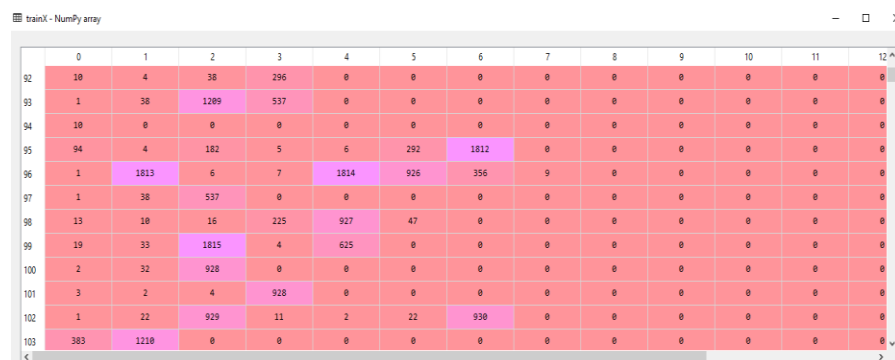
Kode 5.4. Fungsi *Max Length*

#### 5.1.1.5. Fungsi *Encode Sequences*

Fungsi *encode sequences* digunakan untuk menghasilkan daftar urutan integer yang menyandikan kata-kata sesuai kalimat yang diberikan. Perintah *text\_to\_sequences* digunakan untuk menyandikan kalimat ke dalam integer dalam bentuk yang sekuens. Setelah itu, perintah *pad\_sequences* digunakan untuk memastikan bahwa semua sekuens dalam daftar memiliki panjang yang sama sesuai dengan panjang maksimal yang telah ditentukan. Berikut potongan kode program untuk fungsi *encode sequences*.

```
def encode_sequences(tokenizer, length, lines):  
    X = tokenizer.texts_to_sequences(lines)  
    X = pad_sequences(X, maxlen=length, padding='post')  
    return X
```

Kode 5.5. Fungsi *Encode Sequences*



Gambar 5.3. Contoh Hasil Fungsi *Encode Sequences*

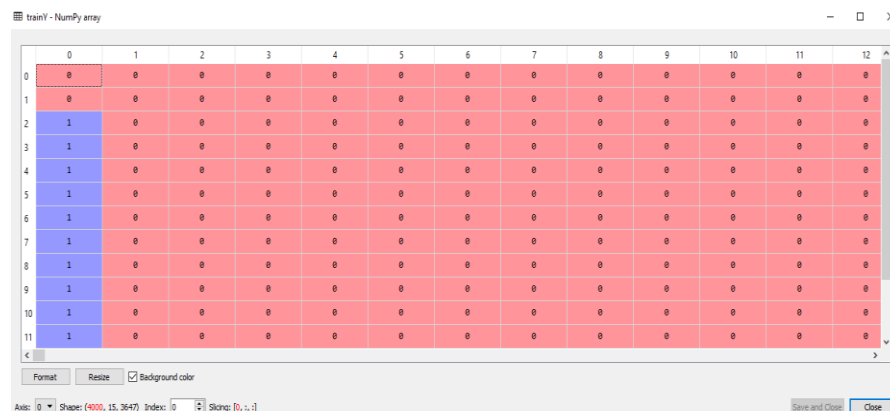


#### 5.1.1.6. Fungsi *Encode Output*

Fungsi *encode output* digunakan untuk mengubah vektor kelas (integer) dari sekuens target ke matriks kelas biner. Hal itu dilakukan untuk mengubah angka-angka itu ke dalam bentuk yang tepat untuk diproses ke dalam model. Berikut potongan kode program untuk fungsi *encode output*.

```
def encode_output(sequences, vocab_size):  
    ylist = list()  
    for sequence in sequences:  
        encoded = to_categorical(sequence, num_classes=vocab_size)  
        ylist.append(encoded)  
    y = array(ylist)  
    y = y.reshape(sequences.shape[0], sequences.shape[1],  
        vocab_size)  
    return y
```

Kode 5.6. Fungsi *Encode Output*



Gambar 5.4. Contoh Hasil Fungsi *Encode Output*

#### 5.1.1.7. Fungsi *Word For Id*

Fungsi *word for id* digunakan untuk memetakan suatu angka integer menjadi kata. Diberikan suatu angka integer dan hasil tokenisasi sampel data, kemudian index yang ada pada seluruh item token itu dibandingkan dengan angka integer yang telah diberikan. Jika index token sama dengan angka integer, artinya kata yang dicari itu ada pada item token. Fungsi ini membalikkan kata yang dicari untuk proses

selanjutnya. Bila tidak ditemukan, fungsi ini tidak membalikkan apa-apa. Berikut potongan kode program untuk fungsi *word\_for\_id*.

```
def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None
```

Kode 5.7. Fungsi *Word For Id*

#### 5.1.1.8. Fungsi *Predict Sequences*

Fungsi *predict sequences* digunakan untuk memprediksi dan menghasilkan kalimat target dengan diberikan konteks yang telah disandikan dalam bentuk sekuens. Pertama, fungsi ini akan membuat prediksi dari konteks sekuens. Prediksi akan dihasilkan oleh model yang telah dimuat sebelumnya, kemudian konteks sekuens itu dimasukkan ke dalam model sehingga menghasilkan prediksi berupa sekumpulan angka *float*. Setelah itu, fungsi akan mencari posisi nilai vektor terbesar dari angka prediksi tersebut. Posisi tersebut berbentuk sekumpulan angka *integer*. Untuk mengubah angka *integer* tersebut ke dalam bentuk kata, maka digunakan fungsi *word\_for\_id*. Fungsi *word\_for\_id* akan mencari angka integer yang sesuai dengan index di daftar token dengan perulangan. Kata yang ditemukan lalu disimpan ke variabel target. Perulangan akan berhenti bila tidak ada kata yang ditemukan. Terakhir, fungsi ini akan menggabungkan kata-kata itu untuk proses selanjutnya. Berikut potongan kode program untuk fungsi *predict sequences*.

```
def predict_sequence(model, tokenizer, source):
    prediction = model.predict(source, verbose=0)[0]
    integers = [argmax(vector) for vector in prediction]
    target = list()
    for i in integers:
        word = word_for_id(i, tokenizer)
        if word is None:
            break
        target.append(word)
    return ' '.join(target)
```

Kode 5.8. Fungsi *Predict Sequences*

prediksi - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	9.3573e-06	0.98844	0.00081172	1.78632e-07	0.00045447	0.000108277	4.22882e-05	5.18993e-06	2.02426e-05	2.01166e-05	0.00346866	8.63317e-07	3.82415e-4
1	5.28479e-05	1.46469e-05	5.40652e-06	0.000224239	5.65892e-06	1.2132e-07	1.25948e-07	0.995341	2.27815e-06	1.17456e-05	6.92745e-06	3.35412e-08	7.06177e-4
2	1.56711e-06	1.75506e-06	1.56236e-05	8.4483e-08	3.83028e-08	8.10549e-08	4.47061e-08	3.87431e-07	2.26502e-07	2.17181e-06	1.50987e-06	7.82411e-08	1.83536e-4
3	9.59515e-05	9.12211e-05	9.93398e-06	8.19421e-06	1.50718e-07	9.05612e-07	3.38679e-05	8.66428e-06	0.000230133	0.000585207	5.89067e-07	4.1824e-06	1.63827e-4
4	2.48278e-06	1.53187e-08	2.97008e-08	3.22797e-08	1.51963e-09	1.9056e-10	1.63000e-09	9.44389e-09	3.74638e-05	1.65600e-08	6.48469e-10	5.13064e-09	1.34702e-4
5	4.10458e-05	2.24645e-09	5.19502e-10	1.81145e-09	1.68913e-07	1.97966e-09	2.83448e-10	2.25501e-08	2.64562e-07	9.14685e-07	2.35447e-10	1.44895e-08	3.86762e-4
6	0.95789	3.40137e-05	9.40273e-08	7.96444e-05	7.79599e-05	0.000209316	1.2922e-05	9.03187e-05	9.59877e-05	0.00367301	3.40446e-07	6.77666e-06	5.11454e-4
7	0.980097	1.13369e-05	5.00657e-07	6.13837e-06	0.000000958	2.6045e-05	1.09355e-05	1.16583e-05	2.04204e-07	0.0111559	4.34427e-08	4.45908e-07	9.36957e-4
8	0.986173	2.33314e-06	2.42852e-07	2.58442e-06	1.93098e-05	2.28294e-05	4.52e-07	3.33422e-08	1.31282e-06	0.00771005	4.80444e-08	2.56034e-07	0.0001905
9	0.999852	4.79531e-08	1.38098e-07	8.0273e-08	1.25848e-07	6.00089e-07	4.85342e-10	1.69838e-08	5.98674e-07	3.89652e-05	1.06849e-10	7.17306e-09	2.25837e-4
10	0.99998	8.75984e-09	2.28404e-09	1.61351e-09	2.52415e-07	3.82849e-09	7.65119e-11	3.3018e-08	5.46235e-08	4.17763e-07	1.42707e-11	1.51252e-09	8.94998e-4
11	0.999995	4.4144e-09	2.48822e-09	2.06424e-10	3.78263e-07	6.40511e-10	2.01716e-11	3.5924e-09	4.67738e-09	1.00286e-06	5.29325e-12	3.68889e-10	5.01689e-4

Gambar 5.5. Contoh Angka Prediksi Dari Fungsi *Predict Sequences*

### 5.1.1.9. Fungsi *Translate*

Fungsi *translate* digunakan untuk melakukan translasi dari konteks yang berbentuk sekuens. Konteks yang diberikan itu berasal dari input kalimat ke program *chatbot*, dengan terlebih dahulu input tersebut disandikan ke dalam bentuk yang sekuens. Setiap konteks sekuens yang ada perlu diubah bentuknya ke dalam satu baris, baru sekuens tersebut diprediksi dengan fungsi *predict\_sequences* melalui perulangan. Proses translasi itu akan menghasilkan generasi kalimat respon untuk siap ditampilkan sebagai output program *chatbot*. Berikut potongan kode program untuk fungsi *translate*.

```
def translate(model, tokenizer, sources):
    for i, source in enumerate(sources):
        source = source.reshape(1, source.shape[0])
        translation = predict_sequence(model, all_tokenizer,
source)
    return translation
```

Kode 5.9. Fungsi *Translate*

### 5.1.1.10. Fungsi *Precision*

Fungsi *precision* digunakan untuk menghitung nilai metrik *precision* yang mencerminkan seberapa dekat prediksi dengan nilai aktual, khususnya seberapa konsisten hasil ketika pengukuran itu diulang. Berikut potongan kode program untuk fungsi *precision*.

```
def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives +
    K.epsilon())
    return precision
```

Kode 5.10. Fungsi *Precision*

#### 5.1.1.11. Fungsi *Recall*

Fungsi *recall* digunakan untuk menghitung nilai metrik *recall* yang menunjukkan seberapa lengkap sistem dalam menemukan kembali hasil prediksi pada nilai yang positif saja. Berikut potongan kode program untuk fungsi *recall*.

```
def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
```

Kode 5.11. Fungsi *Recall*

#### 5.1.1.12. Fungsi *F1*

Fungsi *F1* digunakan untuk menghitung nilai metrik *F1* yang mengkombinasikan nilai *precision* dan *recall* karena pada suatu waktu nilai keduanya dapat memiliki bobot yang berbeda. Berikut potongan kode program untuk fungsi *F1*.

```
def f1(y_true, y_pred):
    result_precision = precision(y_true, y_pred)
    result_recall = recall(y_true, y_pred)
    return
    2*((result_precision*result_recall)/(result_precision+result_recall+K.epsilon()))
```

Kode 5.12. Fungsi *F1*

### 5.1.2. Kode Model Optimasi

#### 5.1.2.1. Model Gerbang LSTM-LSTM

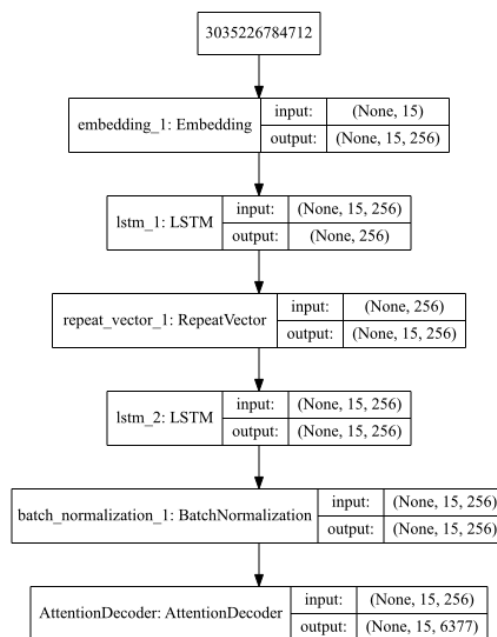
Kode pada model gerbang LSTM-LSTM ini dibuat sesuai dengan struktur *layer* model jaringan yang telah didefinisikan pada

Tabel 4.2. Berikut adalah potongan kode program untuk model gerbang LSTM-LSTM.

```
def define_model_lstm_lstm(vocab, timesteps, n_units):
    model = Sequential()
    model.add(Embedding(vocab, n_units, input_length=timesteps,
                        mask_zero=True))
    model.add(LSTM(n_units, return_sequences=False, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(RepeatVector(timesteps))
    model.add(LSTM(n_units, return_sequences=True, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(BatchNormalization())
    model.add(AttentionDecoder(n_units, vocab))
    return model
```

Kode 5.13. Fungsi Model Gerbang LSTM-LSTM

Kode program diatas akan menghasilkan model gerbang LSTM-LSTM dan berikut adalah visualisasi dari hasil kode program tersebut dengan parameteranya adalah 6377 *vocabulary*, 15 *timesteps*, dan 256 *n\_units*.



Gambar 5.6. Hasil Visualisasi Model Gerbang LSTM-LSTM

### 5.1.2.2. Model Gerbang LSTM-GRU

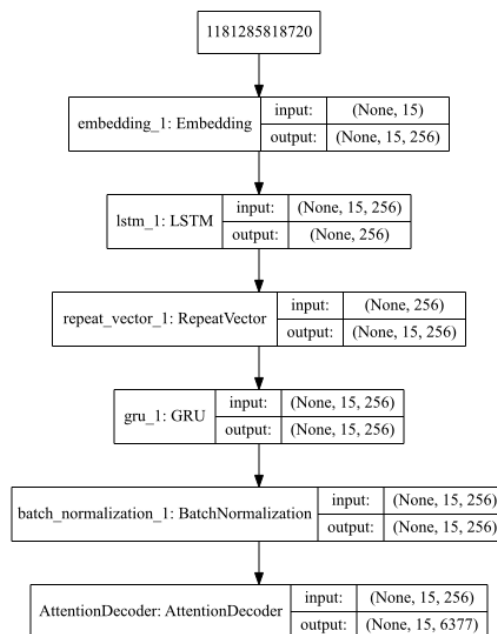
Kode pada model gerbang LSTM-GRU ini dibuat sesuai dengan struktur *layer* model jaringan yang telah didefinisikan pada Tabel 4.3.

Berikut adalah potongan kode program untuk model gerbang LSTM-GRU.

```
def define_model_lstm_gru(vocab, timesteps, n_units):
    model = Sequential()
    model.add(Embedding(vocab, n_units, input_length=timesteps,
                        mask_zero=True))
    model.add(LSTM(n_units, return_sequences=False, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(RepeatVector(timesteps))
    model.add(GRU(n_units, return_sequences=True, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(BatchNormalization())
    model.add(AttentionDecoder(n_units, vocab))
    return model
```

Kode 5.14. Fungsi Model Gerbang LSTM-GRU

Kode program diatas akan menghasilkan model gerbang LSTM-GRU dan berikut adalah visualisasi dari hasil kode program tersebut dengan parameternya adalah 6377 *vocabulary*, 15 *timesteps*, dan 256 *n\_units*.



Gambar 5.7. Hasil Visualisasi Model Gerbang LSTM-GRU

### 5.1.2.3. Model Gerbang GRU-GRU

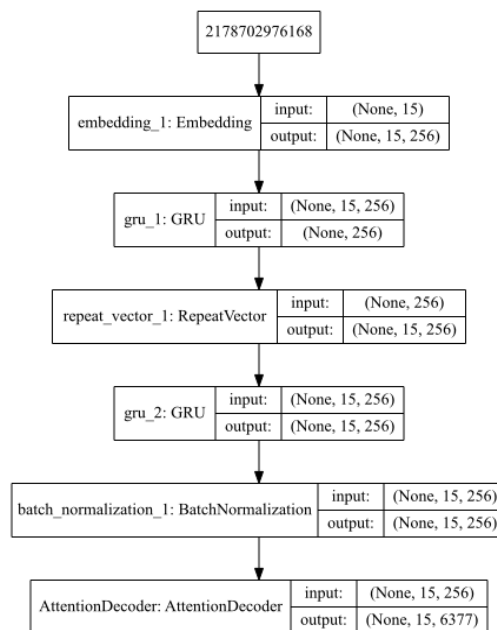
Kode pada model gerbang GRU-GRU ini dibuat sesuai dengan struktur *layer* model jaringan yang telah didefinisikan pada Tabel 4.4.

Berikut adalah potongan kode program untuk model gerbang GRU-GRU.

```
def define_model_gru_gru(vocab, timesteps, n_units):
    model = Sequential()
    model.add(Embedding(vocab, n_units, input_length=timesteps,
                        mask_zero=True))
    model.add(GRU(n_units, return_sequences=False, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(RepeatVector(timesteps))
    model.add(GRU(n_units, return_sequences=True, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(BatchNormalization())
    model.add(AttentionDecoder(n_units, vocab))
    return model
```

Kode 5.15. Fungsi Model Gerbang GRU-GRU

Kode program diatas akan menghasilkan model gerbang GRU-GRU dan berikut adalah visualisasi dari hasil kode program tersebut dengan parameternya adalah 6377 *vocabulary*, 15 *timesteps*, dan 256 *n\_units*.



Gambar 5.8. Hasil Visualisasi Model Gerbang GRU-GRU

#### 5.1.2.4. Model Gerbang GRU-LSTM

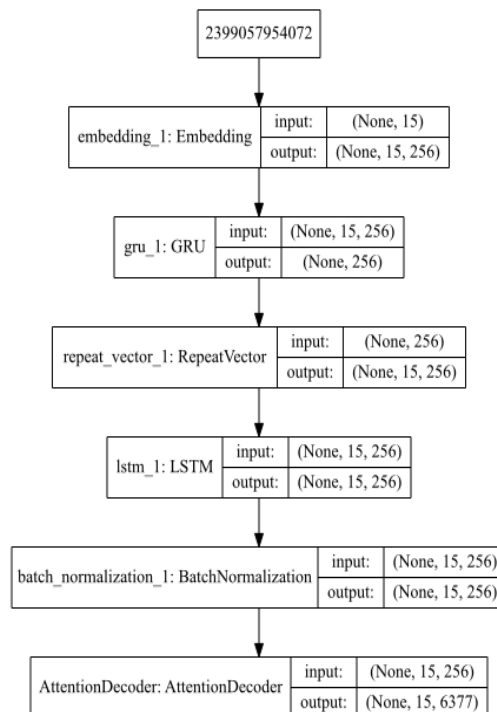
Kode pada model gerbang GRU-LSTM ini dibuat sesuai dengan struktur *layer* model jaringan yang telah didefinisikan pada Tabel 4.5.

Berikut adalah potongan kode program untuk model gerbang GRU-LSTM.

```
def define_model_lstm_gru(vocab, timesteps, n_units):
    model = Sequential()
    model.add(Embedding(vocab, n_units, input_length=timesteps,
                        mask_zero=True))
    model.add(LSTM(n_units, return_sequences=False, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(RepeatVector(timesteps))
    model.add(GRU(n_units, return_sequences=True, dropout=0.5,
                  recurrent_dropout=0.5))
    model.add(BatchNormalization())
    model.add(AttentionDecoder(n_units, vocab))
    return model
```

Kode 5.16. Fungsi Model Gerbang GRU-LSTM

Kode program diatas akan menghasilkan model gerbang GRU-LSTM dan berikut adalah visualisasi dari hasil kode program tersebut dengan parameteranya adalah 6377 *vocabulary*, 15 *timesteps*, dan 256 *n\_units*.



Gambar 5.9. Hasil Visualisasi Model Gerbang GRU-LSTM



### 5.1.3. Kode Proses Pelatihan dan Pengujian

Seperti yang telah dijelaskan pada bagian analisis dan perancangan optimasi model, proses pelatihan terdiri dari pemuatan sampel data, pembagian sampel pelatihan dan pengujian, tokenisasi sampel data, penyandian kata dalam bentuk sekuens, pendefinisian model jaringan, lalu mulai proses pelatihan dan tes. Berikut potongan kode program untuk proses pelatihan.

```
# persiapan pemuatan sampel data
dataset = load_clean_sample_data("OpenSubtitles2018","context-target.tsv")
dataset = np.reshape(dataset, (-1,2))
dataset1 = dataset.reshape(-1,1)

# persiapan pembagian sampel data
train, test = dataset[ : int(len(dataset)*80/100) ], dataset[
int(len(dataset)*80/100): ]
del dataset

# persiapan tokenizer
all_tokenizer = create_tokenizer(dataset1[:, 0])
all_vocab_size = len(all_tokenizer.word_index) + 1
all_length = max_length(dataset1[:, 0])
print('ALL Vocabulary Size: %d' % (all_vocab_size))
print('ALL Max question length: %d' % (all_length))
del dataset1

# persiapan training data
trainX = encode_sequences(all_tokenizer, all_length, train[:, 0])
trainY = encode_sequences(all_tokenizer, all_length, train[:, 1])
trainY = encode_output(trainY, all_vocab_size)
del train

# persiapan test data
testX = encode_sequences(all_tokenizer, all_length, test[:, 0])
testY = encode_sequences(all_tokenizer, all_length, test[:, 1])
testY = encode_output(testY, all_vocab_size)
del test

# definisikan model (lstm-lstm, lstm-gru, gru-gru, gru-lstm)
model = define_model_lstm_lstm(all_vocab_size, all_length, 256)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics =
['accuracy', precision, recall, f1])

# summarisasi model
print(model.summary())

# persiapan visualisasi model
tensor_board = TensorBoard(log_dir='./Graph', histogram_freq=0,
write_graph=True, write_images=True)

# proses pelatihan dan pengujian

# mulai pelatihan
history = model.fit(trainX, trainY, epochs=100, batch_size=64,
verbose=1, callbacks=[tensor_board])

# mulai pengujian
score = model.evaluate(testX, testY, batch_size=64)
print(score)
```

```

# simpan bobot jaringan model
filename = 'Model/MODEL_LSTM_LSTM_ATTNECODER.h5'
model.save(filename)

# visualisasi arsitektur model
SVG(model_to_dot(model, show_shapes=True).create(prog='dot',
format='svg'))

# visualisasi tensorboard : tensorboard --logdir ./Graph

```

Kode 5.17. Proses Pelatihan dan Pengujian

#### 5.1.4. Kode Generasi Respon

Proses generasi respon dimulai dari pemuatan sampel data, tokenisasi sampel data, pemuatan bobot jaringan, input kalimat, penyandian input secara sekuens, proses translasi, dan menampilkan respon sesuai dengan hasil generasi respon. Berikut potongan kode program untuk proses generasi respon.

```

# pemuatan sampel data
dataset = load_clean_sample_data()
dataset = np.reshape(dataset, (-1,2))
dataset1 = dataset.reshape(-1,1)

# persiapan tokenizer
all_tokenizer = create_tokenizer(dataset1[:,0])
all_vocab_size = len(all_tokenizer.word_index) + 1
all_length = max_length(dataset1[:, 0])

# pemuatan model
model = load_model('Model/MODEL_LSTM_LSTM_ATTNECODER.h5',
custom_objects={'AttentionDecoder': AttentionDecoder,
'precision':precision, 'recall':recall, 'f1':f1})

# mulai chatbot
while(True):
    # tampilkan interface untuk input kata
    q = (input(str("YOU: ")))

    # jika input 'bye' maka interface akan berakhir
    if q == 'bye':
        break
    q = q.strip().split('\n')

    # pembuatan sekuens konteks yang diinputkan
    X = all_tokenizer.texts_to_sequences(q)
    X = pad_sequences(X, maxlen=all_length, padding='post')

    # translasi respon percakapan
    a = translate(model, all_tokenizer, X)
    words = a.split()

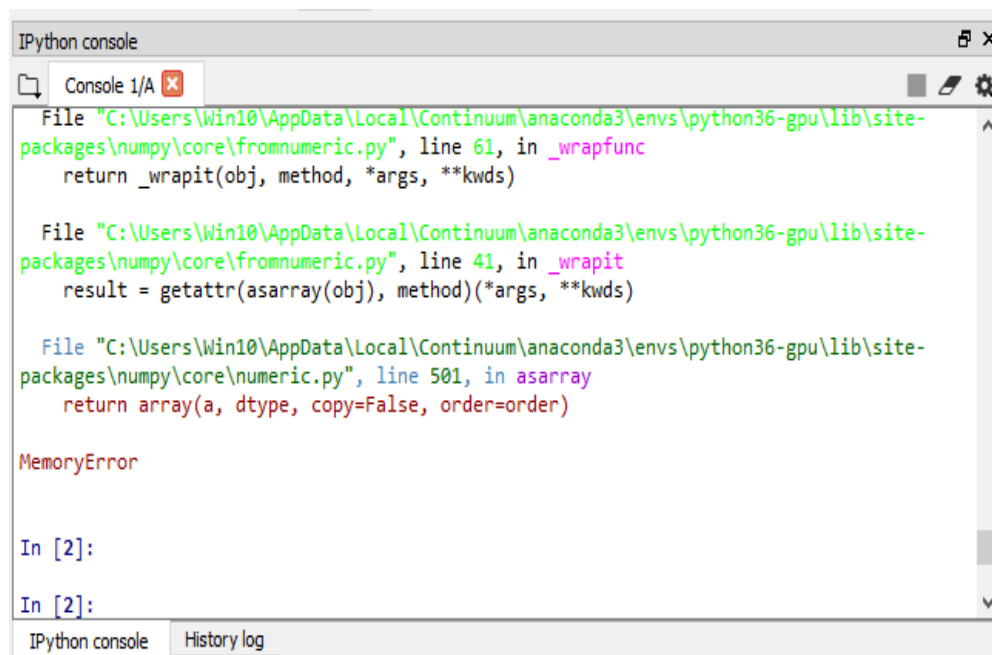
    # print respon percakapan
    print ('ANSWER: ' + " ".join(sorted(set(words), key=words.index)))

```

Kode 5.18. Proses Generasi Respon

## 5.2. Pelatihan Model Optimasi

Dari hasil pelatihan dapat diketahui lama waktu pelatihan, nilai metrik dan loss, ukuran bobot hasil pelatihan, dan mendapatkan analisa dari hasil pelatihan. Sampel data yang digunakan adalah 5-20 ribu. Oleh karena keterbatasan spesifikasi komputer, sampel data diatas 20 ribu tidak dapat diproses karena RAM komputer tidak cukup untuk menampung sampel data ke dalam *array* dan bisa memunculkan kesalahan *memory error*.



```
IPython console
Console 1/A
File "C:\Users\Win10\AppData\Local\Continuum\anaconda3\envs\python36-gpu\lib\site-packages\numpy\core\fromnumeric.py", line 61, in _wrapfunc
    return _wrapit(obj, method, *args, **kwds)

File "C:\Users\Win10\AppData\Local\Continuum\anaconda3\envs\python36-gpu\lib\site-packages\numpy\core\fromnumeric.py", line 41, in _wrapit
    result = getattr(asarray(obj), method)(*args, **kwds)

File "C:\Users\Win10\AppData\Local\Continuum\anaconda3\envs\python36-gpu\lib\site-packages\numpy\core\numeric.py", line 501, in asarray
    return array(a, dtype, copy=False, order=order)

MemoryError

In [2]:

In [2]:
IPython console History log
```

Gambar 5.10. Kesalahan *Memory Error*

### 5.2.1. Lama Waktu Pelatihan

Proses pelatihan akan mengambil data sebanyak 80% dari ukuran sampel data yang telah ditentukan. Jumlah *epoch* pada proses pelatihan sebanyak 400 kali. Dari proses pelatihan yang telah dilakukan, dapat diketahui total lama waktu yang dibutuhkan untuk melatih model Tanh RNN - Tanh RNN dengan *gradient descent* SGD seperti yang digambarkan pada Tabel 5.1. berikut ini. Sesudah itu, diketahui total lama waktu pelatihan untuk model gerbang LSTM-LSTM, LSTM-GRU, GRU-GRU, dan GRU-LSTM dengan

penambahan mekanisme perhatian versi Bahdanau dan *gradient descent* Adam seperti yang digambarkan pada Tabel 5.2. berikut ini.

Tabel 5.1. Lama Waktu Pelatihan Model Tanh RNN – Tanh RNN Dengan *Gradient Descent* SGD

Ukuran Sampel Data	Tanh RNN-Tanh RNN
5 ribu*	23 menit 31 detik
10 ribu*	57 menit 17 detik
15 ribu*	1 jam 37 menit
20 ribu*	2 jam 30 menit

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Tabel 5.2. Lama Waktu Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan *Gradient Descent* Adam

Ukuran Sampel Data	LSTM-LSTM	LSTM-GRU	GRU-GRU	GRU-LSTM
5 ribu*	1 jam 17 menit	1 jam 16 menit	1 jam 15 menit	1 jam 16 menit
10 ribu*	3 jam 18 menit	3 jam 17 menit	3 jam 10 menit	3 jam 15 menit
15 ribu*	6 jam 8 menit	6 jam 4 menit	5 jam 52 menit	5 jam 59 menit
20 ribu*	11 jam 27 menit	11 jam 12 menit	10 jam 54 menit	10 jam 59 menit

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Dari hasil pelatihan tersebut dapat diketahui bahwa lama waktu pelatihan yang paling cepat adalah model Tanh RNN-Tanh RNN pada ukuran sampel 5 ribu. Lama waktu pelatihan yang paling cepat untuk model optimasi pada semua ukuran sampel data adalah model gerbang GRU-GRU. Sebaliknya, waktu pelatihan yang paling lama untuk model optimasi dihasilkan oleh model gerbang LSTM-LSTM. Pada penggunaan gerbang yang

berbeda, seperti model gerbang LSTM-GRU dan GRU-LSTM terlihat membutuhkan waktu pelatihan yang tidak jauh berbeda. Lama waktu pelatihan model gerbang LSTM-GRU dan GRU-LSTM terlihat lebih cepat dibandingkan dengan model gerbang LSTM-LSTM, namun terlihat lebih lambat dibandingkan dengan model gerbang GRU-GRU.

### 5.2.2. Nilai Metrik dan *Loss* Pelatihan

Nilai metrik yang ditentukan dalam pelatihan adalah *accuracy*, *precision*, *recall*, dan *F1*. Nilai metrik yang dihasilkan selama proses pelatihan biasanya cenderung naik, artinya proses pelatihan berlangsung dengan baik. Nilai *loss* menunjukkan nilai kesalahan dari hasil proses pelatihan berdasarkan dengan data asli dari sampel data yang tersedia. Idealnya, nilai *loss* yang dihasilkan selama proses pelatihan cenderung menurun, menunjukkan bahwa proses pelatihan berlangsung dengan baik.

Pada kondisi nilai metrik dan *loss* yang tidak stabil, proses pelatihan tetap dijalankan sesuai dengan jumlah *epoch* yang telah ditentukan sebelumnya. Hal itu dilakukan untuk melihat perkembangan nilai tersebut agar menemukan pada *epoch* mana nilai-nilai tersebut terlihat baik dan optimal sesuai dengan hasil temuan penelitian. Berikut tabel perbandingan dari hasil perhitungan metrik dan *loss* selama proses pelatihan.

Tabel 5.3. Nilai Metrik dan *Loss* Pelatihan Model Tanh RNN – Tanh RNN Dengan *Gradient Descent* SGD

Model	Ukuran Sampel Data	Epoch	Accuracy	Precision	Recall	F1	Loss
Tanh RNN - Tanh RNN *	5 ribu	50	0,6970	1	3634	1,999	1,19e-7
		100	0,6970	1	3634	1,999	1,19e-7
		200	0,6970	1	3634	1,999	1,19e-7
		300	0,6970	1	3634	1,999	1,19e-7
		400	0,6970	1	3634	1,999	1,19e-7
	10 ribu	50	0,6631	1	5123	2	1,19e-7
		100	0,6631	1	5123	2	1,19e-7
		200	0,6631	1	5123	2	1,19e-7

		300	<b>0,6631</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,6631</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
	15 ribu	50	<b>0,6798</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		100	<b>0,6798</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		200	<b>0,6798</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		300	<b>0,6798</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,6798</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
	20 ribu	50	<b>0,6941</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		100	<b>0,6941</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		200	<b>0,6941</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		300	<b>0,6941</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,6941</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

\* Nilai tercetak tebal menandakan nilai paling terbaik.

Tabel 5.4. Nilai Metrik dan *Loss* Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan *Gradient Descent Adam*

Model	Ukuran Sampel Data	Epoch	Accuracy	Precision	Recall	F1	Loss
LSTM-LSTM*	5 ribu	50	0,8359	0,9395	0,7669	0,8444	0,6417
		100	0,9293	0,9613	0,9026	0,9310	0,2544
		200	0,9532	0,9706	0,9453	0,9578	0,1498
		300	0,9561	0,9728	0,9489	0,9607	0,1329
		400	<b>0,9578</b>	<b>0,9737</b>	<b>0,9498</b>	<b>0,9624</b>	<b>0,1221</b>
	10 ribu	50	0,7979	0,9297	0,7220	0,8127	0,8163
		100	0,8964	0,9477	0,8556	0,8992	0,3857
		200	0,9376	0,9624	0,9235	0,9425	0,2213
		300	0,9430	0,9649	0,9326	0,9484	0,1933
		400	<b>0,9441</b>	<b>0,9651</b>	<b>0,9346</b>	<b>0,9496</b>	<b>0,1830</b>
	15 ribu	50	0,7999	0,9311	0,7290	0,8142	0,8630
		100	0,8856	0,9463	0,8418	0,8888	0,4502
		200	0,9271	0,9583	0,9087	0,9235	0,3098
		300	0,9374	<b>0,9626</b>	0,9245	0,9364	0,2539
		400	<b>0,9384</b>	0,9622	<b>0,9267</b>	<b>0,9392</b>	<b>0,2239</b>
	20 ribu	50	0,8049	0,9317	0,7399	0,8247	0,8381
		100	0,8773	0,9436	0,8340	0,8854	0,4864
		200	0,9184	0,9543	0,8972	0,9248	0,3145
		300	0,9262	0,9560	0,9099	0,9324	0,2777
		400	<b>0,9297</b>	<b>0,9577</b>	<b>0,9155</b>	<b>0,9365</b>	<b>0,2584</b>
LSTM-GRU*	5 ribu	50	0,8190	0,9325	0,7534	0,8340	0,7205
		100	0,9099	0,9546	0,8720	0,9118	0,3308
		200	0,9484	0,9678	0,9387	0,9520	0,1766
		300	0,9531	0,9694	0,9457	0,9574	0,1494
		400	<b>0,9573</b>	<b>0,9730</b>	<b>0,9511</b>	<b>0,9619</b>	<b>0,1306</b>

		10 ribu	50	0,7580	0,9142	0,6853	0,7833	1,0440
			100	0,8375	0,9321	0,7750	0,8462	0,6377
			200	0,9102	0,9523	0,8810	0,9152	0,3365
			300	0,9270	0,9570	0,9085	0,9321	0,2629
			400	<b>0,9323</b>	<b>0,9591</b>	<b>0,9173</b>	<b>0,9377</b>	<b>0,2388</b>
		15 ribu	50	0,7784	0,9214	0,7088	0,8012	0,9732
			100	0,8501	0,9373	0,7949	0,8602	0,5961
			200	0,9069	0,9522	0,8806	0,9149	0,3519
			300	0,9196	0,9551	0,8996	0,9265	0,2982
			400	<b>0,9253</b>	<b>0,9566</b>	<b>0,9085</b>	<b>0,9319</b>	<b>0,2743</b>
		20 ribu	50	0,7865	0,9233	0,7216	0,8100	0,9595
			100	0,8485	0,9356	0,7956	0,8599	0,6263
			200	0,8963	0,9469	0,8669	0,9047	0,4077
			300	0,9070	0,9493	0,8836	0,9153	0,3588
			400	<b>0,9127</b>	<b>0,9501</b>	<b>0,8924</b>	<b>0,9203</b>	<b>0,3346</b>
	GRU-GRU*	5 ribu	50	0,8198	0,9320	0,7541	0,8336	0,7199
			100	0,9052	0,9536	0,8654	0,9073	0,3509
			200	0,9449	0,9664	0,9320	0,9494	0,1892
			300	0,9529	0,9702	0,9441	0,9569	0,1554
			400	<b>0,9543</b>	<b>0,9708</b>	<b>0,9472</b>	<b>0,9589</b>	<b>0,1424</b>
		10 ribu	50	0,7750	0,9170	0,7014	0,7947	0,9606
			100	0,8532	0,9341	0,7971	0,8601	0,5615
			200	0,9121	0,9513	0,8869	0,9179	0,3197
			300	0,9260	0,9563	0,9085	0,9318	0,2612
			400	<b>0,9298</b>	<b>0,9580</b>	<b>0,9155</b>	<b>0,9363</b>	<b>0,2433</b>
		15 ribu	50	0,7760	0,9183	0,7074	0,7991	0,9864
			100	0,8382	0,9329	0,7803	0,8497	0,6504
			200	0,8992	0,9489	0,8679	0,9065	0,3845
			300	0,9124	0,9513	0,8892	0,9192	0,3276
			400	<b>0,9168</b>	<b>0,9535</b>	<b>0,8962</b>	<b>0,9239</b>	<b>0,3103</b>
		20 ribu	50	0,7801	0,9185	0,7165	0,8049	0,9939
			100	0,8357	0,9322	0,7800	0,8492	0,6814
			200	0,8875	0,9443	0,8532	0,8964	0,4460
			300	0,8983	<b>0,9461</b>	0,8719	0,9075	0,3938
			400	<b>0,8998</b>	0,9449	<b>0,8753</b>	<b>0,9087</b>	<b>0,3899</b>
	GRU-LSTM*	5 ribu	50	0,8380	0,9386	0,7720	0,8471	0,6162
			100	0,9264	0,9590	0,8987	0,9278	0,2612
			200	0,9508	0,9685	0,9414	0,9547	0,1615
			300	0,9550	0,9721	0,9481	0,9599	0,1364
			400	<b>0,9564</b>	<b>0,9722</b>	<b>0,9507</b>	<b>0,9608</b>	<b>0,1290</b>
		10 ribu	50	0,7954	0,9231	0,7202	0,8091	0,8234
			100	0,8836	0,9433	0,8379	0,8875	0,4312
			200	0,9288	0,9574	0,9103	0,9332	0,2534
			300	0,9396	0,9625	0,9270	0,9444	0,2082
			400	<b>0,9423</b>	<b>0,9653</b>	<b>0,9318</b>	<b>0,9482</b>	<b>0,1916</b>
		15 ribu	50	0,7943	0,9240	0,7258	0,8135	0,8565
			100	0,8708	0,9406	0,8222	0,8774	0,4968
			200	0,9194	0,9553	0,8970	0,9252	0,3009

		300	0,9292	0,9583	0,9132	0,9351	0,2578
		400	<b>0,9325</b>	<b>0,9597</b>	<b>0,9180</b>	<b>0,9384</b>	<b>0,2423</b>
	20 ribu	50	0,7964	0,9239	0,7319	0,8167	0,8836
		100	0,8633	0,9389	0,8150	0,8726	0,5474
		200	0,9096	0,9505	0,8842	0,9161	0,3485
		300	0,9179	0,9533	0,8976	0,9246	0,3105
		400	<b>0,9212</b>	<b>0,9546</b>	<b>0,9033</b>	<b>0,9282</b>	<b>0,2937</b>

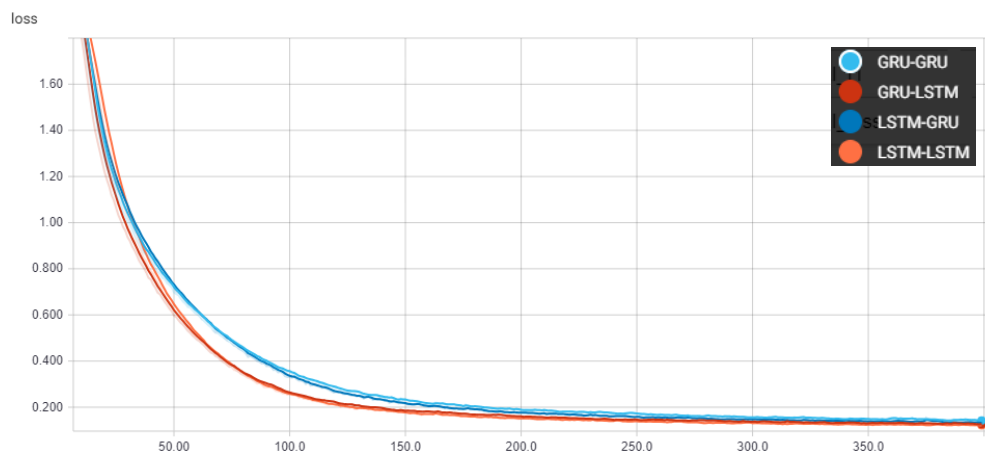
\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

\* Nilai tercetak tebal menandakan nilai paling terbaik.

### 5.2.3. Grafik Nilai *Loss* Pelatihan

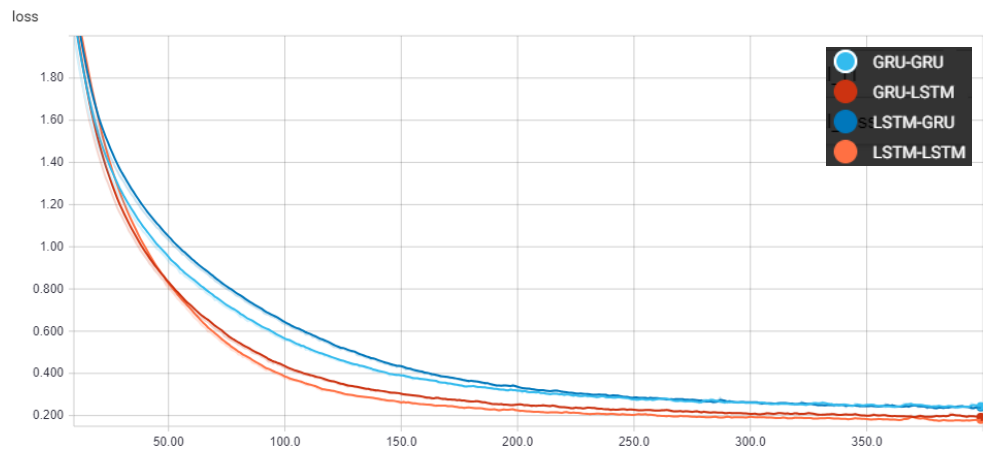
Nilai *loss* menunjukkan nilai kesalahan dari hasil proses pelatihan berdasarkan dengan data asli dari sampel data yang tersedia. Idealnya, nilai *loss* yang dihasilkan selama proses pelatihan cenderung menurun, menunjukkan bahwa proses pelatihan berlangsung dengan baik.

Pada seluruh sampel data, nilai *loss* mengalami penurunan yang tajam di awal pelatihan, yaitu dari *epoch* ke-1 sampai ke-100. Setelah itu, terjadi penurunan yang tidak banyak seperti di awal pelatihan, yaitu dari *epoch* ke-100 sampai ke-200. Diatas *epoch* ke-200, terjadi penurunan yang sangat sedikit dan cenderung melambat sampai *epoch* ke-400. Dari situ terlihat titik jenuh pelatihan karena penurunan nilai *loss* cenderung melambat bahkan stagnan.

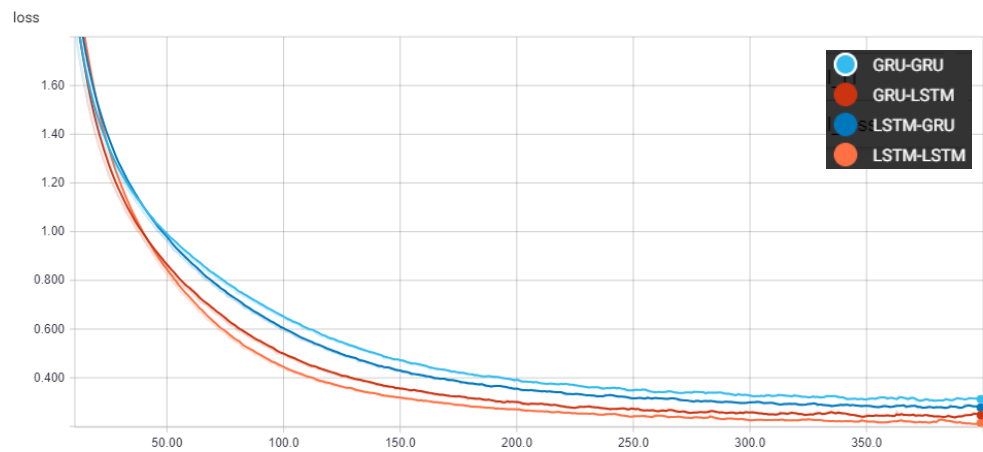


Gambar 5.11. Grafik Nilai *Loss* Pelatihan Pada Sampel Data 5 Ribu

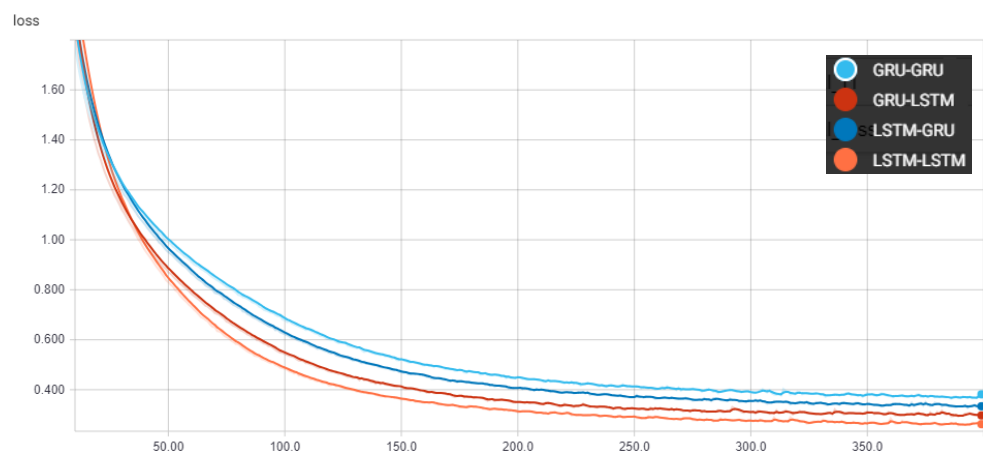




Gambar 5.12. Grafik Nilai *Loss* Pelatihan Pada Sampel Data 10 Ribu



Gambar 5.13. Grafik Nilai *Loss* Pelatihan Pada Sampel Data 15 Ribu



Gambar 5.14. Grafik Nilai *Loss* Pelatihan Pada Sampel Data 20 Ribu

#### 5.2.4. Ukuran Bobot Hasil Pelatihan

Proses pelatihan model pada akhirnya akan menghasilkan nilai bobot yang nantinya akan digunakan pada proses generasi respon. Bobot itu dihasilkan setelah proses pelatihan yang dilakukan sebanyak 400 *epoch*. Berikut tabel perbandingan dari ukuran bobot hasil pelatihan.

Tabel 5.5. Ukuran Bobot Hasil Pelatihan Model Tanh RNN – Tanh RNN Dengan *Gradient Descent* SGD

Ukuran Sampel	Tanh RNN- Tanh RNN
5 ribu*	24 MB
10 ribu*	33 MB
15 ribu*	40 MB
20 ribu*	50 MB

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Tabel 5.6. Ukuran Bobot Hasil Pelatihan Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan *Gradient Descent* Adam

Ukuran Sampel	LSTM-LSTM	LSTM-GRU	GRU-GRU	GRU-LSTM
5 ribu*	233 MB	232 MB	230 MB	232 MB
10 ribu*	409 MB	407 MB	406 MB	407 MB
15 ribu*	593 MB	592 MB	590 MB	592 MB
20 ribu*	924 MB	922 MB	921 MB	922 MB

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pengujiannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Dari hasil pelatihan tersebut dapat diketahui bahwa bobot hasil pelatihan yang paling kecil adalah model Tanh RNN-Tanh RNN pada sampel 5 ribu. Bobot hasil pelatihan yang paling kecil untuk model optimasi pada semua ukuran sampel data dipegang model gerbang GRU-GRU. Sebaliknya, bobot hasil pelatihan yang paling besar untuk model optimasi dipegang oleh model gerbang LSTM-LSTM. Untuk penggunaan gerbang yang berbeda, seperti LSTM-GRU dan GRU-LSTM, terlihat hasil bobot yang tidak jauh berbeda. Akan tetapi, bobot hasil pelatihan dari model LSTM-GRU dan GRU-LSTM terlihat lebih kecil dibandingkan dengan model gerbang LSTM-LSTM, namun terlihat lebih besar dibandingkan dengan model gerbang GRU-GRU.

#### **5.2.5. Analisis Hasil Pelatihan**

Berdasarkan hasil pelatihan data pada sistem ini, dapat diketahui bahwa lama waktu pelatihan sangat bergantung pada besarnya sampel data dan model yang digunakan untuk proses pelatihan. Lama waktu pelatihan yang paling cepat dimiliki oleh model Tanh RNN-Tanh RNN karena model ini adalah model dengan arsitektur RNN yang murni tanpa mekanisme gerbang dan hanya memiliki aktivasi *tanh*. Lama waktu pelatihan yang paling cepat untuk model optimasi dimiliki oleh model gerbang GRU-GRU karena secara umum GRU memerlukan proses komputasi untuk 2 gerbang (*reset* dan *update*), dibandingkan dengan LSTM yang membutuhkan proses komputasi untuk 3 gerbang (*input*, *output*, dan *forget*). Bisa disimpulkan bahwa proses komputasi yang terjadi di dalam model akan mempengaruhi lama waktu pelatihan.

Model yang dipilih juga mempengaruhi ukuran bobot hasil dari pelatihan. Ukuran bobot paling kecil dimiliki oleh model Tanh RNN-Tanh RNN karena model ini hanya menyimpan hasil komputasi dari aktivasi *tanh* di setiap langkah waktu. Ukuran bobot paling kecil untuk model optimasi dimiliki oleh model gerbang GRU-GRU. Mekanisme gerbang GRU memang dirancang untuk memiliki sedikit parameter (bobot) karena jumlah gerbangnya hanya 2

(*reset* dan *update*) dan tidak ada unit memori terpisah seperti LSTM. Ukuran bobot paling besar dimiliki oleh model gerbang LSTM-LSTM, karena LSTM memiliki unit memori terpisah dan dikontrol oleh banyak gerbang (*input*, *output*, dan *forget*).

Nilai metrik dan *loss* yang dihasilkan oleh model Tanh RNN-Tanh RNN bisa dikatakan tidak stabil karena terlihat adanya perbedaan nilai yang cukup tinggi dan mencolok untuk setiap *epoch* dari model tersebut. Jaringan model ini tidak stabil karena masalah dari hilang atau meledaknya nilai gradien. Terdapat kemungkinan gradien model semakin mengecil karena perkalian matriks yang berulang kali dengan bobot yang kecil dan juga bisa meledak karena hasil perkalian bobotnya lebih besar dari batas norma gradien. Oleh karena hilang dan meledaknya gradien, nilai bobot yang dihasilkan menjadi tidak stabil pada setiap *epoch* dan secara tidak langsung memberikan efek kepada nilai metrik, terutama nilai *precision*, *recall*, dan *F1* yang nilainya terlalu besar dan tidak berada diantara 0 dan 1, serta nilai *loss* yang nilainya sangat kecil di seluruh *epoch*.

Nilai metrik yang paling tinggi dan nilai *loss* yang paling rendah untuk model optimasi pada saat pelatihan dihasilkan oleh model gerbang LSTM-LSTM di semua sampel data, kemudian dilanjutkan dengan model GRU-LSTM, LSTM-GRU, dan GRU-GRU. Kombinasi jenis gerbang yang berbeda seperti model gerbang LSTM-GRU dan GRU-LSTM dapat mengungguli model gerbang GRU-GRU di semua sampel data, namun belum mampu mengungguli model gerbang LSTM-LSTM. Hal yang menarik lainnya adalah untuk model dengan kombinasi jenis gerbang yang berbeda, unit gerbang GRU lebih baik digunakan pada *encoder* model dan unit gerbang LSTM lebih baik digunakan pada *decoder* model. Hal ini diperlihatkan dari hasil nilai metrik dan *loss* pelatihan dari model gerbang GRU-LSTM yang lebih baik daripada model gerbang LSTM-GRU pada sampel 10 ribu, 15 ribu, dan 20 ribu.

Dari hasil pelatihan pada 4 model gerbang dapat diketahui bahwa jenis gerbang dapat mempengaruhi lama waktu pelatihan, ukuran bobot hasil pelatihan, serta nilai metrik dan *loss* pelatihan. Unit gerbang GRU membantu

mengurangi lama waktu dan ukuran bobot pelatihan ketika dilekatkan dengan unit gerbang LSTM. Hal itu diperlihatkan dari hasil model gerbang LSTM-GRU dan GRU-LSTM, lama waktu dan ukuran bobot keduanya mampu mengungguli model gerbang LSTM-LSTM. Akan tetapi, hasil nilai metrik dan *loss* pelatihan dari model gerbang LSTM-GRU dan GRU-LSTM itu masih belum bisa mengungguli model gerbang LSTM-LSTM. Kombinasi jenis gerbang yang berbeda ini juga tidak mengungguli lama waktu dan ukuran bobot hasil pelatihan dari model gerbang GRU-GRU.

### 5.3. Evaluasi Model Optimasi

#### 5.3.1. Lama Waktu Pengujian

Proses pengujian akan mengambil data sebanyak 20% dari ukuran sampel data yang telah ditentukan. Jumlah *epoch* pada proses pengujian sebanyak 400 kali. Dari proses pengujian yang telah dilakukan, dapat diketahui total lama waktu yang dibutuhkan untuk melatih model Tanh RNN - Tanh RNN dengan *gradient descent* SGD seperti yang digambarkan pada Tabel 5.7. berikut ini. Sesudah itu, diketahui total lama waktu pengujian untuk model gerbang LSTM-LSTM, LSTM-GRU, GRU-GRU, dan GRU-LSTM dengan penambahan mekanisme perhatian versi Bahdanau dan *gradient descent* Adam seperti yang digambarkan pada Tabel 5.8. berikut ini.

Tabel 5.7. Lama Waktu Pengujian Model Tanh RNN – Tanh RNN Dengan *Gradient Descent* SGD

Ukuran Sampel Data	Tanh RNN- Tanh RNN
5 ribu*	0,54 detik
10 ribu*	1,38 detik

15 ribu*	2,24 detik
20 ribu*	3,54 detik

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pengujiannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Tabel 5.8. Lama Waktu Pengujian Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan *Gradient Descent* Adam

Ukuran Sampel Data	LSTM-LSTM	LSTM-GRU	GRU-GRU	GRU-LSTM
5 ribu*	0,97 detik	0,96 detik	0,91 detik	0,96 detik
10 ribu*	2,45 detik	2,44 detik	2,37 detik	2,38 detik
15 ribu*	4,16 detik	4,08 detik	3,97 detik	4,08 detik
20 ribu*	7,23 detik	7,19 detik	6,88 detik	7,07 detik

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pengujiannya menggunakan GPU NVIDIA GTX 1070 8 GB.

Dari hasil pengujian tersebut dapat diketahui bahwa lama waktu pengujian yang paling cepat adalah model Tanh RNN-Tanh RNN pada ukuran sampel 5 ribu. Lama waktu pengujian yang paling cepat untuk model optimasi pada semua ukuran sampel data adalah model gerbang GRU-GRU dan yang paling lama dihasilkan oleh model gerbang LSTM-LSTM. Pada penggunaan gerbang yang berbeda, yaitu model gerbang LSTM-GRU dan GRU-LSTM, lama waktu pengujian model gerbang GRU-LSTM terlihat sedikit lebih cepat dibandingkan dengan model gerbang LSTM-GRU. Lama waktu pengujian model gerbang LSTM-GRU dan GRU-LSTM juga terlihat lebih cepat dibandingkan dengan model gerbang LSTM-LSTM, namun waktu pengujian keduanya terlihat lebih lambat dibandingkan dengan model gerbang GRU-GRU.

### 5.3.2. Nilai Metrik dan *Loss* Pengujian

Proses pengujian memakai sampel yang sudah dibuatkan untuk pengujian pada saat proses pelatihan, yaitu sebesar 20% dari sampel data. Nilai metrik yang digunakan dalam pengujian adalah *accuracy*, *precision*, *recall*, dan *F1*. Nilai *loss* menunjukkan nilai kesalahan dari hasil proses pengujian berdasarkan dengan data asli dari sampel data pengujian. Semakin tinggi nilai metrik menunjukkan hasil model yang semakin bagus, sementara semakin rendah *loss* menunjukkan tingkat kesalahan model yang rendah. Berikut tabel perbandingan dari hasil perhitungan metrik dan *loss* selama proses pengujian.

Tabel 5.9. Nilai Metrik dan *Loss* Pengujian Model Tanh RNN – Tanh RNN Dengan *Gradient Descent* SGD

Model	Ukuran Sampel Data	Epoch	Accuracy	Precision	Recall	F1	Loss
RNN-RNN*	5 ribu	50	<b>0,6199</b>	<b>1</b>	<b>3634</b>	<b>1,9900</b>	<b>1,19e-7</b>
		100	<b>0,6199</b>	<b>1</b>	<b>3634</b>	<b>1,9900</b>	<b>1,19e-7</b>
		200	<b>0,6199</b>	<b>1</b>	<b>3634</b>	<b>1,9900</b>	<b>1,19e-7</b>
		300	<b>0,6199</b>	<b>1</b>	<b>3634</b>	<b>1,9900</b>	<b>1,19e-7</b>
		400	<b>0,6199</b>	<b>1</b>	<b>3634</b>	<b>1,9900</b>	<b>1,19e-7</b>
	10 ribu	50	<b>0,6749</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
		100	<b>0,6749</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
		200	<b>0,6749</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
		300	<b>0,6749</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,6749</b>	<b>1</b>	<b>5123</b>	<b>2</b>	<b>1,19e-7</b>
	15 ribu	50	<b>0,7538</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		100	<b>0,7538</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		200	<b>0,7538</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		300	<b>0,7538</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,7538</b>	<b>1</b>	<b>6362</b>	<b>2</b>	<b>1,19e-7</b>
	20 ribu	50	<b>0,6936</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		100	<b>0,6936</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		200	<b>0,6936</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		300	<b>0,6936</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>
		400	<b>0,6936</b>	<b>1</b>	<b>8159</b>	<b>2</b>	<b>1,19e-7</b>

\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihannya menggunakan GPU NVIDIA GTX 1070 8 GB.

\* Nilai tercetak tebal menandakan nilai paling terbaik.

Tabel 5.10. Nilai Metrik dan *Loss* Pengujian Model Optimasi Dengan Mekanisme Perhatian versi Bahdanau dan *Gradient Descent* Adam

Model	Ukuran Sampel Data	Epoch	Accuracy	Precision	Recall	F1	Loss
LSTM-LSTM*	5 ribu	50	<b>0,5755</b>	<b>0,7603</b>	0,5473	<b>0,6363</b>	<b>4,4640</b>
		100	0,5735	0,6914	<b>0,5583</b>	0,6177	4,9230
		200	0,5673	0,6510	0,5571	0,6004	5,2720
		300	0,5659	0,6464	0,5545	0,5969	5,3180
		400	0,5676	0,6374	0,5538	0,5926	5,3950
	10 ribu	50	0,6224	<b>0,8043</b>	0,5800	<b>0,6737</b>	<b>3,6690</b>
		100	<b>0,6749</b>	0,7470	0,5887	0,6583	4,0810
		200	0,6152	0,7153	0,5973	0,6509	4,3250
		300	0,6252	0,7083	<b>0,6101</b>	0,6555	4,3540
		400	0,6212	0,7043	0,6047	0,6506	4,4220
	15 ribu	50	<b>0,7159</b>	<b>0,8695</b>	0,6637	<b>0,7525</b>	<b>2,6640</b>
		100	0,7066	0,8222	0,6775	0,7427	2,9240
		200	0,7025	0,7894	<b>0,6790</b>	0,7300	3,1330
		300	0,6994	0,7825	0,6756	0,7250	3,2040
		400	0,7018	0,7797	0,6778	0,7251	3,2440
	20 ribu	50	<b>0,6576</b>	<b>0,8160</b>	0,6226	<b>0,7061</b>	<b>3,5840</b>
		100	0,6372	0,7789	0,6074	0,6824	3,8820
		200	0,6459	0,7432	<b>0,6277</b>	0,6805	4,0710
		300	0,6362	0,7337	0,6170	0,6703	4,1710
		400	0,6331	0,7271	0,6136	0,6654	4,2210
LSTM-GRU*	5 ribu	50	<b>0,5815</b>	<b>0,7549</b>	<b>0,5519</b>	<b>0,6375</b>	<b>4,3540</b>
		100	0,5687	0,6935	0,5470	0,6115	4,7980
		200	0,5637	0,6496	0,5507	0,5960	5,1340
		300	0,5581	0,6356	0,5481	0,5886	5,2980
		400	0,5553	0,6250	0,5468	0,5832	5,4000
	10 ribu	50	<b>0,6374</b>	<b>0,8101</b>	0,5842	<b>0,6786</b>	<b>3,4520</b>
		100	0,6238	0,7611	0,5900	0,6645	3,8070
		200	0,6212	0,7188	0,6016	0,6549	4,0900
		300	0,6153	0,7040	0,5952	0,6450	4,2210
		400	0,6185	0,6957	<b>0,6018</b>	0,6453	4,3100
	15 ribu	50	<b>0,7136</b>	<b>0,8663</b>	0,6579	<b>0,7476</b>	<b>2,5940</b>
		100	0,7015	0,8253	0,6646	0,7361	2,8360
		200	0,7037	0,7930	0,6788	0,7313	3,0490
		300	0,7034	0,7812	<b>0,6808</b>	0,7274	3,1520
		400	0,6938	0,7734	0,6686	0,7171	3,2370
	20 ribu	50	<b>0,6596</b>	<b>0,8183</b>	0,6165	<b>0,7029</b>	<b>3,4630</b>
		100	0,6494	0,7800	<b>0,6216</b>	0,6916	3,7390
		200	0,6408	0,7496	0,6180	0,6774	3,9590
		300	0,6404	0,7384	0,6187	0,6732	4,0450
		400	0,6393	0,7308	0,6169	0,6690	4,0850
GRU-GRU*	5 ribu	50	<b>0,5727</b>	<b>0,7509</b>	0,5401	<b>0,6281</b>	<b>4,3690</b>
		100	0,5651	0,6790	0,5445	0,6042	4,8310
		200	0,5607	0,6464	0,5474	0,5927	5,1620



GRU-LSTM*		300	0,5561	0,6297	0,5447	0,5840	5,3090
		400	0,5576	0,6275	<b>0,5483</b>	0,5852	5,3710
	10 ribu	50	<b>0,6348</b>	<b>0,7871</b>	0,5958	<b>0,6780</b>	<b>3,5370</b>
		100	0,6202	0,7456	0,5890	0,6579	3,8740
		200	0,6174	0,7080	0,5959	0,6470	4,1610
		300	0,6088	0,6919	0,5851	0,6339	4,3020
		400	0,6252	0,6922	<b>0,6141</b>	0,6507	4,3550
	15 ribu	50	<b>0,7169</b>	<b>0,8571</b>	0,6538	<b>0,7414</b>	<b>2,6030</b>
		100	0,7005	0,8237	0,6550	0,7295	2,8140
		200	0,7052	0,7894	0,6827	0,7321	3,0080
		300	0,7023	0,7796	0,6794	0,7260	3,1220
		400	0,7078	0,7793	<b>0,6898</b>	0,7318	3,1570
	20 ribu	50	<b>0,6597</b>	<b>0,8173</b>	0,6098	<b>0,6983</b>	<b>3,4540</b>
		100	0,6435	0,7841	0,6042	0,6822	3,7030
		200	0,6446	0,7514	0,6169	0,6775	3,9130
		300	0,6434	0,7373	<b>0,6212</b>	0,6742	4,0080
		400	0,6417	0,7338	0,6184	0,6711	4,0520
	5 ribu	50	<b>0,5730</b>	<b>0,7440</b>	0,5427	<b>0,6274</b>	<b>4,4520</b>
		100	0,5691	0,6789	0,5515	0,6085	4,9100
		200	0,5663	0,6511	0,5525	0,5977	5,1740
		300	0,5694	0,6421	<b>0,5590</b>	0,5976	5,3090
		400	0,5684	0,6387	0,5573	0,5952	5,3270
	10 ribu	50	<b>0,6301</b>	<b>0,7912</b>	0,5901	<b>0,6758</b>	<b>3,6330</b>
		100	0,6201	0,7433	0,5961	0,6615	4,0040
		200	0,6189	0,7100	0,5992	0,6498	4,2460
		300	0,6210	0,7013	0,6022	0,6479	4,3360
		400	0,6195	0,6926	<b>0,6043</b>	0,6454	4,4100
	15 ribu	50	<b>0,7137</b>	<b>0,8610</b>	0,6648	<b>0,7501</b>	<b>2,6460</b>
		100	0,7040	0,8178	0,6696	0,7361	2,8970
		200	0,7014	0,7920	0,6751	0,7288	3,0850
		300	0,7053	0,7853	<b>0,6806</b>	0,7291	3,1690
		400	0,7033	0,7808	<b>0,6806</b>	0,7272	3,2130
	20 ribu	50	<b>0,6592</b>	<b>0,8115</b>	0,6237	<b>0,7051</b>	<b>3,5230</b>
		100	0,6417	0,7782	0,6102	0,6838	3,7970
		200	0,6491	0,7495	0,6272	0,6828	3,9770
		300	0,6503	0,7373	<b>0,6320</b>	0,6805	4,0540
		400	0,6430	0,7315	0,6244	0,6736	4,1140

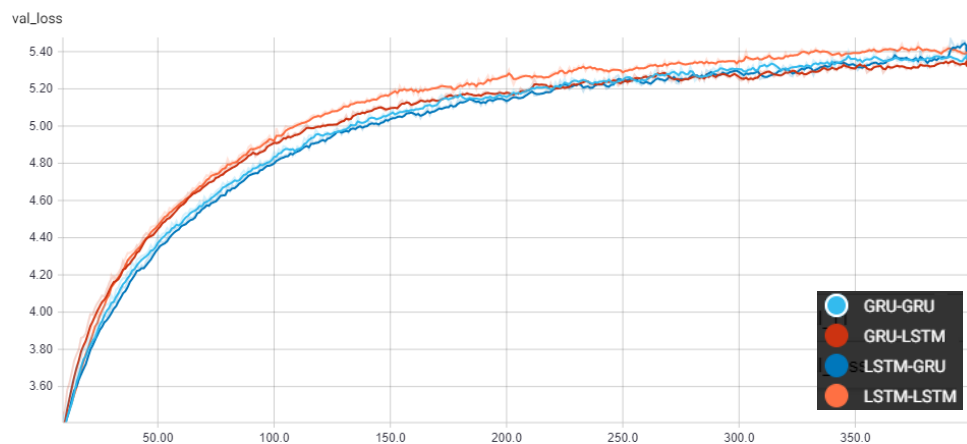
\* Sampel data diambil dari *OpenSubtitle 2018* dan proses pelatihnnya menggunakan GPU NVIDIA GTX 1070 8 GB.

\* Nilai tercetak tebal menandakan nilai paling terbaik.

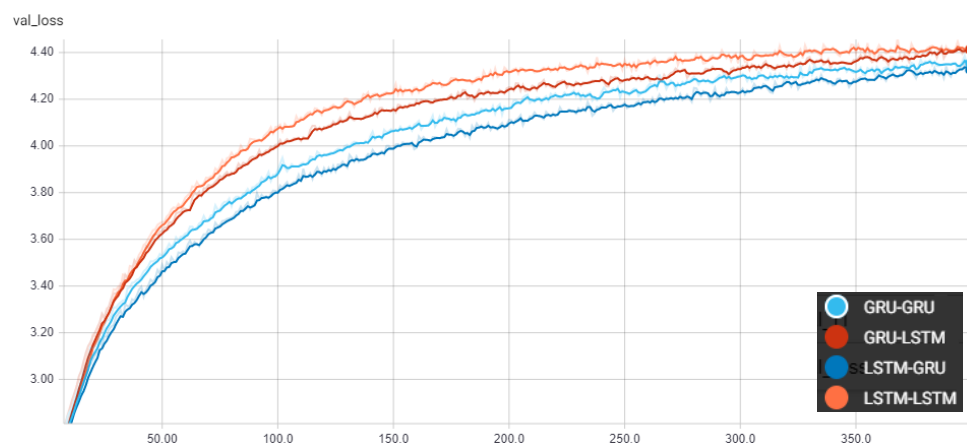
### 5.3.3. Grafik Nilai *Loss* Pengujian

Nilai *loss* menunjukkan nilai kesalahan dari hasil proses pengujian berdasarkan dengan data asli dari sampel data pengujian. Semakin rendah

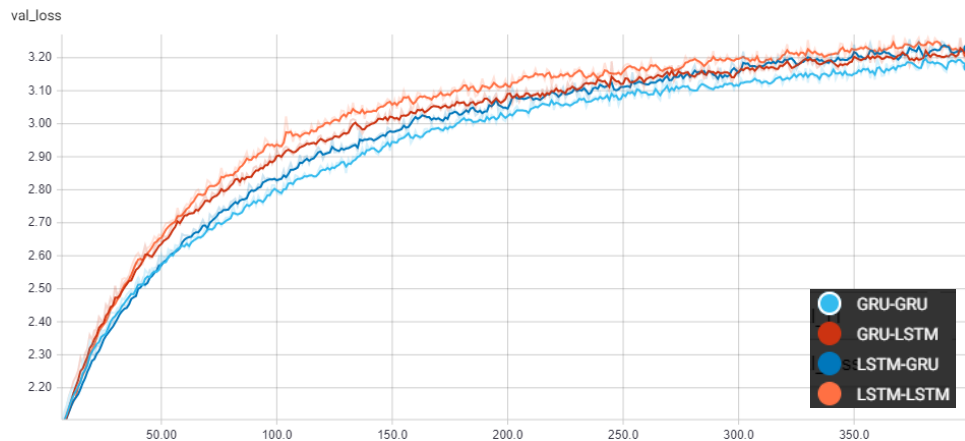
nilai *loss* menunjukkan tingkat kesalahan model yang rendah. Akan tetapi, hasil pengujian menunjukkan nilai *loss* yang semakin bertambah setiap *epoch*. Kenaikan sudah terjadi pada awal pengujian, dari *epoch* ke-1 sampai ke-50. Setelah itu, nilai *loss* mengalami kenaikan secara terus menerus dan mengalami perlambatan kenaikan nilai dimulai dari *epoch* ke-200 sampai ke-400. Performa model yang diuji hanya bagus pada data pelatihan daripada data pengujian, sehingga menyebabkan kondisi *overfitting*. Penyebab kondisi *overfitting* pada model akan dibahas pada analisis hasil pengujian.



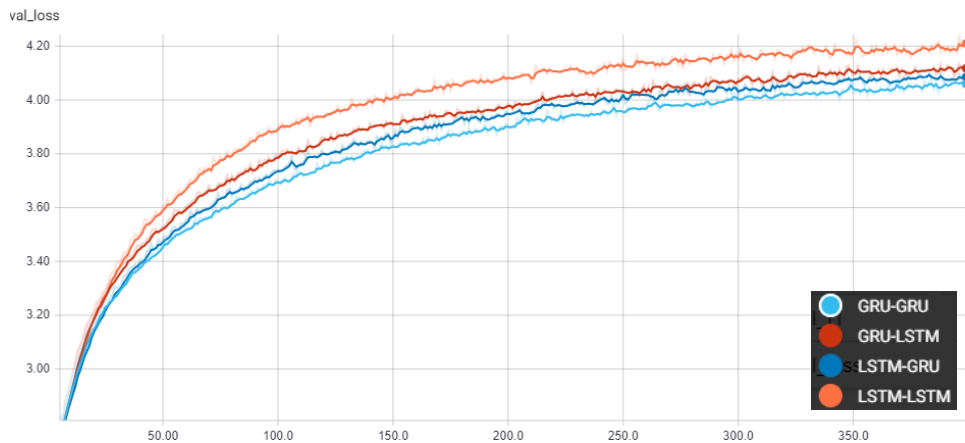
Gambar 5.15. Grafik Nilai *Loss* Pengujian Pada Sampel Data 5 Ribu



Gambar 5.16. Grafik Nilai *Loss* Pengujian Pada Sampel Data 10 Ribu



Gambar 5.17. Grafik Nilai *Loss* Pengujian Pada Sampel Data 15 Ribu



Gambar 5.18. Grafik Nilai *Loss* Pengujian Pada Sampel Data 20 Ribu

### 5.3.4. Analisis Hasil Pengujian

Berdasarkan hasil pengujian data pada sistem ini, dapat diketahui bahwa lama waktu pengujian sangat bergantung pada besarnya sampel data dan model yang digunakan untuk proses pengujian. Lama waktu pengujian yang paling cepat dimiliki oleh model Tanh RNN-Tanh RNN karena model ini adalah model dengan arsitektur RNN yang murni tanpa mekanisme gerbang dan hanya memiliki aktivasi *tanh*. Lama waktu pengujian yang paling cepat untuk model optimasi dimiliki oleh model gerbang GRU-GRU karena secara

umum GRU memerlukan proses komputasi untuk 2 gerbang (*reset* dan *update*), dibandingkan dengan LSTM yang membutuhkan proses komputasi untuk 3 gerbang (*input*, *output*, dan *forget*). Bisa disimpulkan bahwa proses komputasi yang terjadi di dalam model akan mempengaruhi lama waktu pengujian.

Nilai metrik dan *loss* yang dihasilkan oleh model Tanh RNN-Tanh RNN bisa dikatakan tidak stabil karena terlihat adanya perbedaan nilai yang cukup tinggi dan mencolok untuk setiap *epoch* dari model tersebut. Model Tanh RNN-Tanh menghasilkan jaringan yang tidak stabil yang tidak dapat belajar dari data pelatihan sehingga menghasilkan nilai metrik dan *loss* yang tidak bagus pada saat proses pengujian. Oleh karena hilang dan meledaknya gradien saat pelatihan, nilai bobot yang diperbarui itu tidak stabil pada setiap *epoch* dan secara tidak langsung memberikan efek kepada nilai metrik, terutama nilai *precision*, *recall*, dan *F1* yang nilainya terlalu besar dan tidak berada diantara 0 dan 1, serta nilai *loss* yang nilainya sangat kecil di seluruh *epoch*.

Pada perhitungan nilai metrik dan *loss* model optimasi dapat dilihat bahwa hasil pada *epoch* ke-50 untuk setiap model gerbang pada semua sampel data memiliki nilai metrik dan *loss* terbaik, kecuali nilai metrik *recall*. Selisih hasil nilai metrik antara *epoch* terlihat kecenderungan penurunan nilai dimana nilai metrik yang didapatkan lebih tinggi pada *epoch* ke-50 daripada *epoch* ke-400. Hal ini juga terlihat pada nilai *loss* pada setiap model gerbang, dimana angkanya menjadi tinggi untuk setiap *epoch*. Hal tersebut dapat membuktikan bahwa jumlah iterasi maksimal berada pada *epoch* ke-50 dan mulai memperlihatkan titik jenuh ketika jumlah *epoch* tersebut makin besar.



Jumlah *epoch* yang besar tidak menjamin kualitas hasil akurasi yang semakin baik. Hal ini dikenal dengan kondisi *overfitting* dimana nilai metrik yang dihasilkan justru semakin tidak bagus terutama pada pengujian data diluar dari *dataset* pelatihan. Nilai metrik yang dihasilkan pada *epoch* ke-100 tidak lebih baik dari *epoch* ke-50, karena mulai terlihat beberapa kondisi *overfitting* yang terjadi di beberapa hasil pengujian. Kondisi *overfitting* juga

nampak pada *epoch* ke-200, *epoch* ke-300, dan *epoch* ke-400. Dari situ bisa disimpulkan bahwa semakin tinggi *epoch* dapat membuat tidak stabilnya nilai metrik *recall*, lalu bisa menurunkan nilai metrik *accuracy*, *precision*, dan *F1* serta menyebabkan nilai *loss* menjadi semakin tinggi.

Ada beberapa penyebab yang membuat nilai metrik dan *loss* pada proses pelatihan lebih baik dibandingkan dengan pengujian, salah satunya adalah kualitas dari *dataset OpenSubtitles 2018 Bahasa Indonesia*. *Dataset OpenSubtitle* termasuk jenis *open domain* dimana terdapat banyak topik pembicaraan dan jenis *dataset* ini sangat berderau karena satu percakapan yang sama dapat dilakukan oleh banyak aktor. Kualitas *subtitle* yang ada pada *OpenSubtitle* perlu ditinjau lebih lanjut sebab banyak ditemukan kata yang salah ketik, kesalahan ejaan, kesalahan kata saling sambung, serta bentuk kata yang tidak baku dan tidak formal, yang pada akhirnya menyebabkan duplikasi pada kamus kata (*vocabulary*). Susunan kata dalam kalimat percakapan juga terlihat bebas, tidak struktural atau tidak mengikuti tatanan subjek, predikat, objek, dan keterangan (SPOK). Selain itu, terdapat ketidaklogisan pemetaan antara konteks dengan target, sehingga percakapan yang terjadi tidak sesuai dengan topiknya. Oleh karena itu, perlu penelitian lebih lanjut untuk penggunaan *dataset* dari *OpenSubtitle Bahasa Indonesia*.

#### **5.4. Antarmuka dan Hasil Generasi Respon**

Antarmuka sistem yang digunakan pada penelitian ini berupa teks pada *command line prompt* pada program *Spyder*. Penggunaan antarmuka tersebut dipilih karena model hasil implementasi pada penelitian ini ditujukan untuk akademisi dan peneliti pada bidang *natural language processing* dan *deep learning* serta umumnya mengerti penggunaan antarmuka tersebut. Hasil generasi respon pada sistem sederhana ini ditampilkan dalam bentuk teks yang ditampilkan dengan menggunakan bahasa pemrograman Python. Berikut ini contoh antarmuka program serta hasil generasi respon pada penelitian ini.

Console 1/A  

YOU: siapa yang mau mulai ?  
ANSWER: aku

YOU: ada beberapa pertanyaan yang ingin kuajukan  
ANSWER: mungkin pertanyaan itu tidak berarti apa tapi

YOU: beritahu kami apa yang kamu pikirkan  
ANSWER: mungkin kami dapat menunjukkan kesalahan yang kaubuat

YOU: kamu ingin menunjukan pada aku sesuatu yang lainnya?  
ANSWER: unh

YOU: kamu baik baik saja ?  
ANSWER: iya baik saja

YOU: apa yang kaupikir ?  
ANSWER: aku tidak tahu

YOU: kamu jauh lebih bodoh dari yang kukira  
ANSWER: astaga

YOU: kamu bekerja sendirian ?  
ANSWER: memangnya kenapa

YOU: bagaimana kamu mengetahui semua ini ?  
ANSWER: itulah pekerjaanku

YOU: apa pekerjaanmu ?  
ANSWER: aku seorang pembuat jam

YOU: apakah kamu seorang salesman ?  
ANSWER: aku seorang arsitek

YOU: apa kamu yakin?  
ANSWER: iya yakin

YOU: sudah cukup  
ANSWER: kamu menggelikan kek pil tidur

YOU: kaupikir kamu itu siapa ?  
ANSWER: tenang

YOU: kaupikir lucu ?  
ANSWER: lupakan saja kawan

YOU: aku tidak tahu kau tidak bersahabat  
ANSWER: memang tidak kami hanya punya kesepakatan

YOU: aku tidak ingin melakukan hal seperti itu .  
ANSWER: oke

YOU: aku harus kembali bekerja  
ANSWER: tunggu sebentar

YOU: ada sesuatu yang salah ?  
ANSWER: hmm

YOU: kamu berpikir ini lucu ?  
ANSWER: kamu berpikir ini adalah permainan

YOU: apakah kamu ingin aku pergi?  
ANSWER: tidak

YOU: aku sedang bekerja  
ANSWER: aku tahu

```

YOU: aku akan menunggumu lama , lama sekali
ANSWER: nikmati waktumu

YOU: apakah kamu sudah gila ?
ANSWER: tentu saja tidak

YOU: kami hanya membuang waktu
ANSWER: giliranmu sekarang

YOU: maukah kamu dengar alasanku ?
ANSWER: tidak aku mau dengar alasanmu

YOU: aku memiliki beberapa bisnis untuk ditangani
ANSWER: iya

YOU: lalu apa yang kamu inginkan ?
ANSWER: aku hanya ingin bicara

YOU: kamu telah menang
ANSWER: benarkah

YOU: aku baik baik saja .
ANSWER: oke berusaha untuk tetap bertahan

YOU: aku akan berusaha
ANSWER: terima kasih

```

Gambar 5.19. Antarmuka Program *Chatbot*

## 5.5. Kelebihan dan Kekurangan Sistem

Kelebihan:

1. Terdapat empat opsi model gerbang yang dapat digunakan, yaitu LSTM-LSTM, LSTM-GRU, GRU-LSTM, dan GRU-GRU.
2. Pembentukan model jaringan yang fleksibel berkat *library* Tensorflow dan Keras, sehingga model dapat dikembangkan untuk penelitian lebih lanjut.

Kekurangan:

1. Model jaringan tidak dapat maksimal untuk menghasilkan respon dengan pengetahuan diluar dari sampel data.
2. Kamus kata (*vocabulary*) yang berasal dari sampel data termasuk kurang banyak, sehingga kata-kata yang tidak ada dalam kamus kata membuat model tidak mampu mengenali kata tersebut dan memberikan respon yang tidak logis, bahkan tidak sesuai dengan konteks pertanyaan.

## BAB VI

### Kesimpulan dan Saran

#### 6.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka kesimpulan yang didapatkan adalah sebagai berikut.

- a. Penelitian tentang pengembangan model percakapan dengan algoritma SEQ2SEQ berbasis *Recurrent Neural Network* dengan model gerbang LSTM-LSTM, LSTM-GRU, GRU-GRU, dan GRU-LSTM serta penambahan mekanisme perhatian dan *gradient descent* pada sampel data dari *OpenSubtitle 2018* Bahasa Indonesia telah berhasil dilakukan.
- b. Hasil dari pengujian dari 4 model percakapan yang sudah dioptimalkan menunjukkan nilai metrik yang cukup tinggi dan *loss* yang rendah pada sampel 15 ribu dan *epoch* ke-50, dengan skor pengujian terbaik dihasilkan oleh model gerbang LSTM-LSTM, yaitu hasil *accuracy* sebesar 71,59%, *precision* 86,95%, *recall* 66,37%, *F1* 75,25%, dan *loss* 2,664.

#### 6.2. Saran

Menurut penulis, penelitian ini masih bisa dikembangkan. Berikut adalah beberapa hal yang dapat dilakukan untuk mengembangkan penelitian ini lebih lanjut.

- a. Sampel data yang digunakan dapat diganti dengan *dataset* yang sedikit berderau dan perlu penelitian lebih lanjut tentang pemilihan sampel data.
- b. Mengembangkan penggunaan representasi vektor kata yang telah dilatih sebelumnya (*pre-trained*) ke dalam model, seperti *Word2Vec*, *GloVe*, atau *Fast Text* sehingga pemetaan kamus kata (*vocabulary*) menjadi lebih banyak.



## DAFTAR PUSTAKA

- Abadi, M. *et al.* (2016) *TensorFlow: A system for large-scale machine learning*. Tersedia di: <https://tensorflow.org>. (Diakses: 11 January 2019).
- Abu Shawar, B. dan Atwell, E. (2007) ‘Chatbots: are they really useful?’, *LDV-Forum: Zeitschrift für Computerlinguistik und Sprachtechnologie*, 22(1), pp. 29–49. doi: 10.1.1.106.1099.
- Bahdanau, D., Cho, K. dan Bengio, Y. (2014) ‘Neural Machine Translation by Jointly Learning to Align and Translate’. Tersedia di: <http://arxiv.org/abs/1409.0473> (Diakses: 10 January 2019).
- Bay, A. dan Sengupta, B. (2017) ‘StackSeq2Seq: Dual Encoder Seq2Seq Recurrent Networks’. Tersedia di: <http://arxiv.org/abs/1710.04211> (Diakses: 11 January 2019).
- Cho, K. *et al.* (2014) ‘Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation’. Tersedia di: <http://arxiv.org/abs/1406.1078> (Diakses: 21 January 2019).
- Chung, J. *et al.* (2014) ‘Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling’. Tersedia di: <http://arxiv.org/abs/1412.3555> (Diakses: 10 January 2019).
- Karpathy, A., Johnson, J. dan Fei-fei, L. (2016) ‘Visualizing and Understanding Recurrent Networks’, pp. 1–11. doi: 10.1007/978-3-319-10590-1\_53.
- Khurana, D. *et al.* (2017) ‘Natural Language Processing: State of The Art, Current Trends and Challenges’, *ArXiv*, (Figure 1). Tersedia di: <http://arxiv.org/abs/1708.05148>.
- Kingma, D. P. dan Ba, J. (2014) ‘Adam: A Method for Stochastic Optimization’. Tersedia di: <http://arxiv.org/abs/1412.6980> (Diakses: 11 January 2019).
- Lopyrev, K. (2015) ‘Generating News Headlines with Recurrent Neural Networks’. Tersedia di: <http://arxiv.org/abs/1512.01712> (Accessed: 11 January 2019).
- Luong, M.-T., Pham, H. dan Manning, C. D. (2015) ‘Effective Approaches to Attention-based Neural Machine Translation’. doi: 10.18653/v1/D15-1166.

- Lison, P. dan Tiedemann, J. (2018) 'OpenSubtitles 2018'. Tersedia di: <http://opus.nlpl.eu/OpenSubtitles-v2018.php> (Diakses: 20 February 2019).
- Radziwill, N. M. dan Benton, M. C. (2017) 'Evaluating Quality of Chatbots and Intelligent Conversational Agents'. Tersedia di: <http://arxiv.org/abs/1704.04579>.
- Salehinejad, H. *et al.* (2017) 'Recent Advances in Recurrent Neural Networks', pp. 1–21. doi: 10.1162/089976600300015015.
- Sutskever, I., Vinyals, O. dan Le, Q. V. (2014) 'Sequence to Sequence Learning with Neural Networks', pp. 1–9. doi: 10.1007/s10107-014-0839-0.
- Sze, V. *et al.* (2017) 'Efficient Processing of Deep Neural Networks: A Tutorial and Survey'. Tersedia di: <http://arxiv.org/abs/1703.09039> (Diakses: 11 January 2019).
- Vinyals, O. dan Le, Q. (2015) 'A Neural Conversational Model', 37. doi: 10.1210/jc.2006-0173.
- Wetstein, S. (2017) 'Designing a Dutch Financial Chatbot'. Tersedia di: [https://beta.vu.nl/nl/Images/stageverslag-wetstein\\_tcm235-851825.pdf](https://beta.vu.nl/nl/Images/stageverslag-wetstein_tcm235-851825.pdf) (Diakses: 11 January 2019).
- Young, T. *et al.* (2018) 'Recent trends in deep learning based natural language processing [Review Article]', *IEEE Computational Intelligence Magazine*, 13(3), pp. 55–75. doi: 10.1109/MCI.2018.2840738.