|  |
| --- |
| TUGAS AKHIR  Nested Named-Entity Recognition dalam Bahasa Indonesia Menggunakan  Sequence-to-Set Network |
| LOGO ISTTS TRANS PUTIH |
|  |
| Oleh:  Nicoletta Valencia Halim  218116699 |
| PROGRAM SARJANA  PROGRAM STUDI INFORMATIKA  FAKULTAS SAINS DAN TEKNOLOGI  INSTITUT SAINS DAN TEKNOLOGI TERPADU SURABAYA  SURABAYA  2022 |

TUGAS AKHIR

Nested Named-Entity Recognition dalam Bahasa Indonesia Menggunakan Sequence-to-Set Network

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk Memperoleh Gelar

Sarjana Komputer

Pada

Institut Sains dan Teknologi Terpadu Surabaya

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Joan Santoso, S.Kom., M.Kom. (Pembimbing)
2. \_\_\_\_\_\_\_\_, S.Kom., M.Kom. (Penguji I)
3. \_\_\_\_\_\_\_\_, M.App.Sc. (Penguji II)
4. \_\_\_\_\_\_\_\_, S.Kom., M.Kom. (Penguji III)

SURABAYA

JUNI 20

# SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Nicoletta Valencia Halim

Fakultas/ Prodi : Sains dan Teknologi/ Informatika

NRP : 218116699

dengan ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir dengan judul:

NESTED NAMED-ENTITY RECOGNITION DALAM BAHASA INDONESIA MENGGUNAKAN SEQUENCE-TO-SET NETWORK

TUGAS AKHIR

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 01-06-2022

Yang Membuat Pernyataan,

Nicoletta Valencia Halim

218116699

# ABSTRAK

Cabang ilmu komputer yang mempelajari bagaimana caranya computer dapat memahami dan menganalisa bahasa manusia adalah cabang ilmu Pengolahan Bahasa Alami atau *Natural Language Processing* (NLP). Sebagai ilmu yang memahami arti dari kalimat yang diberikan dari bahasa, komputer memiliki beragam *task* yang bisa dilakukan. Karena itu ada banyak topik juga dalam bidang NLP yang membagi task-task tersebut agar mudah untuk mencapai solusi-solusi komputer melaksanakan tugas mereka. Salah satu topik dalam bidang NLP yang umum dan juga akan dibahas di tugas akhir ini adalah Named Entity Recognition (NER).

Task NER memang sudah umum dan banyak diteliti, terutama di Bahasa Inggris. Namun ada task merupakan bagian dari NER yang masih belum seumum NER sendiri untuk diteliti yaitu *Nested* Named Entity Recognition. Perbedaan yang cukup singkat yaitu pengenalan entitas dalam suatu kalimat bisa bersarang. Contohnya Jalan Ir. Soekarno bukan saja entitas lokasi namun juga terdapat entitas bersarang didalamnya yaitu Ir. Soekarno sebagai entitas orang. Terdapat satu metode yang paling sering digunakan dalam beberapa penelitian yang sudah dilakukan untuk Nested NER, metode ini metode *span-based*.Namun karena beberapa kekurangannya seperti komputasi dan akurasi dalam membentuk span memberi suatu halangan, ada satu metode yang baru ditemukan pada tahun 2021 yaitu metode Sequence-To-Set Network.

Berdasarkan hasil penelitiannya metode tersebut, metode ini mengalahkan akurasi dalam performa sebanyak 0,50% - 2,99% terhadap metode span-based dengan dataset berbeda-beda. Hal ini dapat dicapai dengan konsep yang mirip dengan *seq2seq* yaitu menggunakan *encoder* dan *decoder layer* namun dengan isi dan output berbeda. Encoder akan melakukan enkripsi terhadap kalimat input menjadi berbagai macam embedding yang berbeda-beda. Kemudian hasil itu akan dilewatkan decoder layer yang memiliki ilmu dari *self* dan *cross-attention*. Bagian dari decoder ini mengambil inspirasi dengan bentuk arsitektur Tranformers. Sehingga output dari decoder bisa berupa sebuah set yang berisi batasan kata kiri dan kanan dan juga jenis entitas yang diprediksikan. Hal lain yang mendukung Sequence-To-Set Network untuk menjadi metode yang efisien adalah pemilihan *loss function* berdasarkan *bipartite matching* dengan algoritma Hungarian. Tugas akhir ini memiliki tujuan untuk menghasilkan metode yang baik untuk pengenalan entitas bernama/NER dalam bahasa Indonesia dengan dataset berita CNN Indonesia berdomai politik.

# ABSTRACT

The branch of computer science that studies how computers can understand and analyze human language is the branch of Natural Language Processing (NLP). As a branch knowledge that understands the meaning of sentences given from language, computers have various tasks that can be done. Therefore, there are also many topics in the field of NLP that divide these tasks so that it is easy to reach solutions for the computer to carry out their tasks. One of the common topics in NLP that will also be discussed in this final project is Named Entity Recognition (NER).

The NER task is already common and has been widely researched, especially in the English language. However, there is a task that is part of NER which is still not as common as NER itself to be studied, that is the Nested Named Entity Recognition (Nested NER). There is a short difference between them, that is the recognition of entities in a sentence can be nested. For example, Jalan Ir. Soekarno is not only a location entity but also a nested entity in it, which is the word Ir. Soekarno as a person entity. There is one method that is most often used in several studies that have been carried out for Nested NER, this method is the span-based method. However, because some of its drawbacks such as computation and accuracy in forming spans from words, there is one method that was discovered in 2021, namely the Sequence-To-Set Network method.

Based on the results of the research for this method, this method beats accuracy in performance by 0.50% - 2.99% against the span-based method based on different datasets. This can be achieved with a similar concept to seq2seq which uses an encoder and decoder layer but with different layers and outputs. The encoder will encrypt the input sentence with a concatenation of variety different embeddings. Then the results will be passed to a decoder layer which has knowledge of self and cross-attention. Part of this decoder takes inspiration from the architectural form of Transformers. So, the output of the decoder can be a set containing left and right word boundaries and also the predicted entity type. Another thing that supports Sequence-To-Set Network to be an efficient method is the selection of a loss function based on bipartite matching with the Hungarian algorithm. The goal of this final project is to produce a good method for identifying an entity named/NER in Indonesian with a politically-dominated CNN Indonesia news dataset.

# KATA PENGANTAR

Syukur diberikan kepada Tuhan Yang Maha Esa, karena berkat dan bimbingan-Nya, sehingga Tugas Akhir dengan judul “Nested Named-Entity Recognition dalam Bahasa Indonesia Menggunakan Sequence-to-Set Network” ini dapat diselesaikan dengan baik. Penyusunan Tugas Akhir ini pastinya mengalami kesulitan dan masalah. Namun berkat bantuan dan bimbingan secara fisik maupun emosional dari beberapa pihak maka segala kesulitan dan permasalahan tersebut dapat diselesaikan. Karena itu pada kesempatan ini penulis tugas akhir ingin mengucapkan terima kasih kepada :

1. Allah yang Maha Kuasa yang telah memberi hikmat, berkat dan bimbingan selama tugas akhir ini dikerjakan.
2. Papa George Arius Halim, saudara Sabrina Fiorenza Halim, ibu baptis Adriana Maya Politon, almarhum ibu Dona Ira Politon yang telah memberikan dukungan, doa dan motivasi dalam penyelesaian Tugas Akhir ini.
3. Bapak Dr. Ir. Joan Santoso, S.Kom., M.Kom., selaku dosen pembimbing yang telah banyak memberikan arahan, koreksi, serta saran kepada penulis selama penyusunan Tugas Akhir ini. Juga dukungan dan pengertiannya selama kesulitan penyusunan dilaksanakan.
4. Bapak Dr. Ir. Gunawan, M.Kom. selaku dosen wali penulis yang telah memberi ilmu, pengalaman dan juga bantuan selama penulis menjadi asistennya dan juga mahasiswi selama di kampus ISTTS. Pak Ir. Arya Tandy Hermawan, M.T. selaku rektor, pak Ir. Herman Budianto, M.M. selaku wakil rektor III, bu Yuliana Melita Pranoto, S.Kom. M.Kom. selaku ketua BAU yang telah membantu penulis dalam kelancaran perkuliahan di ISTTS. Pak Dr. Yosi Kristian, S.Kom. M.Kom. selaku kepala program studi dan dosen pengajar yang telah memberikan ilmu pengetahuan *computer intelligence* kepada penulis.
5. Teman-teman perkuliahan yang telah membantu penulis selama pengerjaan tugas akhir dengan saran dan bantuannya, Matthew Gunawan, Georgia Nikita, Lawrence Patrick.
6. Teman-teman perkuliahan ISTTS dari Asisten Pusat Studi Angkatan 2018, Asisten Tetap Laboratorium Komputer ISTTS angkatan 2017, 2018, dan 2019, kelompok belajar kelas A dan C angkatan 2018 yang telah memberikan dukungan dan motivasi selama penyusunan Tugas Akhir kepada penulis (Rommy Christensen Yuwono, Heinrich Wisesa, Sastrabudi, Julianto, Winda, Marvella, James Foong, Nickolas, Chen, Jasong, David, Williandy, Widean, Charles, Kevin Setiabudi, Richard G, Enricho Glenn, William G, Johannes Michael, Brigitta Angeline, Jevon, Felix Halim).
7. Teman-teman di luar perkuliahan ISTTS dari OMK Gereja St. Yakobus (Psyduck, Sel Markus, Koor JYC, Tim IT OMK), teman-teman SMA Kr. Petra 1 Kudeta dan Negus, dan teman-teman Imperials yang telah memberikan doa dan dukungan dari awal sampai akhir penyusunan Tugas Akhir kepada penulis. (Puella, Resia, Erika, Ibel, Dinda, Faustine, Adel, Riani, Nansi, Caroline, Daiva, Aldo, Mitzy, Rere, Everton, Marcella, Esa, SW, Aucky, Jeffry, Aldo Tjondro, Eric Aditya, Vina, Carolin Yoko, Titi Wiliana, Carlis Regina, James Riady, Jeremy Loekman, Evalyn Mayrilia).
8. Dan kepada semua pihak yang tidak dapat disebutkan satu persatu yang telah banyak membantu dalam penyelesaian Tugas Akhir ini.

Demikian skripsi ini yang penulis buat dan semoga dapat bermanfaat bagi penulis pada khususnya dan pembaca pada umumnya. Sangat disadari dalam penyusunan Tugas Akhir ini masih terdapat kekurangan karena keterbatasan pengetahuan, pengalaman, dan waktu penyusunan. Kritik dan saran yang bersifat membangun sangat diharapkan untuk karya tulis ini. Akhir kata dengan segala kerendahan hati penulis mohon maaf kepada semua pihak atas segala kekurangan dan kesalahan yang mungkin pernah diperbuat baik itu disengaja maupun tidak disengaja. Semoga Tuhan selalu memberkati dan menyertai kita semua. Amin.

Surabaya, Juni 2022

Penulis

# DAFTAR ISI

**Halaman**

HALAMAN JUDUL i

HALAMAN PENGESAHAN ii

SURAT PERNYATAAN KEASLIAN iii

ABSTRAK iv

ABSTRACT v

KATA PENGANTAR vi

DAFTAR ISI viii

DAFTAR GAMBAR xi

DAFTAR TABEL xii

DAFTAR RUMUS xv

BAB 1 PENDAHULUAN 16

1.1 Latar Belakang 16

1.2 Tujuan 18

1.3 Batasan Penelitian 18

1.4 Sistematika Pembahasan 19

BAB 2 TEORI PENUNJANG 21

2.1 Word Embedding 21

2.2 Named Entity Recognition (NER) 23

2.3 Transformer 24

2.3.1 Bagian Encoder 26

2.3.2 Bagian Decoder 30

2.4 Neural Network (NN) 32

2.4.1 Multi-Layer Perceptron (MLP) 33

2.5 Sequence to Sequence Models 36

2.6 Hungarian Match 37

2.6.1 Simple Assignment Problem 38

2.6.2 General Assignment Problem 41

2.6.3 Hungarian Algorithm 45

2.7 BRAT 47

BAB 3 NESTED NER DALAM BAHASA INDONESIA 52

3.1 Named Enity Recognition (NER) 52

3.2 Arsitektur Sistem 55

3.3 Dataset dan Tagset Bahasa Inggris 63

3.3.1 GENIA 65

3.3.2 Penjelasan Jenis Tagset Bahasa Inggris 67

3.4 Dataset dan Tagset Bahasa Indonesia 68

3.4.1 Jenis Tagset Bahasa Indonesia 69

3.4.2 Statistika Dataset 72

3.5 Preprocessing 73

3.5.1 Struktur dan Pelabelan Dataset 73

3.5.2 Konversi dan Statistika Dataset 77

BAB 4 SEQUENCE TO SET DALAM BAHASA INDONESIA 82

4.1 Sequence To Set Network 82

4.1.1 Sequence Encoder 84

4.1.2 Entity Set Decoder 97

4.1.3 Bipartite Matching 102

4.2 Library dan Tools 103

4.2.1 PyTorch 103

4.2.2 Hugging Face (Transformers) 104

4.2.3 Google Colab 104

4.3 Modifikasi Metode Sequence To Set Network 105

4.4 Contoh Kasus Penggunaan Sequence To Set 106

BAB 5 UJI COBA 107

5.1 Evaluator Performa Hasil dari Tugas Akhir 107

5.1.1 Macro dan Micro F1Score 107

5.2 Uji Coba Sequence-to-Set Network pada Nested NER 110

5.2.1 Pengaruh Batch Size 111

5.2.2 Pengaruh Learning Rate 112

5.2.3 Pengaruh Gradient Norm 113

5.2.4 Pengaruh Dropout 114

5.3 Perbandingan Metode Span-Based Method 114

5.4 Catatan Uji Coba 116

BAB 6 PENUTUP 118

6.1 Kesimpulan 118

6.2 Saran 119

DAFTAR PUSTAKA 120

RIWAYAT HIDUP 124

LAMPIRAN A CONTOH DATASET NER A-1

LAMPIRAN B ENVIRONMENT UJI COBA B-1

# DAFTAR GAMBAR

Gambar Halaman

[2.1 Contoh Visualisasi (a) Bag of Words dan (b) Word Embedding 23](#_Toc104907033)

[2.2 Contoh Nilai Semantik Tersimpan dari Word Embedding 24](#_Toc104907034)

[2.3 Contoh Pengenalan Entitas Bernama 25](#_Toc104907035)

[2.4 Arsitektur Transformers 26](#_Toc104907036)

[2.5 Representasi Transformer bagian Encoder Decoder 27](#_Toc104907037)

[2.6 Contoh Isi dari Matriks Score 28](#_Toc104907038)

[2.7 Representasi Tiap Layer Dari Arsitektur Transformer 31](#_Toc104907039)

[2.8 Visualisasi Arsitektur MLP dengan Backpropagation 35](#_Toc104907040)

[2.9 Bentuk Ringkasan Arsitektur Seq2Seq 37](#_Toc104907041)

[2.10 Contoh Anotasi Text-Span pada BRAT 48](#_Toc104907042)

[2.11 Contoh Normalization Annotation 49](#_Toc104907043)

[3.1 Contoh Hasil Task Nested NER 56](#_Toc104907044)

[3.2 Arsitektur Sistem 57](#_Toc104907045)

[3.3 Arsitektur Sistem Pra Proses 58](#_Toc104907046)

[3.4 Tampilan File Teks pada BRAT 58](#_Toc104907047)

[3.5 Tampilan Window Fitur Pelabelan 59](#_Toc104907048)

[3.6 Struktur Data (a) Data Anotasi BRAT (b) Data Input Model 61](#_Toc104907049)

[3.7 Arsitektur Sistem Proses Prediksi Nested NER 62](#_Toc104907050)

[3.8 Arsitektur Sistem Pasca Proses 64](#_Toc104907051)

[3.9 Ontologi dan Statistika dari GENIA 67](#_Toc104907052)

[3.10 Contoh Struktur Dataset 77](#_Toc104907053)

[4.1 Arsitektur Sistem Sequence-To-Set Network 84](#_Toc104907054)

[4.2 Visualisasi Arsitektur BERT QnA 86](#_Toc104907055)

[4.3 Arsitektur Character-Level Embedding 93](#_Toc104907056)

[4.4 Arsitektur Recurrent Neural Network (RNN) 94](#_Toc104907057)

[4.5 (Satu) Memory Cell LSTM 96](#_Toc104907058)

[4.6 Arsitektur Bagian Entity Set Decoder 99](#_Toc104907059)

[4.7 Alur Kalkulasi Self-Attention 101](#_Toc104907060)

[5.1 Hasil Uji Coba Jumlah Batch Size 113](#_Toc104907061)

[5.2 Hasil Uji Coba Learning Rate 114](#_Toc104907062)

[5.3 Hasil Uji Coba Gradient Norm 114](#_Toc104907063)

[5.4 Hasil Uji Coba Dropout 115](#_Toc104907064)

# DAFTAR TABEL

Tabel Halaman

3.1 Spesifikasi Dataset GENIA 67

3.2 Daftar Tagset ACE 2005 68

3.3 Jenis Tagset Bahasa Indonesia 71

3.4 Spesifikasi Dataset NER Bahasa Indonesia 73

3.5 Statistika Dataset Tugas Akhir 82

5.1 Contoh Data Confusion Matriks 109

5.2 Penghitungan Micro dan Macro 110

5.3 Hasil Uji Coba F1 Score Terbaik 112

5.4 Tabel Perbandingan F1 Score 116

# DAFTAR SEGMEN PROGRAM

Segmen Program Halaman

[3.1 Preprocessing TXT File 78](#_Toc104907377)

[3.2 Preprocessing ANN File 80](#_Toc104907378)

[3.3 Preprocess Indeks ANN 81](#_Toc104907379)

# DAFTAR RUMUS

Rumus Halaman

[2.1 Rumus Softmax 29](#_Toc104907547)

[2.2 Rumus Attention 29](#_Toc104907548)

[2.3 Rumus Multi-Head Attention 29](#_Toc104907549)

[2.4 Rumus Residual Connection Genap 30](#_Toc104907550)

[2.5 Rumus Residual Connection Ganjil 30](#_Toc104907551)

[2.6 Rumus Fungsi Linear 33](#_Toc104907552)

[2.7 Rumus Fungsi Aktivasi 34](#_Toc104907553)

[2.8 Rumus Loss Function 36](#_Toc104907554)

[2.9 Rumus Nilai Gradien 36](#_Toc104907555)

[2.10 Rumus Tahap Backward Pass 36](#_Toc104907556)

[2.11 Rumus Adequate Budget 43](#_Toc104907557)

[2.12 Rumus Adequate Budget 44](#_Toc104907558)

[2.13 Rumus Adequate Budget (Sederhana) 44](#_Toc104907559)

[2.14 Rumus Teorema 6 44](#_Toc104907560)

[2.15 Rumus Corollary 1 44](#_Toc104907561)

[2.16 Rumus Corollary 1 (Sederhana) 45](#_Toc104907562)

[2.17 Rumus Corollary 1 (Sederhana 2) 45](#_Toc104907563)

[2.18 Rumus Perubahan budget 45](#_Toc104907564)

[4.1 Rumus Fungsi Y 95](#_Toc104907565)

[4.2 Rumus Fungsi H 95](#_Toc104907566)

[4.3 Rumus Fungsi F 95](#_Toc104907567)

[4.4 Rumus Fungsi G 95](#_Toc104907568)

[4.5 Rumus Gate Input, 96](#_Toc104907569)

[4.6 Rumus Gate Forget 96](#_Toc104907570)

[4.7 Rumus Vektor Sell 96](#_Toc104907571)

[4.8 Rumus Hidden Sell 96](#_Toc104907572)

[4.9 Rumus Token Awal 97](#_Toc104907573)

[4.10 Rumus Token Forward BiLSTM 98](#_Toc104907574)

[4.11 Rumus Token Backward BiLSTM 98](#_Toc104907575)

[4.12 Rumus Token 98](#_Toc104907576)

[4.13 Rumus Attention 102](#_Toc104907577)

[4.14 Rumus Multi-Head Attention 102](#_Toc104907578)

[4.15 Rumus Output Multi-Head Attention 102](#_Toc104907579)

[4.16 Rumus Probabilitas Class 103](#_Toc104907580)

[4.17 Rumus Probabilitas 103](#_Toc104907581)

[4.18 Rumus Probabilitas Boundary Kiri 103](#_Toc104907582)

[4.19 Rumus Probabilitas Boundary Kanan 103](#_Toc104907583)

[4.20 Rumus Indeks Optimal Matching 104](#_Toc104907584)

[4.21 Rumus Hungarian 104](#_Toc104907585)

[4.22 Rumus Fungsi Loss 104](#_Toc104907586)

[5.1 Rumus Precision 109](#_Toc104907587)

[5.2 Rumus Recall 109](#_Toc104907588)

[5.3 Rumus F1 Score 109](#_Toc104907589)

BAB I

# PENDAHULUAN

Bab pertama di buku ini akan memberi penjelasan mengenai tugas akhir “Nested Named-Entity Recognition dalam Bahasa Indonesia Menggunakan Sequence-to-Set Network” secara keseluruhan, seperti latar belakang dan tujuan dari penyusunan tugas akhir ini, batasan penelitian yang dilakukan, dan juga dijelaskan sistematika pembahasan pelaksanaan tugas akhir di buku ini.

## Latar Belakang

Cabang ilmu komputer yang mempelajari bagaimana caranya computer dapat memahami dan menganalisa bahasa manusia adalah cabang ilmu Pengolahan Bahasa Alami, umumnya dikenal sebagai *Natural Language Processing* (NLP). Ilmu ini memiliki kegunaan yang luas, seperti memahami bahasa manusia meskipun ada beragam bahasa didunia ini. Dengan memahami arti dari kalimat yang diberikan, computer bisa melakukan beragam *task*. *Task* yang saat ini sudah ditemukan untuk dilakukan seperti asisten virtual dalam *handphone* yang paling umum, bisa melakukan perintah untuk telepon kontak, memainkan musik, dan sebagainya. Bisa juga menerima kata – kata yang diucapkan dan diketik secara otomatis oleh komputer. Komputer tidak hanya dapat mengerti secara makna kalimat, tetapi juga bisa mengenal kata – kata sebagai entitas sendiri. Mengenal dalam suatu kalimat bagian apa yang merupakan subyek, obyek, suatu predikat dan sebagainya. Entitas-entitas dalam bahasa memiliki juga bagian sendiri yaitu entitas bernama, contoh kasus entitas bernama adalah entitas seseorang direpresentasikan nama seseorang, bisa juga suatu lokasi seperti rumah sakit, disebutkan nama dan rumah sakit tersebut.

Namun cabang ilmu NLP masih berkembang dalam bahasa Indonesia. Tidak asing bahwa sebagian besar metode yang digunakan terinspirasi dari metode NLP bahasa Inggris, karena memang bidang NLP untuk perkembangannya yang memimpin adalah bahasa Inggris sendiri. Salah satu contoh penggunaan metode dari bidang NLP yang sering dilakukan penelitian adalah pengenalan entitas bernama, lebih umum disebut *Named Entity Recognition* (NER). *Task* ini berguna untuk mengenal entitas dalam kalimat untuk menemukan informasi yang dapat diolah lebih jauh lagi oleh komputer. Kegunaan mengenal entitas – entitas bernama dalam satu kalimat ini dapat ditemukan dalam teknologi terkini seperti *engine* untuk pencarian dan rekomendasi lebih optimal dan cepat karena NER, atau otomasi penentuan kategori tiket layanan *customer service[[1]](#footnote-1)*. Namun, dari *task* NER ada beberapa kesusahan yang ditemukan, salah satunya adalah pengenalan entitas bernama tetapi yang bersarang. Entitas Bernama yang bersarang adalah penemuan yang baru ditemukan beberapa tahun lalu, dan telah dilakukan beberapa penelitian dengan beberapa metode. Tetapi kekurangan dari penelitian tersebut bukan pada akurasi (karena yang didapat adalah akurasi yang tinggi) tetapi tidak melihat sisi komputasi dalam persiapan dan training model. Kesulitan ini terjadi di penelitian sebelum – sebelumnya tentang permasalahan entitas bernama bersarang seperti metode *sequence-to-sequence*[[2]](#footnote-2)juga metode *span-based[[3]](#footnote-3)*. Tugas akhir ini menggabungkan manfaat metode Sequence-to-Set yang melawan metode span-based dan juga manfaat mendalami penelitian mengenai NER, khususnya yang bersarang, dalam bahasa Indonesia.

Dalam tugas akhir ini, penulis akan melakukan pengenalan entitas yang bersarang yang menangkal kekurangan yang disebut dengan metode Sequence-to-Set Network. Pendekatan metode ini akan menggunakan pendekatan *supervised*, dimana metode akan menggunakan data yang sudah dianotasi untuk proses *training* dan *testing*. Bagian model untuk proses *encoding* akan mengggunakan Bidirectional Encoder Representations from Transformers (BERT) dan Bi-directional Long Short Term Memory (BiLSTM), dan untuk bagian *decoder* nya akan menggunakan *non-autoregressive decoder* yang menggunakan *self-attention* juga *cross-attention* untuk mendapatkan ketergantungan antar entitas. Diharapkan hasil dari penelitian ini, yaitu pengenalan entitas yang bersarang dapat digunakan untuk penelitian linguistik komputasi bahasa Indonesia kedepannya.

## Tujuan

Terdapat beberapa tujuan yang akan dicapai dari penelitian ini yang diharapkan oleh penulis. Bab ini menjelaskan beberapa tujuan yang akan dicapai. Beberapa tujuan tersebut terdiri dari:

• Melakukan pengenalan entitas, baik yang bersarang dan tidak, dari setiap kata dalam kalimat.

• Pembuatan program untuk pengenalan entity bersarang dalam bahasa Indonesia untuk membantu aplikasi dan penelitian NLP kedepannya.

## Batasan Penelitian

Pada bagian ini akan batasan untuk menjelaskan bagian yang akan dikerjakan pada tugas akhir ini. Batasan ini adalah hal-hal ditentukan dalam penelitian tugas akhir ini yang tidak akan dilakukan oleh sistem. Berikut adalah beberapa batasan yang dimiliki pada tugas akhir ini :

1. Program ini hanya menerima input dalam Bahasa Indonesia.
2. Program akan berjalan secara offline.
3. Entitas yang diidentifikasikan sebanyak tujuh yaitu person, organization, date, time, event, location, miscellaneous.
4. Input kalimat hanya dalam 2 - 3 kalimat.
5. Dataset yang digunakan berasal dari tugas akhir Georgia Nikita (218116685)[[4]](#footnote-4), dimana dataset bersumber dari berita dari situs berita CNN Indonesia dan telah dilabel kembali dan ditambahkan pelabelan NER secara bersarang dari dataset tugas akhir Christian Nathaniel Purwanto (214116299)[[5]](#footnote-5) dan Amelinda Tjandra Dewi (214116288)[[6]](#footnote-6).
6. Representasi kata Part-of-Speech Tagging tidak digunakan.
7. Jika ada, akan dilakukan penyesuaian terhadap metode yang digunakan dalam menyelesaikan permasalahan pada tugas akhir ini.
8. Hasil akurasi (F1 Score) dari model yang dibuat memiliki target melebihi akurasi metode perbandingan.

## Sistematika Pembahasan

Dalam subbab ini akan dijelaskan garis besar isi dari setiap bab yang ada pada buku tugas akhir ini. Sistematika ini akan membantu pembaca untuk mengetahui struktur dari pembahasan yang dirangkai penulis untuk memudahkan penjelasan penelitian yang dilakukan. Berikut adalah sistematika pembahasan yang dibuat untuk memudahkan pemahaman isi dari setiap bab secara garis besar.

* BAB I : PENDAHULUAN

Dalam bab ini akan dijelaskan mengenai latar belakang, tujuan, ruang lingkup, batasan penelitian dari tugas akhir buku ini.

* BAB II : TEORI PENUNJANG

Teori penunjang adalah bagian bab yang menjelaskan teori - teori yang menjadi rujukkan informasi yang digunakan dalam pembuatan dann penjelasan penelitian tugas akhir.

* BAB III : ARSITEKTUR SISTEM

Pada bab arsitektur sistem akan menjelaskan alur sistem yang dilewatkan tugas akhir ini. Penjelasan akan mengandung arsitektur secara keseluruhan dan juga untuk tiap bagian arsitektur umum tersebut.

* BAB IV : NESTED NER DALAM BAHASA INDONESIA

Bab ini membawakan penjelasan mengenai Nested NER yang dibutuhkan untuk mengetahui teori, informasi, manfaat dan penggunaan Nested NER dalam bahasa Indonesia khususnya. Dataset dan juga proses *preprocessing* akan dibahas secara mendalam dibab ini sebelum memasuki pembahasan program / metode.

* BAB V : SEQUENCE TO SET DALAM BAHASA INDONESIA

Pada bab kelima dijelaskan mengenai teori, cara kerja, dari metode Sequence to Set tersebut. Tiap bagian dari struktur metode akan dibahas secara rinci, juga dalam bab ini akan menjelaskan modifikasi yang dilakukan untuk tugas akhir ini. Juga ada *tracing* dalam contoh kasus untuk membantu penjelasan cara kerjanya metode.

* BAB VI : UJI COBA

Dalam bab ini akan menjelaskan langkah-langkah yang dilakukan dalam melakukan uji coba serta hasil dari uji coba yang telah dilakukan. Juga uji coba untuk metode perbandingan akan dilakukan dan juga diberikan kesimpulan perbandingannya.

* BAB VII : PENUTUP

Dalam bab ini akan membahas kesimpulan dari tugas akhir dan saran bagi pembaca buku tugas akhir.

BAB II

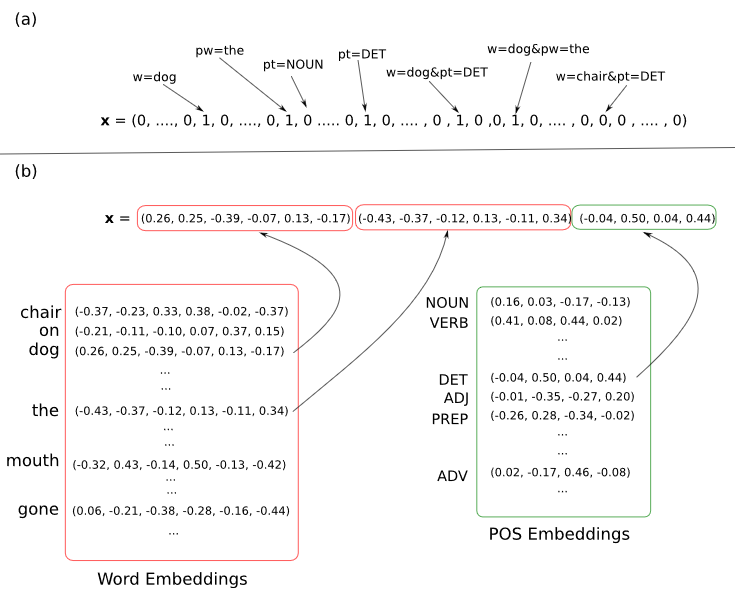
# TEORI PENUNJANG

Bab teori penunjang berisi penjelasan mengenai beberapa teori yang diperlukan sebagai dasar dalam pengerjaan Tugas Akhir ini. Bab ini juga dapat menjadi informasi bantuan bagi pembaca buku tugas akhir ini untuk pemahaman yang lebih baik terhadap hal – hal yang akan disebut atau jelaskan pada bab-bab berikutnya. Teori-teori yang akan disebut dalam bab ini termasuk beberapa teori yang umumnya digunakan dalam penelitian NLP seperti representasi kata (*Word Embedding*), adapun teori *machine learning* seperti Neural Network (NN) dan juga jenis arsitektur NN lainnya yang juga sering digunakan untuk penelitian NER. Beberapa teori lainnya adalah NER, Transformer (Self Attention, Cross Attention), Hungarian Match, BRAT.

## Word Embedding

Representasi kata adalah hal yang penting dalam penelitian NLP karena NLP sendiri mengolah kata agar computer dapat melakukan suatu task. Dan representasi kata memiliki dampak yang signifikan terhadap akurasi hasil dari model yang dibuat. Karena ini, ada beberapa teknik yang diciptakan oleh peneliti untuk menemukan teknik yang dapat membuat representasi kata yang membawa informasi yang penting untuk tiap katanya. Word Embedding, teknik yang pada saat ini menjadi tren dalam NLP karena kemampuannya untuk menyimpan nilai semantik dari satu kata dengan kata lainnya / kata di sekitarnya. [[7]](#footnote-7)

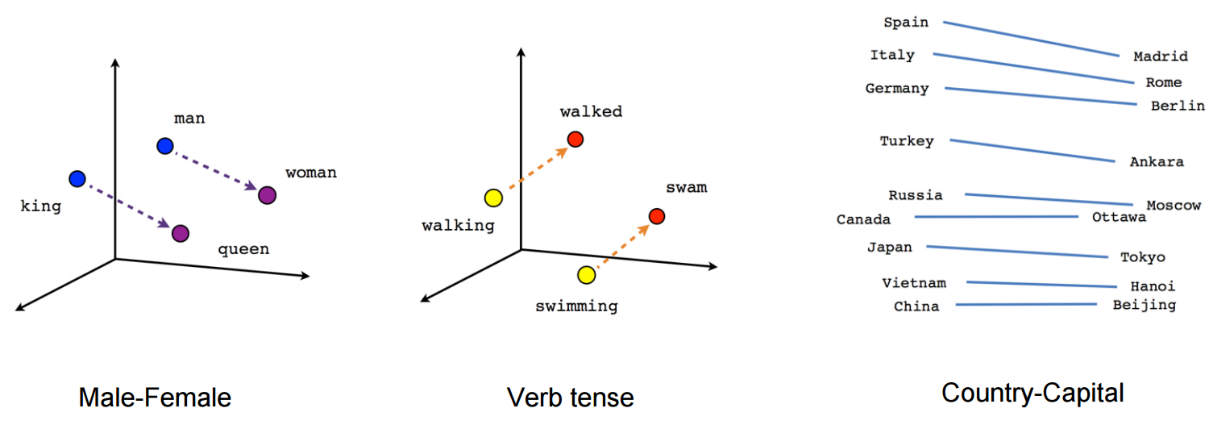
Secara visualisasi, hasil dari word embedding adalah sekumpulan vektor yang berisikan angka riil. Proses konversi kata menjadi vektor. Mengapa word embedding ini menjadi tren dalam penelitian NLP saat ini, salah satunya karena kemampuannya juga untuk memberikan representasi dalam bentuk *dense* (elemen *non-zero*), sedangkan teknik selain word embedding contohnya *Bag of Words*, yang menghasilkan representasi kata *sparse* (elemen yang sebagian besar mengandung nilai nol dan sedikit elemen *non-zero*).



Gambar .  
Contoh Visualisasi (a) Bag of Words dan (b) Word Embedding[[8]](#footnote-8)

Contoh kasus perbandingan representasi kata Bag of Words dengan Word Embedding dapat dilihat pada gambar 2.1. Kata-kata yang perlu di *encode* adalah “the dog”. Jika kita representasikan tiap kata dengan satu dimensi dan memiliki panjang sebanyak kata yang ada dalam satu kosakata dataset tersebut, mengingat kata yang digunakan pasti berjumlah banyak ukuran untuk tiap representasi kata secara keseluruhan akan menjadi besar dan memakan nilai komputasi yang besar. Hal ini yang disebut sebagai representasi sparse. Sedangkan word embedding akan memiliki panjang dimensi yang tidak bergantungan dengan jumlah kata, karena itu tentu akan lebih efisiensi secara komputasi dan memori.

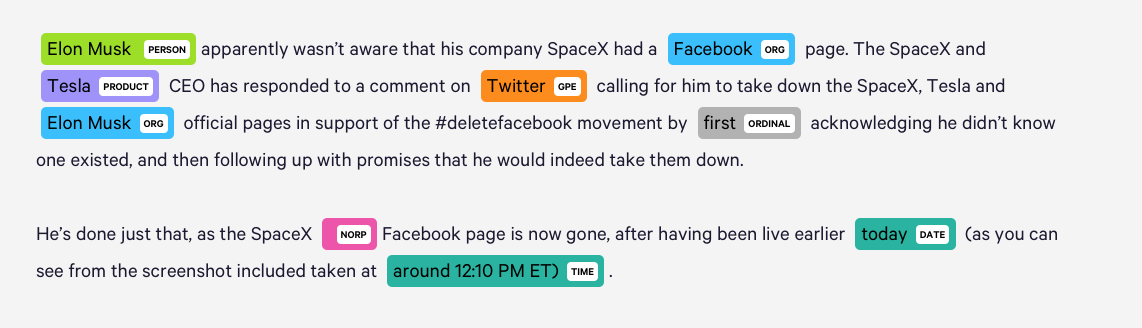
Dengan representasi dense dari word embedding, tiap nilai dari sebuah dimensi memiliki arti. Salah satu yang bisa dicontohkan adalah nilai semantik antar kata lainnya. Dari gambar 2.2 menggambarkan visualisasi nilai semantik yang dapat direpresentasikan. Relasi antar kata yang bisa diberikan seperti lawan kata (Male-Female), kalau dalam bahasa Inggris mampu memberikan nilai relasi bentuk kata kerja yang berbeda (Verb Tense), bahkan juga hubungan secara semantik seperti nama negara dengan nama ibu kotanya (Country-Capital).



Gambar .   
Contoh Nilai Semantik Tersimpan dari Word Embedding

## Named Entity Recognition (NER)

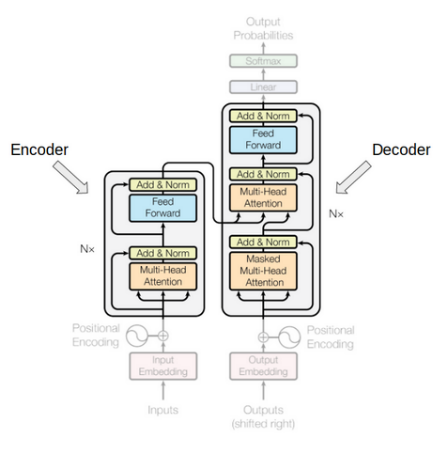
Named Entity Recognition (NER) merupakan task NLP untuk mencari entitas bernama dalam suatu kalimat kemudian menggolongkan entitas tersebut kepada kategori entitas yang ada dari sebuah dokumen/kalimat.[[9]](#footnote-9) Sebuah entitas bernama adalah istilah untuk menyebut sebuah entitas yang ada dalam dokumen teks yang ditulis dengan nama tertentu. Contoh dari kalimat “Perempuan tersebut sedang belajar”, kata perempuan dapat ditandakan sebagai entitas, namun karena entitas tersebut tidak memiliki nama maka tidak termasuk sebagai entitas bernama. Contoh entitas bernama adalah dengan kalimat “Valencia sedang belajar”, maka untuk entitas bernama PERSON (orang) adalah Valencia. Contoh lain entitas bernama adalah lokasi, produk, acara, dan organisasi, waktu dan tanggal. Semua entitas ini dapat memberikan informasi penting dan dapat dimanfaatkan pengguna untuk keperluan analisis lebih lanjut. orang) adalah Valencia. Penggambaran hasil dari task NER dapat dilihat dari gambar dibawah (gambar 2.3).



Gambar .   
Contoh Pengenalan Entitas Bernama[[10]](#footnote-10)

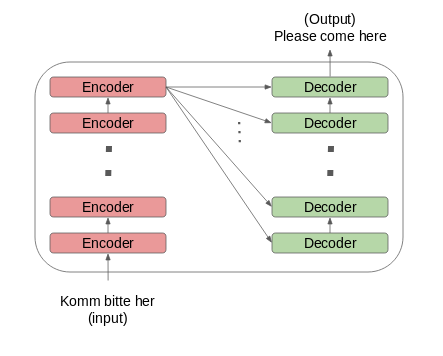
## Transformer

Konsep adanya Transformer (Gambar 2.4) muncul dari model *sequence-to-sequence* (*seq2seq*), di mana model tersebut memiliki tujuan untuk mengerti konversi sebuah sequence, contohnya adalah penerjemahan bahasa dari Inggris ke Indonesia. Model ini melakukan tugasnya dengan baik tetapi ada kesulitan dalam nilai ketergantungan yang berjangka lama. Transformer memiliki cara kerja yang berbeda dengan seq2seq, di mana encoder dan decodernya seq2seq menggunakan urutan selaras RNN, Transformer bergantung pada penggunaan *attention* untuk menghitung representasi input dan output nya.



Gambar .  
Arsitektur Transformers[[11]](#footnote-11)

Arsitektur dari Transformer berorientasi hanya pada encoder dan decoder yang bisa juga diatur jumlahnya. Isi dari bagian encoder dan decoder Transformer sebenarnya adalah encoder dan decoder yang berjumlah banyak (yang merupakan parameter yang bisa diatur tetapi yang ditetapkan pada paper “All You Need is Attention” adalah enam) dan bertumpukan (dapat dilihat pada Gambar 2.5). Secara detail, 1 bagian Encoder memiliki Multi-Head Attention kemudian diikuti Feed Forward Neural Network. Bagian dari 1 decoder memiliki layer yang sama dengan encoder tetapi ditambahkan dengan Masked Multi-Head Attention. Untuk penjelasan lebih rinci mengenai arsitektur Transformer, dapat dilihat pada subbab berikut mengenai bagian encoder dan decoder sendiri.



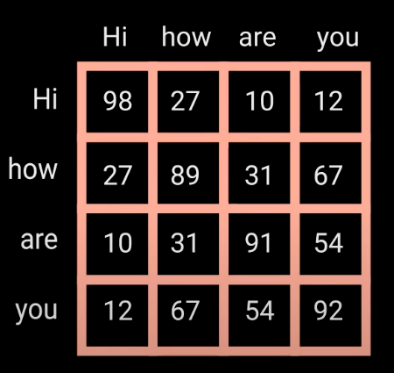
Gambar .  
Representasi Transformer bagian Encoder Decoder

### Bagian Encoder

Bagian encoder memiliki peran untuk merubah urutan input menjadi representasi baru yang memegang informasi dari input tersebut. Struktur bagian ini memiliki 2 bagian besar, *multi-headed attention* dan *feed forward network* (FFN, yang akan dijelaskan pada subbab 2.4) ditambah dengan *residual connection* setelah tiap bagian tersebut. Untuk bagian pertama dari encoder ini adalah multi-headed attention, bagian ini mengaplikasikan ilmu self-attention. Self-attention adalah mekanisme dari ilmu attention dimana *model* dalam machine learning dapat memerhatikan informasi kata-kata sekitarnya dengan jarak yang jauh ke belakang (lebih jauh dibandingkan dengan arsitektur-arsitektur model yang lain seperti RNN, GRU, dan LSTM). [[12]](#footnote-12) Untuk memperjelas, self-attention akan dijelaskan dengan langkah-langkah sebagai berikut:

1. Membuat vektor *Query* (Q), *Key* (K) dan *Value* (V)
2. Membuat matriks *score*
3. Membagi nilai matriks score dengan akar pangkat dimensi vektor key ()
4. Matriks score dilewatkan pada fungsi aktivasi *softmax*
5. Mengalikan matriks score dengan vektor value
6. Menjumlahkan seluruh matriks yang dihitung

Self-attention menggunakan 3 vektor penting yang bernama *Query*, *Key* dan *Value*. Vektor-vektor ini memiliki konsep yang mirip dengan system pengambilan data. Contoh kasus pada website *Youtube*, jika ingin mencari suatu video user akan memasukan sebuah query dan system akan melakukan pencarian berdasarkan query tersebut dengan kumpulan set yang ada (seperti detail video yaitu judul, deskripsi, dsb). Kemudian hasil dari pencarian tersebut akan dikembalikan dalam sebuah nilai (value). Nilai vektor-vektor tersebut diambil dari hasil perkalian word embedding dari input dengan matriks yang dimiliki query, key and value masing-masing yang telah dilewatkan proses training sebelumnya. Gambar 2.6 menunjukkan isi dari matriks score yang memiliki nilai tiap kata dengan kata lain untuk menunjukkan kepentingan relasi antar kata.



Gambar .  
Contoh Isi dari Matriks Score

Selain ketiga vektor itu, ada juga matriks *score* yang dibutuhkan self-attention, matriks ini cukup jelas kegunaannya berdasarkan namanya yaitu untuk memberi skor/nilai terhadap tiap kata. Skor tersebut menandakan seberapa besar fokus pada kata tersebut dengan kata-kata sekitarnya saat ini. Semakin tinggi nilainya, semakin besar fokus yang diberikan. Dan matriks score didapatkan dari perkalian *dot product* antara vektor query dan vektor key.

Matriks score tersebut jika dibiarkan dengan nilai yang relative tinggi, akan menyebabkan efek yang *exploding* kemungkinan besar seperti *exploding gradients* (permasalahan saat training model, dimana model akan memiliki network yang tidak stabil, weight dari model memiliki nilai yang terlalu besar sehingga terjadi overflow bahkan sampai nilai NaN). Karena ini, matriks score perlu diturunkan nilai nya dengan dibagi dengan akar pangkat panjang dimensi vektor key. Nilai matriks score yang baru ini akan membantu untuk menghasilkan nilai gradien yang lebih stabil. Langkah selanjutnya adalah matriks score yang baru akan dilewatkan fungsi aktivasi softmax, rumus ini dapat dilihat pada rumus 2.1. Seperti fungsi aktivasi yang lainnya, tujuannya adalah memuncakkan nilai yang tinggi dan merendahkan nilai yang kecil, dengan representasi tiap nilai antara 0 dan 1. Hasil akhir dari softmax ini akan dipanggil *attention weights*. Attention weights tersebut akan dikalikan dengan vektor value yang menjadi output vektor dari bagian encoder ini. Dengan output terakhir ini, nilai tinggi dari attention weights (output softmax) tadi akan memberi dampak tinggi pada kata-kata yang berarti, dan menurunkan nilai/dampak pada kata-kata yang tidak relevan.

Agar dapat diimplementasikan kepada multi-head attention, tiap operasi self-attention ini (yang disebut juga sebagai *head*) akan terjadi sebanyak N kali dan untuk tiap head mendapatkan vektor query, key dan value yang telah dibagi sebanyak N vektor dan memiliki rumus yang ditulis pada rumus 2.2. Output untuk setiap head akan disambung menjadi 1 vector yang sama. Secara teori, dengan tiap head/self-attention melakukan perhitungan yang berbeda, maka informasi yang didapatkan untuk melakukan *decoding* akan menjadi lebih banyak/besar. Dan dapat disimpulkan multi-head attention adalah self-attention yang dilakukan sebanyak N kali (nilai N adalah parameter yang dapat ditentukan sendiri), multi-head memiliki rumus yang mudah dimengerti dapat dilihat pada rumus 2.3. Dari penjelasan sebelumnya langkah-langkah self-attention didapatkan rumus sebagai berikut:

(.)

(.)

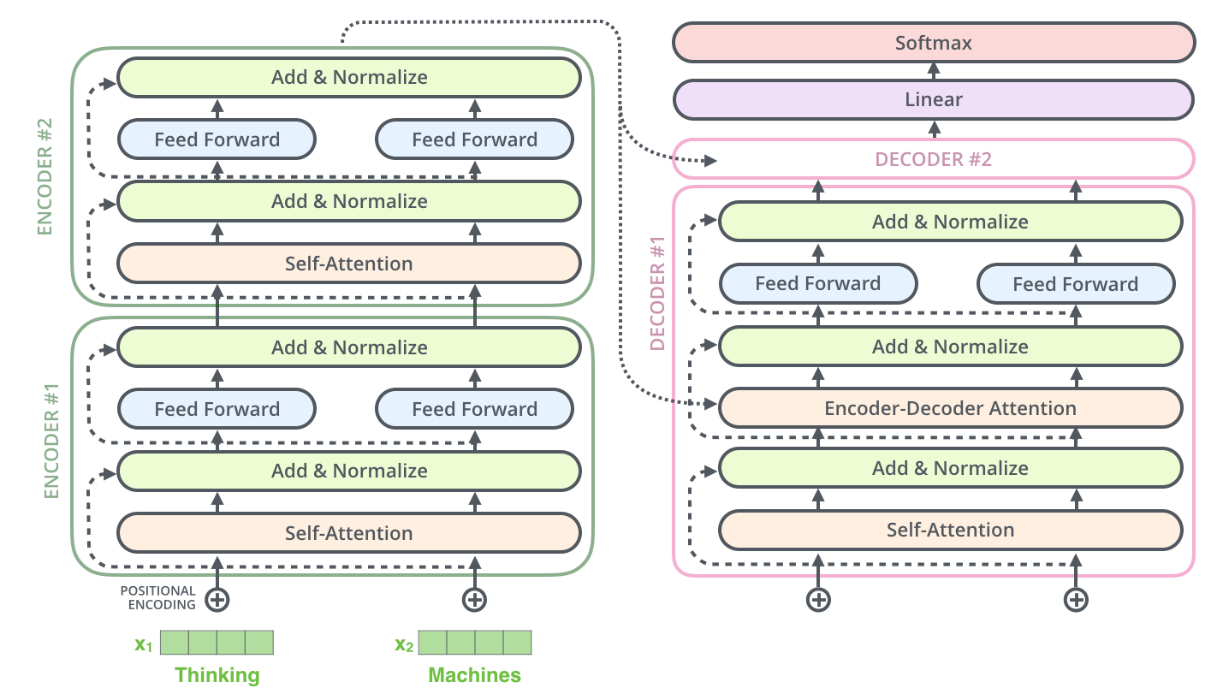
(.)

Sebelum lanjut kepada bagian kedua yaitu FFN, perlu diketahui adanya sublayer setelah multi-head attention dan FFN. Sublayer ini adalah *residual connection* dan normalisasi. Residual connection adalah proses menambahkan *positional encoding* dengan input. Selain di dua sublayer tersebut, positional encoding dapat ditemukan saat sebelum word embedding input dimasukkan kepada multi-head attention. Gunanya positional encoding ini adalah memberi informasi posisi dari input embedding nya. Melihat rumus 2.4 dan 2.5, secara mudah disimpulkan untuk indeks yang ganjil akan dibuatkan vector dengan rumus cos, sedangkan untuk yang genap akan dibuat dengan rumus sin. Setelah residual connection, hasil itu dilanjutkan kepada normalisasi atau LayerNorm.

(.)

(.)

Setelah mengenal residual connection dan LayerNorm, penjelasan akan lanjut kepada FFN. Input dari FFN merupakan output dari residual connection dan LayerNorm, perlu diingat input dari proses residual connection dan LayerNorm tersebut adalah output multi-head attention. Struktur FFN ini adalah dua layer linear dengan fungsi pengaktifan ReLU diantarnya. Dan seperti yang disebut, setelah FFN ini output tersebut dilewatkan proses residual connection dan LayerNorm sekali lagi.



Gambar .  
Representasi Tiap Layer Dari Arsitektur Transformer[[13]](#footnote-13)

Sebagai penutup subbab ini, encoder merupakan bagian yang menghasilkan representasi kata yang baru dengan informasi dari mekanisme attention. Hal ini membantu decoder untuk fokus kepada kata-kata penting yang menjadi input. Dengan banyaknya encoder yang ditumpuk, tiap encoder dapat mempelajari informasi/representasi yang berbeda tiap hitungan. Multi-head attention berguna untuk mencari informasi dengan pandangan attention, residual attention menjadi faktor pembantu training model karena membantu gradien untuk mengalir lewat network dengan lancar. LayerNorm digunakan untuk memberi kestabilan pada network, dengan network yang stabil dapat mengurangi durasi training, dan FFN melakukan proyeksi/analisa terhadap inputnya sehingga berpotensi untuk memberikan representasi dengan informasi yang banyak. Gambar 2.7 adalah penggambaran isi secara detail mengenai layer encoder dan decoder.

### Bagian Decoder

Decoder memiliki tujuan untuk mengambil hasil *encoding* dan generasi *text sequence*. Strukturnya mirip dengan encoder namun diantara multi-head attention dengan FFN terdapat satu multi-head attention lagi tetapi dengan mekanisme masking ditambahkan, juga seperti biasa terdapat sublayer residual connection dan normalisasi setelahnya. Dan pada akhir dari decoder yaitu setelah FFN dan sublayer terakhirnya, diberikan layer linear sebagai *classifier* dan fungsi softmax untuk mendapatkan probabilitas katanya. Dalam Transformer, bagian decoder ini bersifat autoregressive, karena ini dibutuhkan masking dalam salah satu multi-head attentionnya. Tetapi karena pada paper yang dirujuk oleh tugas akhir ini menggunakan decoder yang non-autoregressive, masked multi-head attention tidak akan dibahas.

Bagian decoder dimulai dari input dengan langkah yang sama dengan encoder. Input kata-kata akan dirubah menjadi word embedding, yang kemudian akan ditambahkan dengan positional embedding. Embedding tersebut akan dimasukkan ke layer multi-head pertama tanpa melupakan pembuatan vektor query, key dan value juga matriks score. Kemudian matriks score yang sudah dilewatkan fungsi pengaktifan softmax akan dikalikan denagn query value dan menghasilkan output terakhir multi-head attention yang pertama.

Sebenarnya dalam paper “All You Need is Attention”, seperti yang disebut diawal subbab ini, terdapat masked multi-head attention. Di mana output dari multi-head attention ini terdapat nilai yang masked bertujuan karena sifat autoregressive decodernya, sehingga membutuhkan solusi untuk menghindari informasi yang bocor (*leakage*). Karena paper yang dirujuk pada tugas akhir ini dibuat dengan sifat non-autoregressive decoder, maka permasalahan itu tidak ada, dan tanpa masking decoder ini bisa mendapatkan nilai ketergantungan antar entity nya.

Setelah multi-head pertama, tentu ada sublayer residual connection dan normalisasi. Kemudian dilanjutkan dengan satu multi-head attention tetapi bagian ini menggunakan mekanisme bukan self-attention namun *cross-attention*. Perbedaan dari self-attention dan cross-attention ini hanya berada di pembuatan vektor query, key dan value. Sebelumnya pada self-attention ketiga vektor ini dibuatkan sendiri dengan weight yang sudah ditentukan dan input yang diterima. Tetapi untuk cross-attention, vektor query dan key didapatkan dari output paling akhir bagian tumpukan encodernya. Dan vektor valuenya diambil dari output multi-head attention decoder yang pertama. Dengan cara ini, cross-attention dapat menyocokkan input encoder dengan input decoder dan decoder mempelajari input encoder bagian mana yang perlu difokuskan. Output dari multi-head attention kedua ini akan dilanjutkan kepada layer FFN yang sebelumnya terdapat sublayer yang sudah disebut sebelum-sebelumnya.

Dan pada tahap paling akhir yaitu layer linear dan layer softmax pada akhir decoder. Kedua layer ini ditempatkan setelah seluruh tumpukan decoder telah dilewatkan. Layer linear seperti yang telah disebut, berperan sebagai *classifier* sebesar *class* yang kita miliki. Untuk visualisasi, diberikan contoh 100 kelas untuk 100 kata. Maka output dari layer linear memiliki ukuran sepanjang 100. Output dari layer ini dilewatkan kepada layer softmax yang menghasilkan nilai probabilitas antara 0 dan 1. Dan nilai yang tertinggi adalah kata yang telah diprediksikan. Tahap decoding tidak berakhir disini. Hasil dari output prediksi tersebut akan ditambahkan kepada daftar input decoder yang paling pertama, sehingga input decoder tidak hanya input encoder tetapi juga *concat* dari output yang telah diprediksikan sebelumnya.

## Neural Network (NN)

Neural Network (NN) merupakan arsitektur paling sering digunakan dalam dunia *machine learning* dan *deep learning*. Struktur dari NN terinspirasi dari struktur otak manusia yang menggunakan neuron untuk menyalurkan informasi dari satu ke yang lainnya. Setiap neuron dalam NN direpresentasikan sebagai *node* yang akan menjalankan penghitungan fungsi matematika linear yang memiliki nilai informasi yang didapatkan dari input nya. Dapat dilihat contoh fungsi linear yang digunakan dalam rumus 2.6, *x* adalah nilai input yang diterima, dan *w* sebagai nilai *weight* yang diterima. Nilai node tersebut kemudian dilewatkan dengan *activation function* (fungsi yang meningkatkan sebuah nilai yang melewati sebuah *threshold*, dan menurunkan nilai yang tidak melewati *threshold* tersebut), contohnya pada rumus 2.7 dengan *a* sebagai activation function.

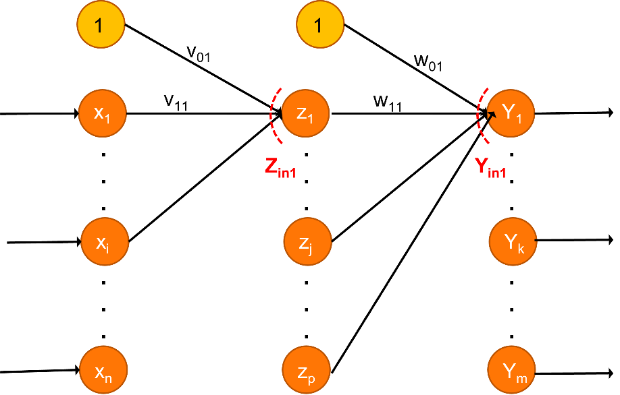
(.)

(.)

Tiap *layer* (lapisan) dari NN ini akan terdiri dari node yang saling berhubungan, dan tiap node memiliki sifat regresi linear (menghitungkan nilai prediksi berdasarkan nilai variabel yang ada sebelumnya). Untuk struktur dari Neural Network sendiri dapat dibagi menjadi 3 bagian / layer: *input layer*, *hidden layer*, *output layer*. Penggambaran dan penjelasan dari ketiga layer tersebut dapat dilihat pada subbab berikut mengenai Multi-Layer Perceptron (MLP).

### Multi-Layer Perceptron (MLP)

Sebelum membahas MLP perlu mengetahui mengenai *Feed Forward Neural Network* (biasa disingkat menjadi FFN). FFN adalah bentuk neural network yang paling pertama dan paling sederhana dibandingkan dengan NN yang sudah berkembang saat ini.[[14]](#footnote-14) Contoh alur ini dapat dilihat dari gambar 2.8. Di mana koneksi antar node di FFN tidak membentuk siklus, namun informasi atau nilai yang diberikan mengarah ke depan (*forward*). Dari input layer, melewati hidden layer dan berakhir pada output layer. FFN perlu diketahui terlebih dahulu karena MLP memiliki struktur yang mirip dengan FFN, tetapi memiliki perbedaan di mana MLP memiliki tiap layer adalah *fully connected layer* dan dalam beberapa kasus setiap layer memiliki jumlah node yang sama.



Gambar .  
Visualisasi Arsitektur MLP dengan Backpropagation

MLP adalah hasil perkembangan[[15]](#footnote-15) dari algoritma yang dibuat Rosenblatt yaitu *Perceptron*.[[16]](#footnote-16) Alasan adanya MLP adalah untuk menghindari kesulitan Perceptron, yang hanya terdiri dari 1 neuron/node, tidak bisa mengaplikasikan data yang non-linear. MLP mengambil sisi FFN di mana input dikombinasikan dengan *weight* yang awalnya diinisiasi secara *random*. Dan mendapat sisi Perceptron yang mengambil nilai input yang dikalikan dengan weight dan diberikan kepada sebuah activation function.

MLP menggunakan *Backpropagation* sebagai metode trainingnya, terdapat 3 tahap besar[[17]](#footnote-17) yaitu: Forward Pass, Loss Calculate, dan Backward Pass. Forward Pass adalah tahap yang telah dibahas sebelumnya dapat bab Neural Network, di mana MLP akan menghitung nilai output dari rumus 2.6 dari layer pertama sampai akhir. Pada Loss Calculate, sesuai dengan namanya, akan melakukan penghitungan jauhnya perbedaan antara nilai output saat ini dengan output sebenarnya. Banyak jenisnya loss function yang bisa digunakan, secara umum MLP menggunakan rumus Cross-Entropy, tetapi penjelasan di bab ini akan menggunakan *Sigmoid Function* (dapat di lihat pada rumus 2.8) untuk loss function nya. Dengan variabel adalah nilai loss function, sebagai target output yang diinginkan, nilai node saat itu, sebagai turunan pertama activation function.

(.)

(.)

Rumus 2.9 adalah rumus penghitungan nilai gradien berdasarkan nilai loss function yang didapatkan. Nilai gradien adalah cara mengevaluasikan nilai loss function, yang kemudian nilai ini akan digunakan dalam proses perubahan weight. sebagai nilai gradien, merupakan nilai *learning rate* dan sebagai nilai node sebelumnya. Tahap yang terakhir adalah tahap backward pass, tahap yang akan melakukan perubahan pada weight sesuai dengan loss function dan gradien yang telah hitung. Untuk penjelasan ini, rumus yang digunakan untuk dapat dilihat pada rumus. Rumus tersebut (dilihat pada rumus 2.10) diambil dari nilai–nilai sebelumnya yang telah dihitung atau ditemukan.

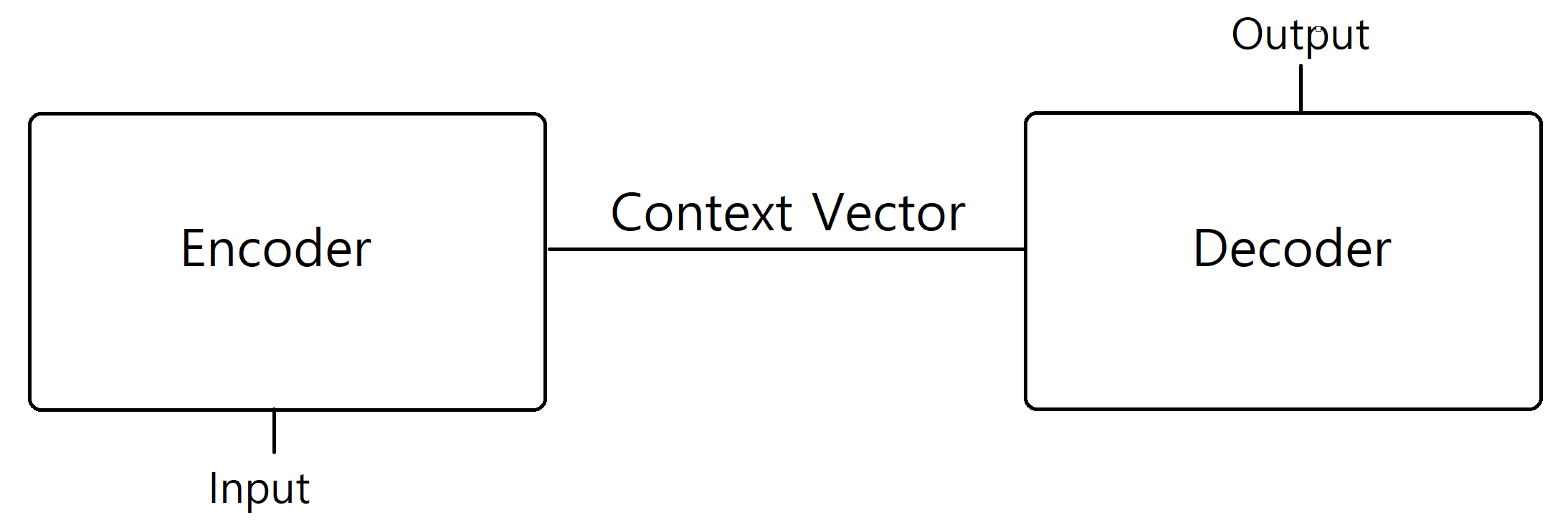
(.)

Rumus-rumus yang telah disebut dihitung untuk tiap node yang ada dalam arsitektur MLP, dan tahap-tahap tersebut dilakukan berulang kali sampai iterasi yang telah ditentukan atau saat perubahan nilai gradien bersifat konvergen. MLP sering digunakan dalam dunia machine learning sebagai *classifier* atau penentuan kelas / kategori dari suatu input karena secara hasil MLP terbukti efektif untuk menjadi classifier. Seperti contohnya penggunaan MLP pada bidang pengenalan suara, pengenalan gambar, dan perangkat lunak terjemahan mesin[[18]](#footnote-18) meskipun pada akhirnya *support vector machines* (SVM) lama kemudian lebih dipilih untuk *classifying*. Namun juga saat ini algoritma backpropagation yang digunakan dalam MLP tetap digunakan karena popularitasnya dalam deep learning.

## Sequence to Sequence Models

Adanya subbab seq2seq karena metode sequence-to-set dari tugas akhir ini terinspirasi dari arsitektur seq2seq. Karena itu, perlu adanya pengenalan mengenai metode yang menginspirasikan metode tugas akhir ini. Sequence-to-sequence (seq2seq) model adalah salah satu jenis yang berasal dari arsitektur Recurrent Neural Network (RNN). RNN sendiri ada untuk menyelesaikan permasalahan dari input sequence (kumpulan kata, huruf, timestep, dsb) karena kemampuannya untuk mengingat informasi beberapa timestep/kata/huruf sebelumnya.

Metode seq2seq dikenalkan oleh pihak Google dan kemudian metode tersebut sering digunakan dalam kasus translasi bahasa, dapat diambil contoh dari salah satu penelitian dari Google sendiri mengenai penggunaan metode seq2seq untuk meningkatkan kualitas translasi bahasa.[[19]](#footnote-19) Atau penelitian lainnya dari Google penggunaannya dalam *speech recognition*. Penamaan sequence-to-sequence didapat dari input dan outputnya yaitu *sequence of items*, contoh dari bentuk dari sequence adalah input kalimat dan output kalimat. Sama seperti penggunaan translasi bahasa di mana input kalimat satu bahasa dan output menjadi bahasa lainnya dengan informasi atau konteks yang sama.



Gambar .  
Bentuk Ringkasan Arsitektur Seq2Seq

Tiap bagian yang tercatat di Gambar 2.9 akan dibahas secara ringkas. Bagian encoder akan mengolah tiap token dari kalimat input untuk mendapatkan informasi sebanyak mungkin dan dijadikan sebuah vektor dengan panjang tetap, biasa disebut *context vector*. Context vector adalah vector yang mengandung nilai/inti/informasi dari input saat ini yang akan diberikan kepada decoder, dengan informasi tersebut decoder menerimanya sebagai input dan mulai menentukan prediksi yang akurat sesuai dengan context vector yang diberikan. Untuk decoder, telah disebutkan tugasnya untuk melakukan prediksi sesuai dengan context vector yang diberikan dan memberikan output sequence.

## Hungarian Match

Terdapat kesulitan dari penghitungan loss saat training model untuk Sequence-to-Set, yaitu cara memberi nilai terhadap prediksi entitas dan kata yang diprediksikan dengan *golden entities* (target prediksi sebenarnya). Hungarian match[[20]](#footnote-20) adalah algoritma yang akan membantu permasalahan ini dengan kemampuannya untuk mencari *optimal* *assignment* (pencocokan/penyesuaian optimal) antara prediksi dengan golden entity. Kata *hungarian* dipilih untuk nama algoritma ini oleh penulis Khun karena inspirasinya terhadap dua algoritma lain yang ditemukan oleh penulis yang kedua-duanya berasal dari Hungarian. Dan tujuan dari algoritma Hungarian adalah untuk memberikan solusi terhadap *assignment* problem. Salah satu contoh umum untuk assignment problem adalah pembagian beberapa tugas dengan sejumlah pekerja dengan suatu nilai perbandingan. Nilai perbandingan itu bisa harga jasanya, atau mungkin akurasi dari pekerjaan itu dapat dilakukan dengan lancar.

Algoritma Hungarian, seperti yang telah dijelaskan sebelumnya, berasal dari dua algoritma solusi untuk dua assignment problem yaitu Simple Assignment Problem dan General Assignment Problem. Terdapat tujuh teorema yang dinyatakan dari Kuhn dimana ketujuh ini terinspirasi dari cara kerja kedua assignment problem asalnya. Pada subbab ini akan dijelaskan seringkas namun sejelas mungkin mengenai tujuh teorema ini, dan penjelasan lebih fokus kepada cara kerja algoritma Hungarian ini.

### Simple Assignment Problem

Penjelasan algoritma pertama yang berhubungan dengan algoritma Hungarian adalah Simple Assignment Problem. Penjelasan mengambil contoh masalah assignment dengan empat individu (dilambangkan i = 1,2,3,4) dengan empat pekerjaan (dilambangkan j = 1,2,3,4). Untuk kemudahan penjelasan, individu akan disebut sebagai ***person***, dan pekerjaan akan disebut sebagai ***job***. Tiap person memiliki kemampuan berbeda-beda untuk mengerjakan suatu job, sehingga disediakan himpunan keterangan tiap person untuk job yang bisa mereka kerjakan. Person pertama dapat mengerjakan job {1,2,3}, person kedua dapat mengerjakan job {3,4} dan dua person terakhir dapat mengerjakan job {4}. Agar informasi ini dapat mudah dipahami disediakan **matriks kualifikasi** (dilambangkan dengan Q). Di mana isi dari matriks ini hanya angka 0 (nol) dan 1 (satu), dan untuk tiap baris adalah representasi tiap person, dan tiap kolom representasi tiap job. Person yang dapat mengerjakan job tersebut akan disebut sebagai ***qualified person*** dan bernilai 1. Yang tidak bisa mengerjakan job tersebut disebut sebagai ***unqualified person*** dan bernilai 0. Sehingga matriks kualifikasi dapat digambarkan sebagai berikut.

|  |
| --- |
|  |

Pertanyaan untuk permasalahan Simple Assignment ini adalah apa nilai terbesar untuk jumlah job yang bisa ditugaskan kepada individu yang memenuhi syarat (qualified person)? Pertanyaan ini juga mempunyai syarat di mana tidak boleh ada lebih dari satu pekerjaan yang diberikan untuk tiap individu. Selama pembahasan Simple Assignment, ditemukan empat teorema yang membantu mencari jawaban untuk pertanyaan tersebut.

Pertanyaan ini dapat diaplikasikan kepada matriks Q yang telah dibuat sebelumnya, dengan pertanyaan baru yaitu berapa banyak angka 1 yang dapat dipilih untuk tiap baris tanpa ada lebih dari satu angka 1 yang dipilih untuk tiap baris. Langsung saja percobaan dimulai dengan melakukan penugasan (assignment) pertama yaitu individu 1 dengan job 3 (tiga), individu 2 (dua) dengan job 4 (empat). Penandaan penugasan tersebut akan menggunakan lambang bintang (\*).

|  |
| --- |
|  |

Dari matriks di atas, dapat dinyatakan bahwa tidak ada penugasan yang bisa dilakukan lagi, sehingga status saat ini untuk penugasan ini adalah **selesai** (assignment ***complete***). Untuk melanjutkan algoritma, assignment yang complete perlu dilakukan proses ***transfer***, agar menghasilkan status penugasan dari matriks kualifikasi menjadi tidak selesai (***incomplete***). Proses transfer adalah pemindahan tugas dari satu individu kepada individu lain, dalam transfer pertama ini yang dipindahkan adalah individu 1 kepada job 1 dan individu 2 kepada job 3. Pemindahan job yang berbeda karena yang terjadi dalam proses transfer adalah tiap individu akan dipindahkan kepada job berikutnya yang individu tersebut bisa melakukannya. Contoh untuk individu 1, karena job berikutnya (job 4) individu 1 tidak memenuhi syarat sehingga penugasannya berpindah kepada job 1. Hasil matriks kualifikasi saat ini seperti berikut.

|  |
| --- |
|  |

Hasil Q matriks terbaru ini menyatakan kolom job 4 dapat ditugaskan kepada individu 3 atau 4. Dengan ini, meskipun untuk individu 1 dapat melakukan transfer juga kepada job 2, assignment ini dianggap complete (gambaran matriks terbaru berada dibawah paragraph ini). Sampai tahap ini, ditemukan beberapa pernyataan yang dapat mendukung teorema pertama nanti. Sebagai lemma pertama, dinyatakan bahwa jika ada person yang ditugaskan kepada suatu job artinya antara person tersebut atau job tersebut penting (atau kata yang akan digunakan adalah *essential*), namun yang essential adalah salah satu antara individu atau job, tidak bisa kedua-duanya.

|  |
| --- |
|  |

Diberikan juga *corollary* (bukti yang mendukung) untuk lemma pertama tadi yaitu jumlah person yang ditugaskan pasti sama dengan jumlah essentialperson / job. Arti dari kata essential/arti kepentingan dari suatu person atau job akan digunakan untuk algoritma Hungarian dan dijelaskan kepada dua lemma berikutnya. Lemma kedua, mengatakan jika transfer dilakukan dan suatu person yang sudah ditugaskan, dapat ditugaskan kepada job lain, artinya person tersebut penting/essential. Lemma ketiga menyatakan bahwa jika transfer dilakukan dan job yang sebelumnya sudah ditugaskan kepada seorang person dan masih bisa ditugaskan oleh individu lain, artinya job tersebut yang penting/essential. Dan dari ketiga lemma tersebut disimpulkan menjadi teorema pertama yaitu : untuk sebuah penugasan/assignment, jika suatu proses transfer menghasilkan penugasan lengkap/complete assignment, untuk setiap individu/person yang memenuhi syarat untuk sebuah pekerjaan/job, antara individu tersebut atau pekerjaan tersebut memiliki sifat penting/essential, bisa juga kedua-duanya yang penting/essential.

Jika diperhatikan, setiap proses penugasan (assignment) yang dilakukan menghasilkan dua skenario. Pertama adalah hasil dari complete assignment, yang kedua adalah setidaknya satu individu dapat ditugaskan setelah sebuah proses transfer. Teorema kedua muncul karena setidaknya maksimal n individu dapat ditugaskan, maka ini membuktikan pasti ada penugasan yang selesai (complete assignment) setelah semua kemungkinan transfer terjadi.

Tetapi ada masalah yang masih belum dipertimbangkan dalam penugasan ini yaitu adanya suatu anggaran yang dibutuhkan untuk menugaskan seorang individu kepada sebuah pekerjaan. Sebuah anggaran dianggap memadai jika, untuk setiap individu yang memenuhi syarat untuk suatu pekerjaan, diberikan satu unit anggaran, entah itu untuk individunya saja, pekerjaannya saja, ataupun keduanya diberikan satu unit. Penjelasan anggaran dan anggaran yang memadai akan digunakan istilah *budget* dan *adequate budget*. Karena adanya anggaran yang dipertimbangkan, teorema ketiga ini membantu memahami perannya anggaran dengan masalah penugasan ini. Teorema ketiga menyatakan bahwa total alokasi/pembagian (bisa juga disebut *allotment*) dari anggaran/budget, harus lebih dari sama dengan nilai terbesar dari jumlah pekerjaan yang bisa ditugaskan kepada individu yang memenuhi syarat (qualified persons).

Pertimbangkan sebuah penugasan/assignment apa pun dianggap selesai/complete setelah semua transfer dilakukan (teorema kedua), dan anggara/budget yang dialokasikan adalah satu unit untuk individu atau pekerjaan yang penting/essential, dan dialokasikan sebanyak nol unit untuk individu atau pekerjaan yang tidak penting/inessential. Dari corollary 1, dapat dibuktikan teorema 4 yaitu pasti ada adequate budget dan assignment yang jumlah alokasi budget yang sama dengan jumlah job yang ditugaskan kepada qualified persons. Dan dengan teorema 3 menyiratkan bahwa teorema 4 ini optimal, ditemukan untuk jawaban Simple Assignment Problem yaitu nilai terbesar untuk jumlah job yang dapat ditugaskan kepada qualified persons adalah sama dengan jumlah alokasi budget terkecil dari budget yang memadai (adequate budget). Sebuah penugasan/assignment dianggap optimal **jika dan hanya jika** penugasan tersebut selesai dan setelah semua kemungkinan transfer telah dilakukan.

### General Assignment Problem

General Assignment Problem memiliki permasalahan yang sama yaitu permasalahan penugasan antara sejumlah n individu (i = 1, … n) kepada sejumlah n pekerjaan (j = 1, …, n). Perbedaan antara General Assignment Problem dengan Simple Assignment Problem adalah adanya sebuah matriks baru yang dipertimbangkan dalam permasalahannya yaitu matriks dengan isinya bilangan positif yaitu rating matriks (dilambangkan ). Dan pertanyaan dari General Assignment Problem adalah penugasan manakah yang total dari penjumlahan rating adalah nilai rating terbesar?

Adapun *dual problem* yang lebih memerhatikan kepada permasalah anggarna yang harus memadai (adequate budget). Dual problem merupakan permasalahan yang berhubungan sangat dekat dengan permasalahan pertama, sehingga solusi optimal dari satu masalah secara otomatis memberikan solusi optimal untuk yang lain. Suatu anggaran dinyatakan memadai jika jumlah dari nilai alokasi budget u untuk tiap individu dan alokasi budget v untuk tiap pekerjaan mendapatkan hasil yang lebih besar sama dengan rating job tersebut. Dalam perumusan matematika, sebuah adequate budget dinyatakan dalam rumus 2.11.

(.)

Sehingga muncul pertanyaan baru khusus untuk dual problem ini yaitu berapa nilai terkecil dari total alokasi budget (contoh total alokasi budget adalah ) untuk budget adequate? Dari Simple Assignment Problem, teorema 3 memiliki hubungan dengan teorema berikutnya teorema 5 yaitu, niali dari jumlah alokasi adequate budget apa pun adalah lebih besar sama dengan jumlah rating dari assignment apa pun. Agar lebih jelas, diberikan pembuktian (*proof*) untuk teorema 5. Karena setiap individu dan pekerjaan muncul tepat hanya sekali tiap penugasan/assignment, total alokasi anggaran (tiap individu atau pekerjaan) tepat sama dengan nilai jumlah seluruh alokasi anggaran. Anggaran assignment tersebut menjadi memadai/adequate dan bernilai tidak kurang dari total jumlah rating individu yang ditugaskan kepada suatu pekerjaan. Pelambangan dalam rumus matematika, rumus 2.12 ditemukan dengan syarat anggaran yang dihitung termasuk anggaran yang memadai (adequate budget) dan rumus tersebut lebih disederhanakan menjadi rumus 2.13.

(.)

(.)

Dampak dari adanya pernyataan dari rumus 2.13 adalah pernyataan baru yaitu, jika suatu adequate budget dan assignment yang bisa diatur agar total jumlah dari alokasi anggaran bisa sama dengan nilai jumlah rating penugasan tersebut, artinya nilai-nilai tersebut adalah solusi dari assignment problem dan juga dual problemnya. Secara singkat, peraturan untuk menyatakan suatu individu memenuhi syarat (qualified person) untuk suatu pekerjaan adalah jika , jika tidak bisa memenuhi syarat tersebut, individu tersebut tidak memenuhi syarat (*not qualified*). Dapat dilihat secara langsung teorema berikutnya (teorema 6), jika semua n individu dapat ditugaskan (assigned) dan memenuhi syarat (qualified person), dan anggaran penugasan tersebut memadai (adequate budget), artinya penugasan dan anggaran tersebut adalah solusi untuk permasalahannya dan nilai jumlah alokasi anggaran sama dengan nilai jumlah rating penugasan. Hasil dari teorema 6 dapat ditulis ulang secara matematis dari rumus 2.14.

(.)

Namun, jika **tidak semua** n individu tidak bisa ditugaskan (meskipun qualified dan memiliki adequate budget), sehingga budget tersebut perlu diperbaiki dengan sebuah prosedur. Sebelum prosedur perbaikan anggaran/budget dijelaskan, perlu diingat bahwa nilai alokasi untuk anggaran harus **positif** agar nilai rating untuk individu atau pekerjaan tidak negatif. Asumsikan beberapa lambang yang akan digunakan, *m* sebagai nilai terbesar jumlah individu ditugaskan (sehingga m < n). *r* sebagai jumlah essential person (individu yang bisa ditugaskan), dan *s* sebagai jumlah essential job (pekerjaan yang bisa ditugaskan). Dengan lambang yang sudah disebut, corollary 1 menyatakan rumus 2.15 .

(.)

Pernyataan corollary 1 dapat mendefinisikan perbaikan budget dapat dinyatakan:

(.)

(.)

Inti dari rumus 2.16 dan 2.17, perubahan *u* (budget individu) adalah untuk individu pertama sampai individu ke *r* (jumlah individu penting/essential) akan memiliki nilai yang sama, namun untuk seluruh individu lainnya akan dikurangi (dengan syarat nilai akan non-positif/lebih dari sama dengan nol). Dan untuk perubahan *v* (budget pekerjaan) adalah untuk pekerjaan pertama sampai pekerjaan ke *s* (jumlah pekerjaan penting/essential) akan ditambahkan nilai satu, pekerjaan lainnya akan tetap dengan nilai awalnya.

Setelah prosedur perubahan/perbaikan budget, perlu diperiksa pakah budget terbaru ini memadai/adequate dan total alokasi anggaran telah berkurang dari sebelumnya. Perubahan dianggap gagal apalah bila telah berkurang dan tidak berganti, dengan kegagal ini menyatakan bahwa individu/pekerjaan tersebut tidak penting/inessential. Dan sifat inessential ini ditegaskan oleh teorema 1 karena . Perubahan budget dianggap adequate karena dapat dinyatakan dalam rumus 2.18.

(.)

Dapat dibuktikan juga bahwa jumlah alokasi anggaran telah berkurang karena *n* – *r* dan beratmbah sebanyak *s*, dapat diringkas bahwa telah berkurang sebanyak *n* – (*r + s*) = *n – m > 0*. Semua ini menyimpulkan teorema 7 yaitu, jika paling banyak m < n individu bisa ditugaskan, memenuhi syarat dan memiliki anggaran yang memadai, maka nilai jumlah alokasi anggaran bisa berkurang dengan nilai integral yang positif. Dan berakhir pada penemuan jawaban untuk pertanyaan General Assignment Problem, nilai terbesar dari jumlah rating yang memungkinkan adalah sama dengan nilai terkecil jumlah alokasi anggaran dari anggaran apapun yang memadai.

### Hungarian Algorithm

Seperti yang pernah disebut sebelumnya, algoritma Hungarian mendapatkan inspirasi dan menggabungkan dari dua permasalahan dan teorema Simple dan General Assignment Problem. Bagian dari subbab ini akan lebih frontal dalam menjelaskan bagaimana cara algoritma ini berjalan. Untuk beberapa istilah yang sudah disebut sebelumnya akan digantikan sesuai dengan penggunaannya di algoritma ini. Berikut adalah istilah baru yang mereferensikan istilah sebelumnya, anggaran memadai (atau nominal yang memenuhi syarat adequate budget) akan disebut sebagai ***cover*** dan posisi (i,j) untuk matriks kualifikasi yang dapat dianggap bisa ditugaskan (qualified) disebut sebagai ***marked/mark***. Untuk posisi (i,j) matriks yang tidak memenuhi syarat untuk ditugaskan, akan disebut sebagai ***blank***. Satu set/kumpulan marks disebut sebagai ***independent*** dimana tiap mark yang ada tidak boleh berada di kolom atau baris yang sama. Jumlah anggota dari independent (marks) sejumlah *m*, maka baris/kolom sejumlah *m* tersebut dapat dipilih (hal ini berhubungan dengan pernyataan dari Simple Assignment Problem pekerjaan ditugaskan kepada individu yang memenuhi syarat yang disini menjadi independent marks).

Jika diberikan nilai cover (budget) untuk matriks rating (dilambangkan R), maka ditemukan sebuah kumpulan independent mark terbesar. Jika set tersebut memiliki sejumlah n (jumlah individu/pekerjaan) marks, maka sesuai dengan teorema 6, mark tersebut adalah yang diinginkan untuk assignment tersebut. Jika set tersebut kurang dari n marks, maka cover dari set tersebut perlu diperbaiki atau diubah (terorema 7). Berikut adalah langkah-langkah untuk algoritma Hungarian:

1. Menghitung jumlah rating tiap baris dan kolom, dan mencari nilai terbesar dari semua baris (dilambangkan sebagai b), dan nilai terbesar dari semua kolom (dilambangkan sebagai a).
2. Untuk menentukan nilai dari anggaran ditentukan dari perbandingan nilai a dan b sebelumnya. Jika b a, maka = dan =0. Untuk sebaliknya, a > b, maka = dan =0, dimana lambang untuk budget individu adalah *u*, dan budget pekerjaan adalah *v*.
3. Seperti Simple Assignment Problem, diperlukan matriks kualifikasi (dilambangkan sebagai Q), mulai nya langkah ini dapat dinyatakan proses assignment dilakukan. Dari nilai-nilai budget yang telah ditentukan digunakan untuk menentukan nilai q dengan peraturan, diisi angka 1 apabila , jika tidak akan diisi angka 0.
4. Kemudian isi dari matriks Q akan diberikan marks dan menjadi independents (sekumpulan marks). Cara untuk menentukan mulai dari mana penandaan marks ini bergantung pada nilai a dan b yang digunakan dalam langkah 2. Jika b a, maka periksalah baris yang isi dari kolomnya memiliki angka 1 yang belum ditandakan bintang (\*) dan nilai dari kolom tersebut 1. Jika ditemukan, angka 1 tersebut diberikan lambang bintang (\*) dan menjadi 1\*. Untuk syarat a > b, dilakukan hal yang sama namun kolom dan baris bertukar peran.
5. Setelah disediakan matriks Q yang telah melewati proses penandaan, akan dicarikan baris (pekerjaan/job) dan kolom (individu/person) yang dapat ditandakan sebagai penting/essential. Peraturan untuk penandaan ini adalah jika suatu kolom memiliki isi 1\*, maka otomatis satu kolom tersebut ditandakan sebagai essential. Namun, apabila dalam satu baris terdapat 1\* dengan proses transfer yang dapat dilakukan, maka baris itu ditandakan sebagai essential.
6. Langkah ini akan menjalankan semua transfer yang dapat dilakukan dari matriks Q dari kolom-kolom yang memenuhi syarat.
7. Perubahan nilai budget untuk individu atau pekerjaan ditentukan dari **jumlah inessential rows.**
8. Seluruh langkah ini diulang sampai jumlah anggota independent sebanyak n individu. Jika masih belum mencapai n, proses mulai dari langkah 3.

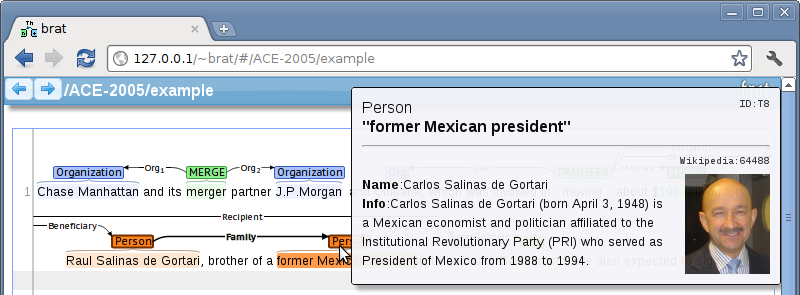
## BRAT

Untuk subbab BRAT ini akan fokus terhadap BRAT sebagai alat anotasi secara keseluruhan baik fitur yang digunakan maupun yang disediakan namun tidak digunakan di tugas akhir ini. BRAT adalah *web-based tool* anotasi untuk dokumen teks. Tujuan adanya BRAT ini untuk memudahkan anotasi dan pemberian catatan/*notes* pada suatu dokumen dengan memberikan juga dokumen anotasi/catatan dengan format yang tetap dan dapat dibaca oleh komputer. Salah satu contoh penggunaan dapat dilihat dari Gambar 2.10. Pada gambar diilustrasikan anotasi untuk anotasi *text-span* (yang digunakan untuk anotasi data Named Entity Recognition) dan anotasi *relation* (yang digunakan untuk anotasi data Relational Extraction). Beberapa jenis fitur untuk anotasi disediakan seperti, fitur *n-ary* associations, BRAT menyatakan dapat menghubungkan sejumlah anotasi lain yang berpartisipasi dalam peran tertentu. Contohnya n-ary associations adalah anotasi untuk jenis anotasi TRANSFER akan dihubungkan secara relation dengan 3 jenis anotasi yang lain dengan relasi yang berbeda-beda (dengan jenis anotasi Money berelasi Money, dengan Person berelasi Beneficary, dengan Org berelasi Recipient). Dan konfigurasi penentuan kategori anotasi, tipenya, bahkan juga peraturan (contohnya peraturan untuk relation Family harus terkoneksi dengan anotasi jenis Person) dapat dilakukan dengan mudah.



Gambar .  
Contoh Anotasi Text-Span pada BRAT

Adapun fitur *normalization annotations* (gambar 2.11), fitur ini mengidentifikasi entitas yang dapat ditemukan dunia nyata yang dirujuk dalam teks yang dianotasikan merupakan bagian penting dalam menganalisis makna teks. Namun untuk fitur ini dibutuhkan persiapan data yang membantu BRAT mengetahui entitas yang ada di dunia nyata. Langkah-langkah tersebut dapat ditemukan dalam halaman https://brat.nlplab.org/normalization.html. Dan fitur terakhir yang menjadi tambahan kecil pada alat anotasi ini adalah penulisan catatan untuk tiap anotasi yang diberikan.



Gambar .  
Contoh Normalization Annotation[[21]](#footnote-21)

Sudah ada banyak penelitian task yang dilakukan dengan bantuan BRAT, dan dalam web BRAT disebutkan beberapa task yang berhasil dan masih berjalan menggunakan BRAT dalam penelitiannya. Beberapa dari contoh penggunaan BRAT dalam task yang ada adalah *entity mention detection*, *event extraction*, *coreference resolution*, *normalization*, *chunking*, *dependency syntax*, *meta-knowledge*. Entity mention detection merupakan anotasi entitas dengan cara text-span yang telah disebut sebelumnya. Event extraction, anotasi yang digunakan untuk mengetahui relasi antar entitas. Coreference resolution adalah task yang dapat mengetahui relasi coreference antar entitas. Normalization telah dijelaskan sebelumnya. Chunking, task membagi teks menjadi anotasi yang tidak tumpang tindih dengan anotasi yang lainnya, anotasi chunking ini sering digunakna untuk mengetahui jenis kata dari suatu teks seperti label NP (Noun Phrase). Dependency syntax merupakan analisis sintaksis, tugas untuk menetapkan *binary relations* antara kata-kata untuk menandai hubungan yang bergantung pada kata utama mereka. Meta-knowledge, tugas mengidentifikasi bagaimana pernyataan faktual harus ditafsirkan, sesuai dengan konteks tekstualnya. Seluruh kegunaan/task yang disebut pada bagian ini dapat ditemukan informasi selengkapnya (contoh visualisasinya, dataset/penelitian yang memberi contoh sesuai) pada halaman website BRAT https://brat.nlplab.org/examples.html#corpus-examples-brat.

Selain fitur utama yang telah disebutkan sebelumnya, fitur yang digunakan secara umum dari BRAT juga banyak dan sangat berguna dalam perannya sebagai alat anotasi. Tugas akhir ini membatas penggunaan fitur sesuai yang dibutuhkan yaitu anotasi text-span saja. Juga dengan penggunaan fitur umumnya seperti visualisasi, alat anotasi BRAT (intuitive editing), zero setup, easy export, always saved, configurable, search. Untuk mempermudah penjelasan dan mempersingkat, penjelasan dapat dibaca sebagai berikut:

* Comprehensive visualization

Konsep “*what you see is what you get*”: semua aspek anotasi yang berdasarkan representasi secara visual dengan cara yang intuitif.

* Intuitive editing

Cara anotasi berbasis penggunaan *mouse* pada umumnya dan menggunakan gerakan intuitif yang familiar dari editor teks, perangkat lunak presentasi, dan banyak alat lainnya. Untuk menandai rentang teks, cukup pilih dengan mouse dengan menekan, menahan, dan menandakan kata-kata yang diinginkan atau dengan mengklik dua kali pada sebuah kata.

* Integration with external resources

Dapat menggunakan data eksternal, bukan berasal dari BRAT, seperti Freebase, Wikipedia, dan Open Biomedical Ontologies.

* Zero setup

BRAT dibuat sepenuhnya dengan teknologi web standar, dan tidak perlu meng-instal perangkat lunak lokal atau plugin browser apa pun untuk menggunakannya.

* Annotation of texts in any language

Memiliki *full Unicode support*, sehingga mendukung hampir 100 skrip yang berbeda.

* Integrated annotation comparison

BRAT mencakup sejumlah fitur untuk membandingkan beberapa set anotasi untuk dokumen yang sama, termasuk perbandingan otomatis untuk mengidentifikasi dan menandai perbedaan dan visualisasi berdampingan.

* An address for each annotation

Setiap anotasi BRAT dapat memiliki URL *address* secara unik. Bersama dengan URL server, memudahkan apabila memerlukan akses langsung terhadap suatu jenis anotasi tertentu (didapatkan pada tombol Link di menu modal apabila menekan atau melakukan anotasi baru).

* Integration with automatic annotation tools

Adanya fitur integrasi dengan metode canggih untuk dukungan anotasi dasar seperti pemisahan kalimat (Inggris dan Jepang) dan tokenization (Jepang).

* High-quality visualization at any scale

Visualisasi anak nakal didasarkan pada Scalable Vector Graphics (SVG), yang dapat di-render dalam detail dan presisi yang diinginkan

* Easy export in multiple formats

Anotasi yang dibuat di brat dapat dieksport dengan mudah dalam format standoff sederhana yang dapat dengan mudah dianalisis, diproses, dan dikonversi ke format lain.

* Always saved, always up to date

BRAT menghilangkan risiko kehilangan anotasi apabila terjadi *crash*, lupa untuk menyimpan pekerjaan, atau bahkan kegagalan total komputer annotator dengan melakukan operasi edit dari annotator ke server brat saat selesai.

* Real-time collaboration

Arsitektur dan desain dari BRAT client-server memungkinkan beberapa annotator untuk bekerja secara bersamaan pada kumpulan dokumen yang sama, atau bahkan pada dokumen yang sama, melihat pengeditan satu sama lain.

* Detailed annotation process measurement

Secara opsional, BRAT dapat dikonfigurasi untuk merekam waktu yang tepat saat annotator membuka dokumen, setiap tindakan edit, dan bahkan waktu yang dihabiskan untuk memilih jenis yang akan ditetapkan ke anotasi setelah memilih tempat untuk menempatkannya.

* Rich set of annotation primitives

BRAT menyediakan serangkaian kategori dasar anotasi yang beragam : anotasi untuk text-span, binary relations, equivalence classes, n-ary associations dan attributes.

* Fully configurable

Semua konfigurasi anotasi menggunakan bahasa konfigurasi sederhana. Setiap kumpulan dokumen memiliki konfigurasinya sendiri, memungkinkan satu server BRAT untuk meng-host banyak proyek dengan target anotasi yang berbeda. Selain itu, sebagian besar visualisasi seperti font, anotasi warna kotak dan busur serta kepala panah dan gaya menggambar busur dapat dikontrol secara detail menggunakan spesifikasi gaya HTML/CSS yang terdokumentasi dengan baik dan dikenal luas.

* Always validated

BRAT memiliki validasi anotasi yang mampu memeriksa semua batasan yang dapat didefinisikan dalam konfigurasi ekspresifnya.

* Search

BRAT mengimplementasikan serangkaian fungsi lengkap untuk mencari dokumen atau koleksi dokumen untuk anotasi jenis apa pun dengan serangkaian batasan yang dapat dikonfigurasi secara terperinci.

* Concordancing

BRAT mendukung key-word-in-context (KWIC) untuk tampilan search berdasarkan kata

BAB III

# NESTED NER DALAM BAHASA INDONESIA

Pada bab ini akan dibahas mengenai pendahuluan mengenai NER, juga teori nested NER juga contoh-contoh penggunaan tugas NER dalam kehidupan sehari-hari. Selain itu, dataset yang digunakan dalam tugas akhir ini juga akan dibahas dengan detail. Penjelasan baik untuk dataset bahasa Inggris yang utama digunakan dari penelitian, maupun dataset bahasa Indonesia yang digunakan dalam penelitian ini. Dataset akan dibahas struktur/bentuk, visualisasi dan juga jenis tagset/jenis entitas yang ditentukan dalam tiap dataset. Bab ini juga ada subbab bagian pra proses (*preprocessing*) dengan rinci untuk mengetahui apa saja yang perlu dimodifikasi dari dataset mentah menjadi dataset yang akhir agar dapat diterima untuk training model.

## Named Enity Recognition (NER)

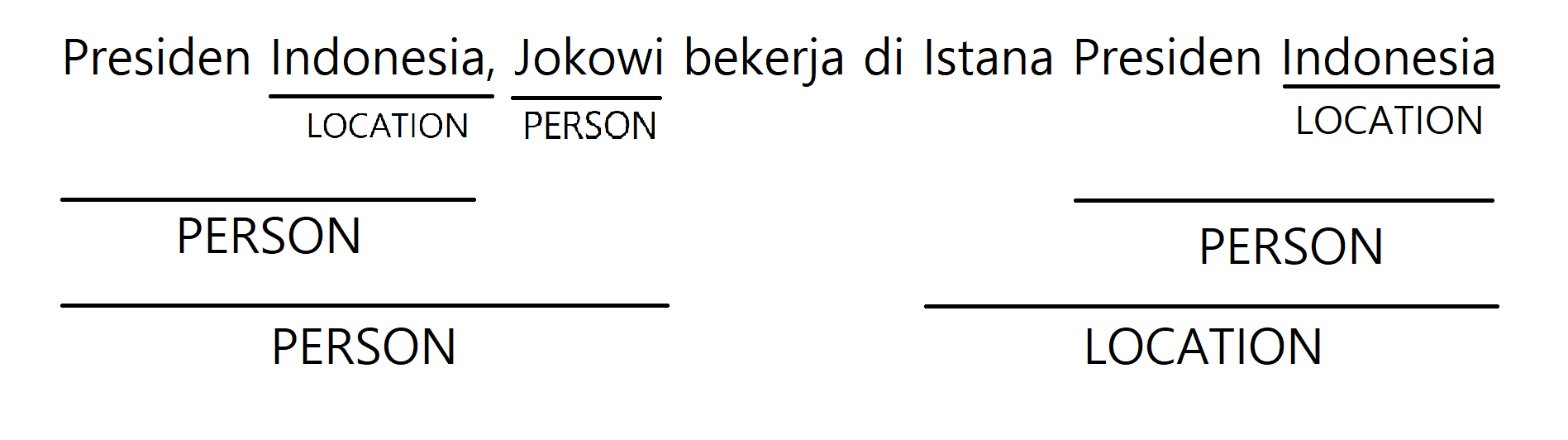
Sebelum mencoba mengerti apa itu tugas pengenalan named entity (atau NER) secara mendalam, perlu diketahui terlebih dahulu apa yang dapat disebut sebagai sebuah named entity. Istilah named entity awalnya dianggap memiliki relasi dekat dengan pembahasan mengenai *rigid designators* oleh Kripke.[[22]](#footnote-22) Namun diskusi named entity mengarah kepada rigid designators menjadi terlalu filosofis. Sehingga dalam penelitian dicarikan penjelasan yang lebih jelas dan ringkas. Sampai saat ini belum ada persetujuan yang resmi dari bidang NLP mengenai definisi resmi NER. Tetapi dalam skenario penelitian NER dapat disimpulkan definisi umum untuk tugas NER maupun arti sebuah named entities ini. Oleh pihak CoNLL 2002 sendiri, named entities adalah frase yang mengandung nama oleh seseorang, suatu organisasi, sebuah lokasi, sebuah waktu dan sejumlah kuantitas.[[23]](#footnote-23) Dengan arti yang jelas dan ringkas, tugas pengenalan named entities menjadi persetujuan secara umum definisinya seperti yang telah dinyatakan pihak CoNLL 2002.

Untuk mempermudah dan memperjelas penjelasan, akan dijelaskan dengan perbedaan pengenalan suatu entitas dengan entitas bernama (named entity). Suatu entitas adalah kata-kata yang termasuk kategori jenis entitas namun tidak memiliki suatu nama. Contoh kalimatnya seperti, “Anak kecil itu bermain sepak bola di lapangan”. Entitas yang dapat diketahui adalah anak kecil sebagai orang/*person*, lapangan dapat diketahui sebagai lokasi/*location*. Namun dengan tugas NER kedua entitas itu tidak berlaku karena tidak memiliki nama. Untuk tugas NER, kalimat yang tepat sebagai contoh adalah “Budi bermain sepak bola di Lapangan Gelora 10 November”. Dengan entitas bernama person untuk Budi, dan entitas bernama jenis location untuk Lapangan Gelora 10 November.

Pada tahun 1997 ada konferensi yang diadakan yang menghasilkan dokumen Message Understanding Conference (MUC-7).[[24]](#footnote-24) Isi dari dokumen ini menjadi salah satu standar umum dalam dunia NLP untuk menentukan kata-kata yang dianggap merupakan named entities dan juga standar jenis-jenis entitas yang akan umumnya sering dipakai ke depannya. Dijelaskan dalam MUC-7 terdapat dianotasikan ekspresi bernama yang unik (seperti person, location, *organization*), ekspresi bilangan (seperti *monetary values, percentage*) dan ekspresi waktu (*date, time*). Dan terdapat banyak petunjuk untuk contoh kasus menganotasikan named entities, namun untuk tugas akhir ini tidak seluruhnya mengikuti petunjuk MUC-7 ini. Contohya untuk poin A.1.3, menyatakan jika sebuah entitas yang mengandung substring sebuah entitas bernama yang lain tidak akan dinyatakan entitas bernama sendiri karena tidak dapat dipecahkan. Poin ini tidak berlaku untuk tugas akhir ini karena pada saat ini sedang meneliti metode untuk memecahkan poin tersebut. Contoh dari poin yang dimaksud dalam bentuk kalimat adalah “Arthur Anderson Consulting”, dimana Arthur Anderson Consulting merupakan named entity organization, namun Arthur Anderson tidak dianggap person karena menurut MUC-7 tidak dapat dipisahkan.

Nested NER merupakan task yang tidak jauh berbeda dengan task NER sendiri. Bahkan persentase nested NER pada beberapa dataset signifikan, contohnya pada GENIA terkandung 17% nested NER dan 30% untuk dataset ACE. Namun task ini sering dilewatkan, dimana penelitian pada umumnya hanya fokus pada task NER bukan yang nested/bersarang. Jenny Rose Finkel dan Christopher D. Manning[[25]](#footnote-25) menjelaskan dengan baik mengenai perkiraan alasan mengapa hal ini terjadi. Finkel dan Manning menjelaskan bahwa dipercaya ada dua penyebab NER lebih meningkat daripada nested NER yaitu karena alasan untuk kepraktisan dan alasan teknologi.

Fakta bahwa penelitian terhadap NER disengaja tidak fokus pada nested NER hanya karena kepraktisan adalah karena sebagian besar dataset NER, seperti CoNLL, MUC-6, and MUC-7 NER, memang memutuskan untuk menganotasikan named entity dengan string terpanjang. Semisal untuk kata Bank Indonesia, akan dianotasikan Bank Indonesia dengan named entity organisasi, namun Indonesia tidak akan dianotasikan, di mana seharusnya kata Indonesia seharusnya dianotasikan lokasi. Dan mengapa dipercaya karena teknologi adalah karena tersedianya dataset yang tidak memfasilitasi named entity yang bersarang sehingga banyak penelitian yang mengembangkan untuk pengenalan *flat named entity* (named entity yang tidak bersarang). Contoh nested NER berada pada Gambar 3.1.



Gambar .  
Contoh Hasil Task Nested NER

Penelitian terhadap nested NER sebelumnya sudah ada seperti penelitian sekitar tahun 2006-2007. Meskipun begitu, nested NER masih belum diperjelas sebagai topik yang perlu diteliti lebih dalam atau topik yang perlu dikembangkan dengan metode selain penggantian dataset.[[26]](#footnote-26) Sehingga Finkel dan Manning melakukan penelitian dengan nested NER sebagai topik dan judul utamanya. Setelah tahun 2009 sedikit demi sedikit penelitian nested NER lebih sering dilakukan meskipun tidak banyak.

## Arsitektur Sistem

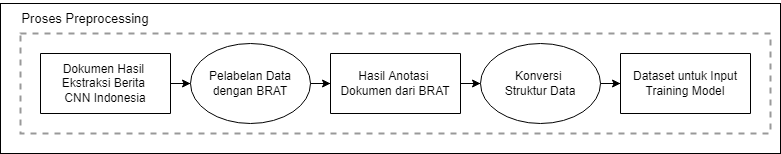
Subbab ini akan menjelaskan arsitektur sistem secara keseluruhan selama pengerjaan tugas akhir. Arsitektur sistem ini akan diulas secara detail pada subbab-subbab berikutnya. Subbab arsitektur sistem akan membahas aliran data dari sub-proses pertama sampai akhir untuk mengetahui gambaran secara umum sistem ini. Visualisasi arsitektur sistem dapat dilihat dari Gambar 3.2 dan dapat dilihat dari gambar tersebut terjadi kurang lebih lima buah proses. Dan untuk setiap beberapa proses telah dikelompokkan menjadi kelompok sub-proses agar dapat dimengerti proses tersebut dilakukan dengan tujuan tertentu. Semisal untuk pra proses, mungkin untuk prediksi NER atau pasca proses untuk programnya.

Gambar 3.2 memiliki pembagian tiga sub-proses, pra proses, proses prediksi nested NER, dan pasca proses. Pra proses adalah proses paling pertama, dan memiliki dua proses yang berhubungan dengan dataset. Proses ini fokus pada memroseskan dataset menjadi input yang sesuai untuk model program. Dari dataset mentah dari ekstraksi berita CNN Indonesia, akan diubah menjadi struktur data sesuai dengan struktur untuk model yang digunakan di tugas akhir ini. Kemudian lanjut kepada sub-proses berikutnya adalah proses prediksi nested NER. Proses yang dilakukan pada sub-proses ini adalah bagian dari model. Metode-metode yang ditentukan untuk training model sehingga dapat melakukan prediksi nested NER. Dan metode ini akan dijelaskan lebih detail pada bab-bab berikutnya. Setelah proses paling akhir proses ini, yaitu penghitungan *loss*, proses ini menghasilkan model final yang telah memiliki informasi dan kemampuan untuk menentukan nested NER. Dan model akhir ini akan menjadi input utama dari proses berikutnya yaitu pasca proses. Sub-proses bagian pasca proses ini bertujuan untuk menyiapkan model yang diterima, untuk menerima input dan output yang dapat dilihat oleh user / orang. Untuk detail dari masing-masing arsitektur sistem akan dijabarkan dalam subbab-subbab berikutnya.



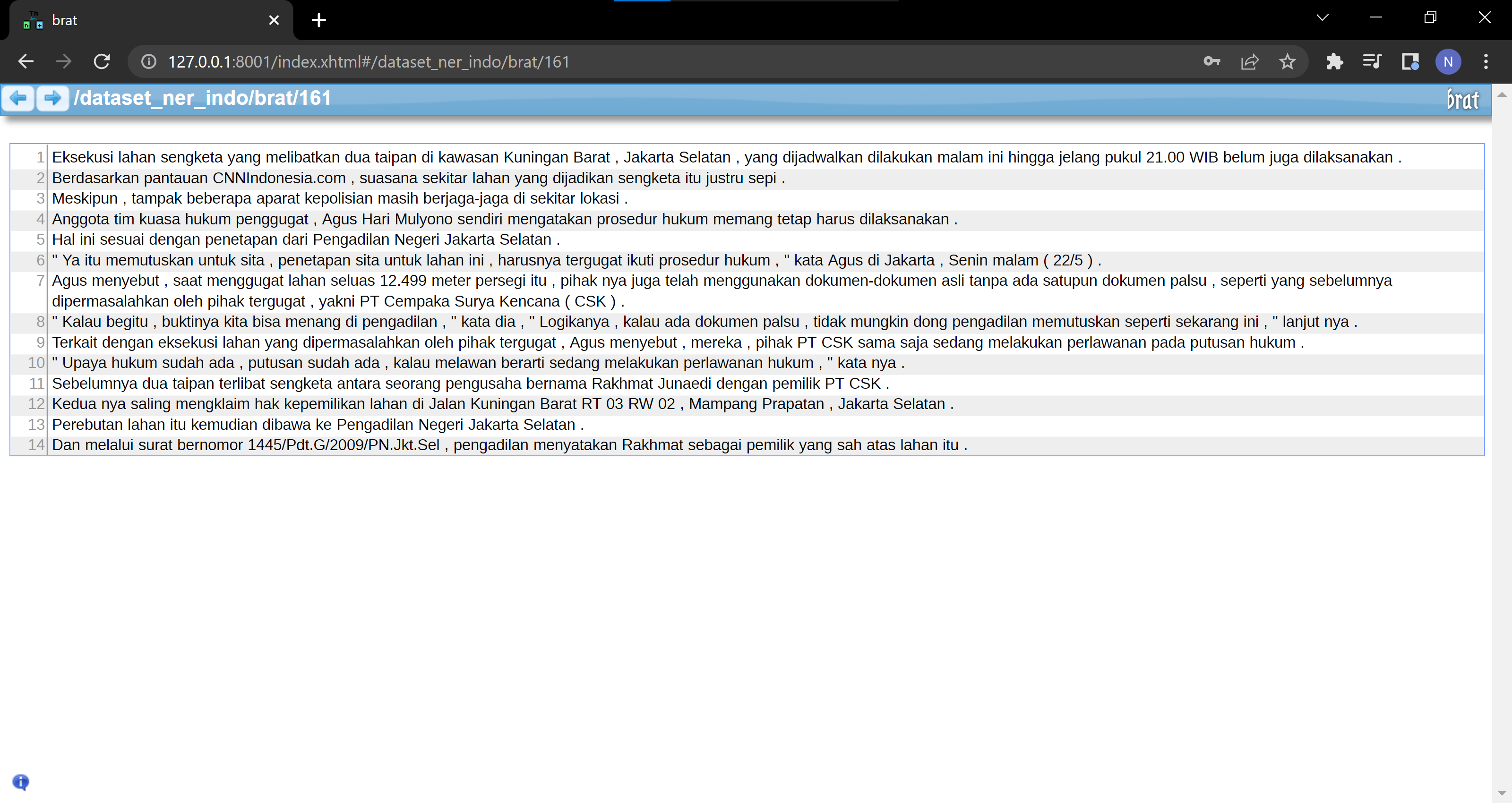
Gambar .  
Arsitektur Sistem

Penjelasan berikut akan membahas tahap pra proses yang akan dilakukan dalam arsitektur sistem ini. Pembahasan pada bagian ini akan menjabarkan mengenai proses yang akan dilakukan dalam tahap ini serta input dan output yang akan digunakan dan dihasilkan dalam tahap pra proses ini. Gambaran setiap tahap pra proses dapat dilihat pada Gambar 3.3. Dapat dilihat terdapat dua proses (digambarkan dengan lingkaran) dan tiga data (digambarkan dengan persegi panjang) yang akan digunakan dalam proses ini. Setiap data akan diberikan penggambaran dan akan diberikan penjelasan bagaimana data itu akan diterima atau bagaimana akan diolah.



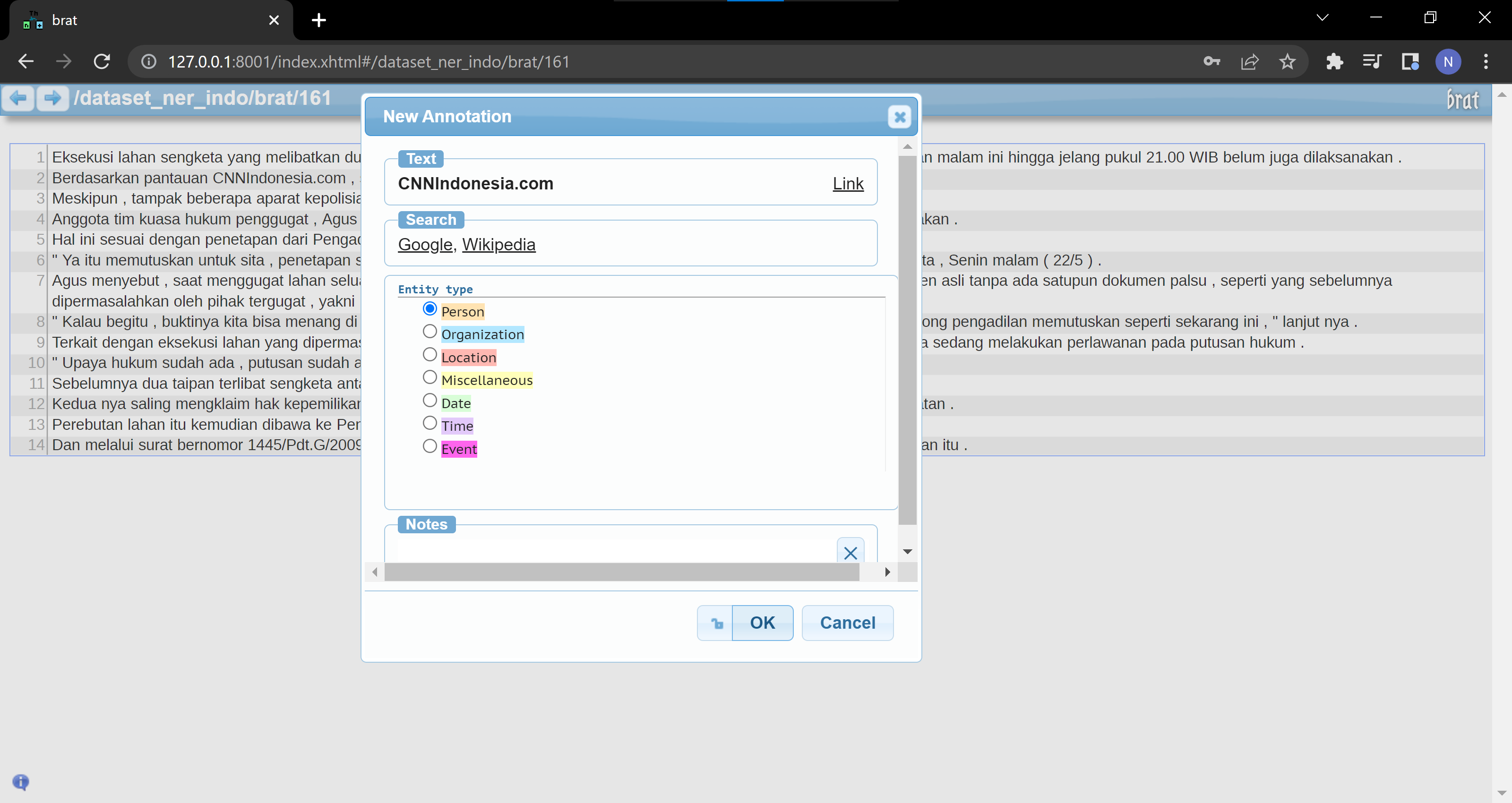
Gambar .  
Arsitektur Sistem Pra Proses

Input pada tahap pertama pra proses adalah hasil ekstraksi berita dari CNN Indonesia. Ekstraksi ini berupa format *text file* (format txt) kemudian data tersebut akan menjadi input untuk proses pelabelan data dengan alat anotasi BRAT. Seperti yang telah dijelaskan, pada bab kedua subbab BRAT, data ini akan diterima dan setiap kalimat akan dipisah dengan fitur *linebreak* dari BRAT sendiri (pemisahan setiap kalimat menjadi 1 baris sendiri dalam sebuah text file). Dan hasil file yang telah diubah ini akan menjadi teks yang muncul pada halaman anotasi BRAT. Isi dari file teks akan dilampirkan dalam halaman BRAT seperti pada Gambar 3.4.



Gambar .  
Tampilan File Teks pada BRAT

BRAT dipilih sebagai alat anotasi karena kemudahan kegunaannya dalam pelabelan. Karena penggunaan fitur yang intuitif dan mirip dengan perintah mouse dengan teks pada umumnya, cara pelabelan sebuah / beberapa kata dapat dilakukan dengan menekan dan menggeser *mouse*, memilih kata yang ingin dilabelkan. Kemudian akan muncul *window* jenis-jenis label yang ingin dipilih. Window tersebut juga memiliki fitur lain yang membantu seperti fitur Link untuk menyimpan URL address dari kata yang dianotasikan tersebut untuk menuju ke kata tersebut dengan mudah dan cepat. Fitur search dengan tombol Google atau Wikipedia untuk membantu mencari arti yang relevan dari kata tersebut. Fitur Notes untuk memberi catatan pada anotasi tersebut. Jika selesai melakukan pelabelan atau fitur lain, cukup tekan tombol OK untuk menyimpan hasil perubahan. Tampilan window ini dapat dilihat pada Gambar 3.5.

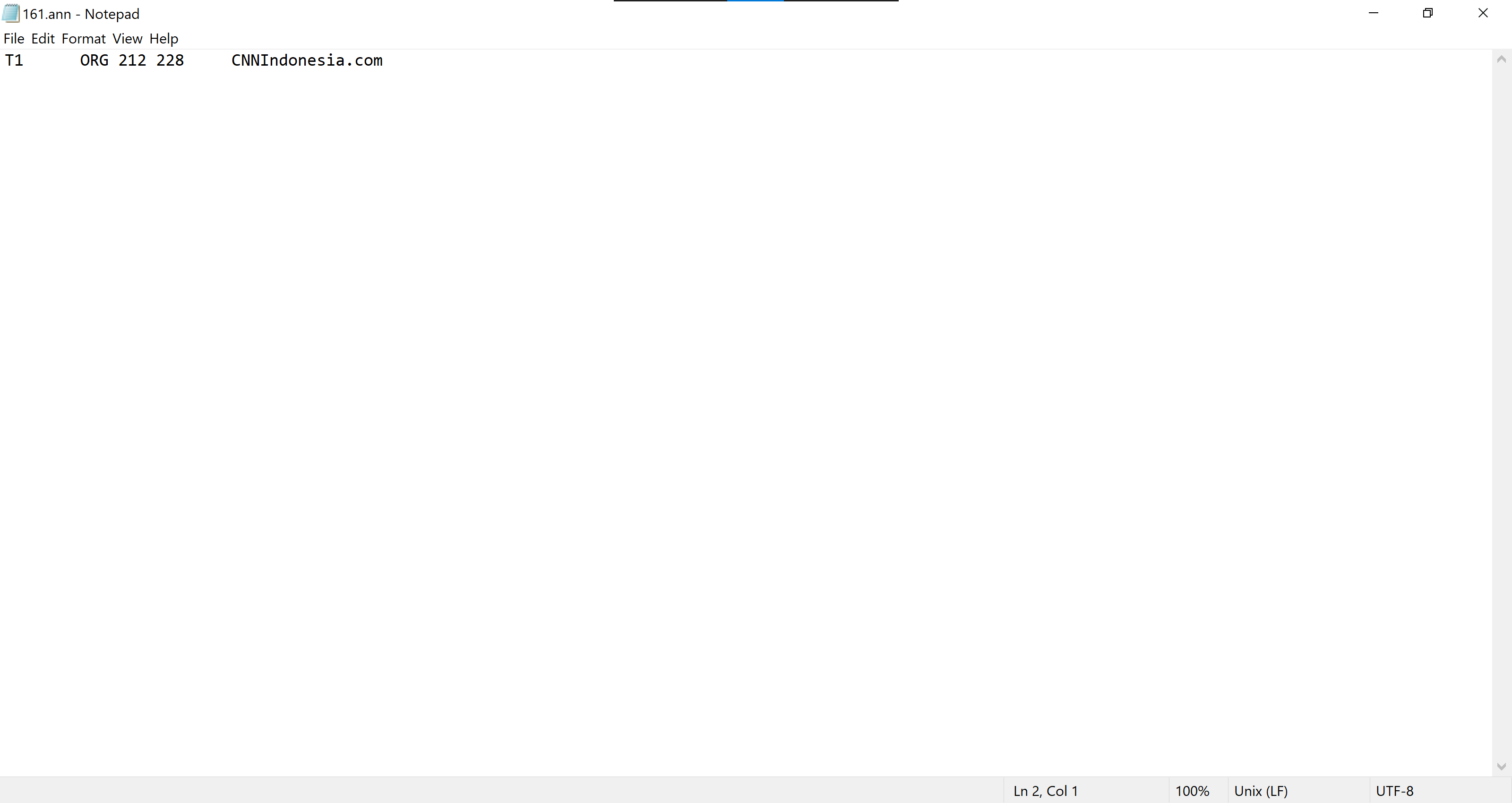


Gambar .  
Tampilan Window Fitur Pelabelan

Contoh pada Gambar 3.4 adalah satu file teks/dokumen dari dataset penelitian. Setiap dokumen memerlukan file teks agar dapat dilabelkan, dan untuk pencatatan pelabelan/anotasi yang telah dilakukan akan disimpan dalam jenis file yang dibuatkan sendiri oleh BRAT secara otomatis yaitu jenis file ann (annotation). File teks ini akan menjadi output dari proses pertama yaitu pelabelan data berita CNN Indonesia juga menjadi input untuk proses berikutnya yaitu konversi struktur data. Visualisasi struktur data dari BRAT dan struktur data untuk input training model akan ditampilkan pada Gambar 3.6.

Struktur data dari file teks anotasi BRAT memiliki 4 jenis catatan. Gambar dibawah akan diambil sebagai contoh untuk menjelaskan keempat jenis catatan dari BRAT. Catatan terdiri dari T1, ORG, 212, 228 dan CNNIndonesia.com. T1 merupakan kode pelabelan untuk dokumen tersebut, tiap catatan pelabelan akan mendapatkan kode tersebut dan sifatnya unik dan *incremental* (T1, T2, T3, dst). Catatan ORG adalah jenis label yang dipilih, dalam tugas akhir ini, ORG adalah label/jenis entitas organisasi. Angka 212 dan 228 adalah indeks pertama dan terakhir dari dokumen itu untuk mengambil kata/huruf yang dilabel untuk pelabelan tersebut. Sehingga jika dari dokumen diambil kata-kata dari 212 sampai dengan 228 akan mendapatkan kata-kata yang sama dengan catatan berikutnya yaitu CNNIndonesia.com.

Dapat dilihat pada Gambar 3.6 (b), jenis filenya struktur data input model bentuk JSON (Java Script Object Notation). Penjelasan atribut dari JSON hanya atribut yang digunakan dalam tugas akhir ini. Tokens adalah *array* kata-kata dalam 1 kalimat yang sedang dianotasikan saat ini. Entities adalah semua label entitas yang ditemukan dalam 1 kalimat itu, dengan bentuk *array of JSONs* yang memiliki 3 atribut. Atribut start dan end adalah index awal dan akhir dari kalimat saat ini, dan type sebagai catatan label/jenis entitas apa untuk kata tersebut. Meskipun tugas akhir ini memiliki batasan tidak menggunakan POS Tag, namun model tetapi meminta POS Tag untuk mengolah data. Karena itu atribut pos itu akan berisi token *unknown* hanya sebagai data pengisi agar tidak memunculkan *error* dalam training program. Ltokens dan rtokens memiliki tujuan yang sama dengan tokens perbedaannya adalah ltoken merupakan kalimat sebelum kalimat saat ini. Dan rtokens adalah kalimat setelah kalimat saat ini.



**(a)**



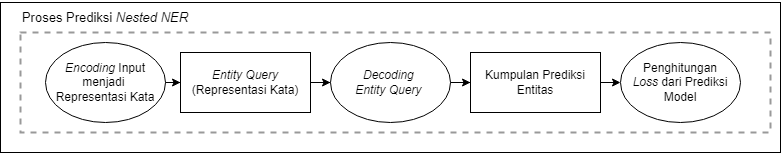
(b)

Gambar .  
Struktur Data (a) Data Anotasi BRAT (b) Data Input Model

Dalam tugas akhir ini telah dibuatkan program untuk merubah struktur data BRAT menjadi struktur data Sequence-to-Set, karena tidak adanya program konversi tersebut. Setelah struktur data dari file teks BRAT diubah menjadi struktur data yang dibutuhkan, menghasilkan dataset dengan struktur sesuai untuk menjadi input training model. Dengan ini, penjelasan persiapan dataset dalam bagian pra proses selesai dan dataset tersebut akan dilanjutkan kepada proses berikutnya yaitu proses prediksi nested NER.

Berikut menjelaskan proses prediksi nested NER dengan melakukan training dengan metode Sequence-to-Set Network. Gambar 3.7 menggambarkan visualisasi aliran system untuk prediksi nested NER ini. Terdiri dari tiga proses dan dua data yang digunakan. Untuk detail seperti metode apa yang digunakan dalam tiap proses, modifikasi atau parameter yang digunakan dalam tiap proses tidak akan dijelaskan dalam bab ini namun pada bab-bab kedepannya.

Aliran sistem ini akan dimulai dengan sebuah proses yang menerima input dari proses sebelumnya yaitu tahap pra proses. Input merupakan dataset yang sudah diubah sesuai struktur data yang ditentukan oleh model Sequence-to-Set Network. Proses ini mengambil inputnya dan melakukan encoding, hal ini bertujuan untuk membuat representasi kata yang baru dengan informasi yang terdapat dari input dataset saat itu. Hal ini dicapai dengan bantuan berbagian jenis embedding yang saling digabungkan. Representasi kata yang baru ini dibutuhkan agar membantu computer untuk training model dengan representasi dalam bentuk yang dapat mudah diolah yaitu angka bukan kata-kata.



Gambar .  
Arsitektur Sistem Proses Prediksi Nested NER

Representasi kata tersebut merupakan output dan juga input pertama dari sub-proses ini, dimana representasi kata itu akan bersama dengan sebuah set vektor yang *trainable* yaitu *Entity Query/Queries*. Secara teori, isi dari representasi kata ini adalah gabungan berbagai jenis embedding yang ada. Jenis embedding ini akan dijelaskan pada sub-bab berikutnya. Dan data ini, dari urutan input, memiliki panjang *l x d*, dimana *l* adalah panjang urutan input saat ini, dan *d* sebagai dua kali panjang *hidden size* dari LSTM yang digunakan pada proses encoding (proses sebelumnya).

Lanjut pada proses berikut yang menerima entity queries dan representasi kata (tokens) adalah bagian kedua dari arsitektur sistem dari model ini yaitu proses *decoding* entity queries. Target dari proses ini adalah untuk melakukan membaca informasi yang telah “diringkas” dalam entity queries. Sebagian besar dari pekerjaan *decoder* dilakukan dengan mekanisme Attention yang diambil dari metode Transformer (metode ini telah dijelaskan pada bab sebelumnya mengenai bagian decoder). Decoder akan mempelajari kata-kata yang perlu diperhatikan dan ketergantungan antar entitas untuk mengetahui pola prediksi nested NER. Setelah entity queries dan tokens dilewatkan self-attention dan cross-attention, hasil dari cross-attention akan melewati bagian Feed Forward Network (FFN), bagian ini bertugas untuk lebih mengetahui hubungan antar nilai dalam embedding. Dan output dari FFN akan menjadi input pada layer terakhir bagian decoder yaitu Multilayer Perceptron (MLP) untuk mengklasifikasikan embedding FFN menjadi hasil akhir batasan kiri dan kanan prediksi jenis entitas dan juga jenis entitas yang diprediksikan.

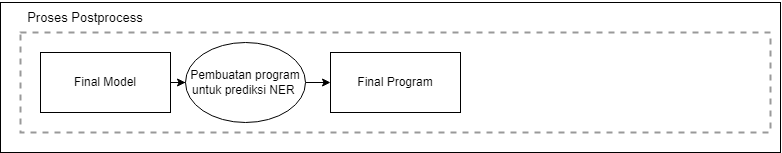
Output dari MLP ini adalah data “kumpulan prediksi entitas” dalam Gambar 3.7. Dan dengan output ini akan memasuki proses paling akhir yaitu penghitungan *loss* untuk menentukan performa dari model yang sedang training. Hal ini dilakukan dengan metode *bipartite matching*. Suatu metode yang sering digunakan untuk membantu penghitungan *assignment matching*. Dalam bagian ini, akan diusahakan untuk mencari nilai loss, dibutuhkan nilai optimal pasangan prediksi entitas dengan target entitas sebenarnya. Nilai optimal tersebut diambil dari algoritma Hungarian. Nilai loss ini akan membantu model untuk mempelajari cara prediksi nested NER.

Hasil dari proses prediksi ini adalah model yang sudah melewati proses training dan memiliki ilmu untuk prediksi nested NER dengan optimal. Setelah mendapatkan model yang optimal, model ini diperlukan untuk memprediksikan input kalimat yang ingin dicarikan entitas nya. Karena itu perlu dilewatkan tahap pasca proses yang menyiapkan model untuk menjadi program yang dapat digunakan user dengan mudah.

Proses pasca dari bagian prediksi akan dibahas dalam bagian subbab ini. Alur dari proses tersebut dapat dilihat pada Gambar 3.8 bahwa hanya terdapat satu proses yang dijalankan dengan dua hasil proses yang disediakan. Final model merupakan hasil akhir model dengan metode sequence-to-set network yang telah ditraining sedemikian rupa dengan akurasi terbaik. Model ini akan berbentuk file ekstensi .bin dan beberapa file lainnya yang akan menjadi file pendukung untuk menjalankan model tersebut. File tersebut adalah config.json, extra.state, pytorch\_model.bin, special\_tokens\_map.json, tokenizer\_config.json, vocab.txt.

Config.json adalah konfigurasi untuk model tersebut saat akan dijalankan untuk prediksi dari input program. Extra.state adalah metadata untuk model mengenai versi model dan data lainnya. Pytorch\_model.bin adalah model tersebut yang akan dijalankan dengan bantuan informasi lain seperti konfigurasinya (config.json), token spesial yang akan digunakan dalam proses analisa dataset (special\_tokens\_map.json), juga kosakata dari kata-kata yang pernah muncul didataset (vocab.txt). File tokenizer\_config.json adalah file berisi informasi mengenai tokenizer model yang digunakan untuk perubahan kata-kata dari dataset menjadi token sesuai dengan konfigurasi yang ditentukan. Contoh konfigurasi adalah apakah semua kata-kata dikonversikan menjadi *lower case*, nama dari model tokenizer yang digunakan, token-token spesial yang dapat digunakan tokenizer untuk kata/karakter spesial (kata-kata yang tidak diketahui diberikan *unknown token* biasanya dalam bentuk [UNK]/<UNK>. Seluruh file ini akan digunakan dalam proses berikutnya yaitu prembuatan program prediksi NER dengan final model.

Proses pembuatan program untuk prediksi NER adalah proses dimana file-file dari model yang telah melewati proses training, akan ikut serta dalam pembuatan program baru. Program ini akan menerima input user, kemudian program akan memberikan input ini kepada final model untuk memprediksi tag named entity dari kalimat input user tersebut. Program ini jika selesai dapat digunakan untuk menerima input beberapa kalimat user (3-4 kalimat), kemudian akan dioutput kan dalam file bentuk .ann, seperti yang pernah disebut, file pelabelan untuk BRAT untuk mempermudah visualisasi.



Gambar .  
Arsitektur Sistem Pasca Proses

Program ini akan terdiri dari beberapa file .py (file bahasa pemrogramman Python), dimana file-file tersebut juga memiliki alur sendiri. Pertama program menerima input kalimat dalam bentuk txt, kemudian kalimat tersebut akan dikonversikan menjadi struktur data yang sesuai untuk diterima model agar dapat diprediksi. Setelah dikonversikan, program akan melempar kalimat yag sudah dikonversi kedalam model yang otomatis akan memberikan sejumlah entitas prediksi yang diprediksikan. Program akan mengubah hasil prediksi ini menjadi file .ann agar dapat melihat visualisasi hasil prediksi tersebut dalam program BRAT.

## Dataset dan Tagset Bahasa Inggris

Dataset yang digunakan pada penelitian yang dirujuk di tugas akhir ini menggunakan beberapa dataset, tetapi sebagai rujukan utama di penelitian utama ini akan fokus pada dataset GENIA[[27]](#footnote-27). Sebagai informasi tambahan, dataset lain yang digunakan oleh peneliti Sequence-to-Set Network adalah ACE 2004[[28]](#footnote-28), ACE 2005[[29]](#footnote-29), KBP 2017[[30]](#footnote-30). Keempat dataset yang digunakan merupakan beberapa dataset standar yang digunakan dalam penelitian NER bahasa Inggris. Berikut penjelasan singkat untuk dataset yang digunakan dalam Sequence-to-Set Network.

Dataset GENIA adalah dataset utama untuk penulisan biomedis dalam penelitian bidang NLP. Tujuan utama adanya dataset ini untuk membantu perkembangan ekstraksi informasi dan *mining* teks dalam domain *molecular biology*. GENIA dapat digunakan dalam beberapa kategori penelitian seperti *Part-of-Speech annotation*, *Constituency* (*phrase structure*) *syntactic annotation*, *Term annotation*, *Event annotation*, *Relation annotation*, *Coreference annotation*. Bahkan GENIA adalah salah satu dataset yang pada awalnya menganotasikan dataset nya secara nested bukan flat (tidak menghiraukan nested named entities).

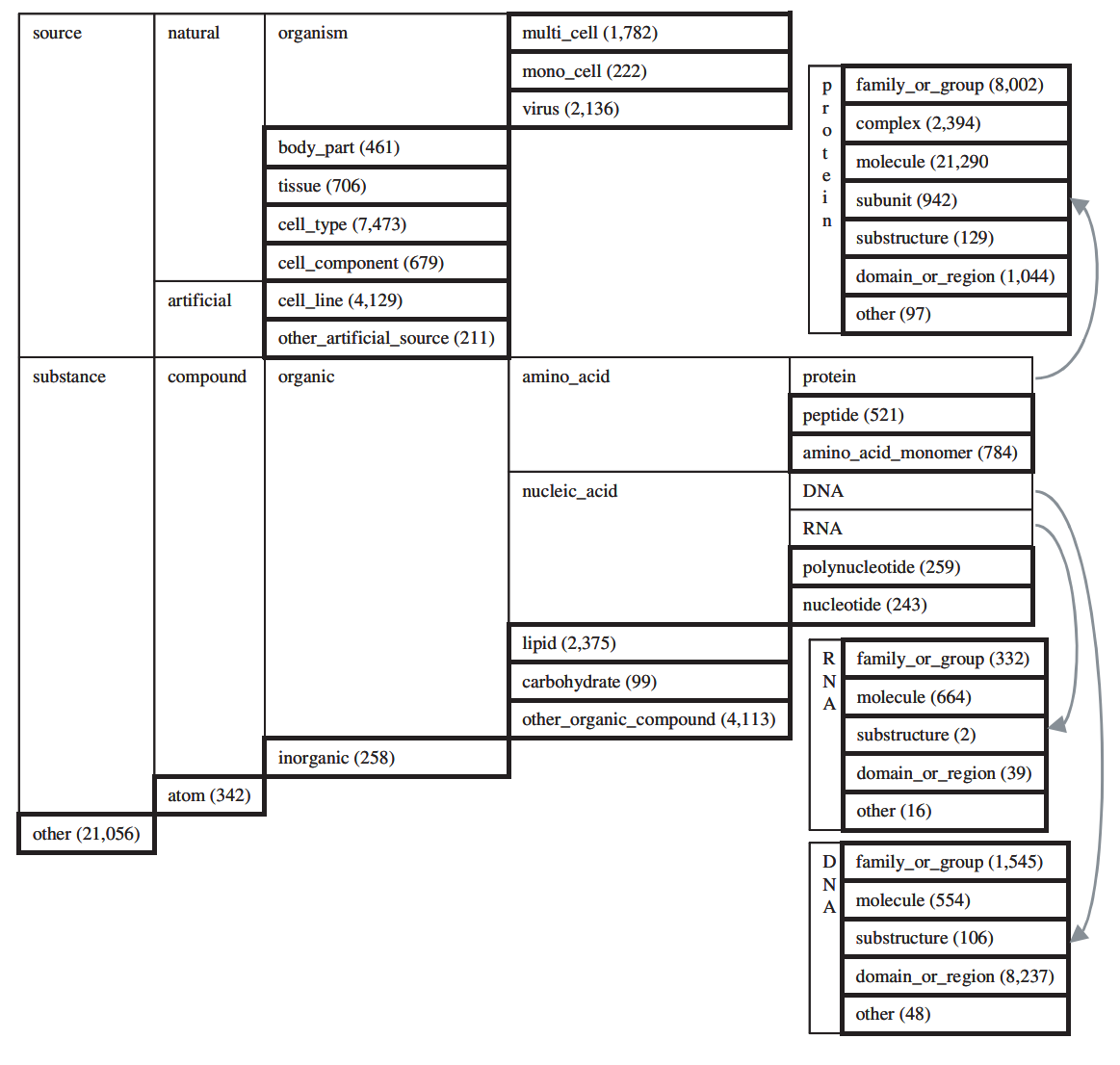
Sedangkan dataset ACE 2004 dan ACE 2005 berasal dari program riset yang sama yaitu ACE (Automatic Content Extraction) oleh institut NIST (National Institute of Standards and Technology) yang mengutamakan pengembangan ekstraksi informasi dan sudah berjalan dari tahun 1999 sampai 2008. Tidak hanya bahasa Inggris, dataset ACE berkembang tiap tahunnya dengan menyediakan untuk bahasa Mandarin, bahasa Arab, dan bahasa Spanyol. Teks yang disediakan ACE berasal dari teks berita maupun teks siaran.

Selain ACE, NIST juga menyediakan dataset lain dari konferensi mereka Text Analysis Conference 2017 dengan nama dataset Knowledge Base Population (KBP) 2017. TAC adalah konferensi yang menyediakan *evaluation workshop* untuk membantu berkembangnya penelitian di bidang NLP dengan menyediakan data dan cara evaluasi kemudian mengadakan forum untuk membahas hasil-hasil yang telah ditemukan oleh peneliti/organisasi lainnya. Ada beberapa jenis task yang disediakan oleh TAC untuk diteliti dan salah satunya adalah Entity Descovery and Linking (EDL) dalam area KBP tersebut. Task selain EDL adalah *Cold Start KB* (CSKB), *Slot Filling* (SF), *Event*, *Belief and Sentiment* (BeSt) dan area tersendiri *Adverse Drug Reaction Extraction* (ADR). Dataset yang disediakan untuk EDL digunakan sebagai salah satu dataset untuk Sequence-to-Set Network.

### GENIA

GENIA, dataset utama biomedis untuk penelitian NLP, telah berkembang dari tahun 1998 sampai saat ini, dengan fokus utama yaitu perkembangan ekstraksi informasi dan mining teks dalam domain molecular biology. Pada 2003, GENIA telah menghasilkan dataset versi ketiga yang terdiri dari 2.000 abstrak, lebih dari 400.000 jumlah kata, dan kurang lebih 100.000 anotasi dengan campur tangan manusia.

Isi dari dataset ini terdiri dari ekstraksi artikel dari MEDLINE. Bentuk dari GENIA adalah artikel yang menggunakan bentuk mark-up berbasis XML dengan ID MEDLINE, judul dan abstrak untuk setiap artikel MEDLINE. Setiap abstrak terdapat teks yang terbagi menjadi kalimat-kalimat. GENIA mengutamakan kualitas dari anotasinya, seluruh abstrak dan judulnya ditentukan oleh dua pakar domain untuk istilah biologis, dan istilah biologis tersebut telah dianotasi secara semantik dengan deskripsi dari ontologi GENIA (dapat dilihat pada Gambar 3.9).



Gambar .  
Ontologi dan Statistika dari GENIA[[31]](#footnote-31)

Tabel 3.1 adalah spesifikasi dataset untuk penelitian nested NER dari GENIA. Penelitian Sequence-to-Set Network menggunakan spesifikasi Tabel 3.1 untuk pembagian data training dan test. Hasil penghitungan ini terdapat dari sejumlah 2000 abstrak yang dimiliki. Dan penghitungan jumlah named entity dan nested named entity dari 18.546 kalimat. Tabel ini menjadi referensi untuk dataset yang digunakan dalam tugas akhir ini dengan pernyataan minimal 10-20% nested entities didapat dari jumlah entities yang dimiliki dataset tersebut.

Tabel .  
Spesifikasi Dataset GENIA

|  |  |  |
| --- | --- | --- |
| **Statistika** | **Jumlah Train** | **Jumlah Test** |
| Kalimat | 16.692 | 1.854 |
| Kalimat dengan Nested Entities | 35.22 | 446 |
| Rata-rata panjang kalimat | 25.35 | 25.99 |
| Seluruh Entity | 50.509 | 5.506 |
| Seluruh Nested Entities | 9.064 | 1199 |
| Persentase Entities (%) | 17.95 | 21.78 |

### Penjelasan Jenis Tagset Bahasa Inggris

Dataset utama yang digunakan untuk uji coba Sequence-to-Set Network dalam tugas akhir ini adalah GENIA. Namun, tagset yang dirujukkan lebih mengarah kepada tagset yang digunakan ACE dengan beberapa tambahan tagset lain (terinspirasi dari jenis tagset dari *open library* spaCy). Tabel di bawah adalah seluruh tagset yang digunakan untuk anotasi dataset ACE 2005 juga ACE 2004. Pada Tabel 3.2 terdapat kolom entitas yang berisi istilah yang digunakan untuk named entity nya. Istilah dari entitas ada dua jenis yaitu istilah panjangnya dengan istilah pendek, istilah pendek digunakan untuk memudahkan visualisasi dan anotasi saat proses pelabelan. Kemudian kolom keterangan untuk memberi penjelasan singkat mengenai entitas apa yang akan di anotasikan sebagai named entity. Kolom terakhir adalah contoh konkret subjek yang dapat dianotasikan sebagai named entity.

Tabel .  
Daftar Tagset ACE 2005

|  |  |  |
| --- | --- | --- |
| **Entitas** | **Keterangan** | **Contoh** |
| Person (PER) | Seseorang/kumpulan orang | President Obama |
| Organization (ORG) | Perusahaan, agensi, grup dalam struktur organisasi | Secret Service |
| Facility (FAC) | Bangunan, gedung, perumahan | White House |
| Location (LOC) | Lokasi geografis (darat, laut, dsb) | Red Sea |
| Geo-Political Entity (GPE) | Wilayah geografis dalam kekuasaan politik atau grup sosial | United States of America |
| Vehicle (VEC) | Alat untuk memindahkan | Limousine |
| Weapon (WEA) | Alat untuk menyakiti/menghancurkan | Sniper, Knife |

ACE 2004 dan ACE 2004 berada di bawah lisensi dan dataset ini tidak dapat diperoleh secara gratis. Harga yang dicantumkan untuk yang bukan anggota dari organisasinya akan mendapatkan tarif $ 4.000,00. Tabel merepresentasikan ketujuh tagset dari ACE, untuk contoh yang diambil secara langsung dari dataset ACE tidak dapat dilampirkan karena lisensi dari dataset tersebut. Karena itu, untuk beberapa subbab kedepan mengenai penjelasan tagset bahasa Inggris akan menggunakan contoh dan visualisasi dari penulis sendiri.

## Dataset dan Tagset Bahasa Indonesia

Pada saat ini, belum ada dataset umum untuk penelitian nested NER dalam bahasa Indonesia yang dapat digunakan sebagai referensi maupun yang dapat digunakan untuk penelitian sebagai dataset. Karena ini, dataset yang digunakan dalam tugas akhir ini diambil dari tugas akhir sebelumnya yang juga meneliti dalam topik NER, yaitu dataset dari tugas akhir mahasiswa Georgia Nikita (218116685)[[32]](#footnote-32) yang juga menggunakan topik nested NER dalam tugas akhirnya. Dataset ini juga berasal dari tugas akhir sebelum Georgia Nikita yaitu mahasiswa Christian Nathaniel Purwanto (214116299)[[33]](#footnote-33). Seperti yang diulas dalam tugas akhir Christian Nathaniel Purwanto, bahwa pemilihan tagset/jenis entitas dalam dataset ini mengambil inspirasi dari jenis tagset bahasa Inggris yang ada, namun juga tagset yang dipilih disesuaikan dengan dataset bahasa Indonesia tugas akhir ini.

Dokumen dataset nested NER tugas akhir ini bersumber dari beberapat situs berbeda. Secara singkat, sumber dataset berasal dari berita CNN Indonesia, situs ensiklopedia Wikipedia Indonesia dan crawling beberapa situs dari Google yang telah ditranslasikan ke dalam bahasa Indonesia. Secara keseluruhan, dokumen dataset ini memiliki domain berita politik sehingga entitas yang digunakan telah disesuaikan dengan seringnya beberapa jenis entitas sering disebutkan. Beberapa contoh jenis entitas yang paling sering disebut adalah person, organization, dan date atau time.

### Jenis Tagset Bahasa Indonesia

Berikut adalah penjelasan mengenai jenis-jenis entitas/tagset yang ditentukan dalam dataset tugas akhir ini. Terdapat tujuh jenis entitas dan tiap penjelasan dapat ditemukan dari Tabel 3.3. Tabel 3.3 punya kemiripan dengan tabel sebelumnya yang menjelaskan jenis tagset bahasa inggris, yaitu memiliki tiga jenis kolom. Kolom nama dengan singkatan nama jenis entitas, penjelasan mengenai jenis entitas tersebut dan contoh konkret kata-kata yang dianggap jenis entitas tersebut. Untuk inspirasi tiap dataset didapatkan dari beberapa sumber. Person, organization, location adalah tagset yang paling sering digunakan. Hampir seluruh dataset umum NER menggunakan ketiga jenis ini seperti ACE 2004, ACE 2005, CoNLL 2003, MUC 7. Untuk jenis tagset Date, Time berasal dari jenis tagset yang digunakan oleh MUC 7. Dan jenis Event merupakan salah satu tagset yang bukan berasal dari tagset yang umum, namun ditemukan dalam salah satu *library* umum yaitu spaCy. Dan untuk Miscellaneous adalah jenis entitas yang ditemukan namun berada diluar ketujuh entitas yang telah ditentukan.

Bagian penjelasan ini akan mendeskripsikan contoh kasus kata-kata yang akan dianggap merupakan jenis entitas dari jenis tagset tersebut. Penentuan kata yang termasuk entitas person akan mengarah kepada nama seseorang (Basuki Tjahaja Purnama), maupun pangkat seseorang (Irjenad Letjen TNI Rudianto). Seseorang yang memiliki panggilan seperti Bapak Ahok, Saudara Rudianto akan diketahui juga disebut entitas person. Subyek/obyek yang disebut hanya dengan jabatan seseorang dapat dianggap sebuah entitas, namun perlu adanya konteks jabatan disebut sebagai orang bukan hanya sebagai jabatan. Contohnya terdapat dua kalimat, kalimat A adalah “Itulah pekerjaan Presiden Indonesia”, dan kalimat B adalah “Akan dilakukan oleh Presiden Indonesia”. Kalimat A mereferensikan kata Presiden sebagai sebuah jabatan. Sedangkan untuk kalimat B, mereferensikan kata Presiden sebagai seseorang, sehingga untuk kalimat ini Presiden dapat ditentukan sebagai entitas PERSON. Namun untuk panggil yang tidak bernama seperti kata dia, perempuan/lelaki itu, pejabat tersebut, penghuni itu, panggilan-panggilan yang tidak memiliki konteks sebelumnya (seperti sebutan nama siapakan orang tersebut) siapa tidak akan ditandai sebagai entitas.

Tabel .  
Jenis Tagset Bahasa Indonesia

|  |  |  |
| --- | --- | --- |
| **Entitas** | **Keterangan** | **Contoh** |
| Person (PER) | Seseorang/kumpulan orang | President Obama |
| Organization (ORG) | Perusahaan, agensi, grup dalam struktur organisasi | Secret Service |
| Location (LOC) | Lokasi geografis seperti daratan, lautan, dan bentukan geologis | Red Sea |
| Date (DATE) | Tanggal, bulan, tahun | 17 Agustus 1945,  bulan Juli |
| Time (TIME) | Jam, menit, detik,  zona waktu, kurun waktu | 19:00, malam hari,  Senin pagi, era Soekarno |
| Event (EVENT) | Kejadian, peristiwa yang dirancang untuk terjadi | Pilkada 2019,  selebrasi Tahun Baru |
| Miscellaneous (MISC) | Entitas selain enam entitas sebelumnya | 10 persen, Rp. 10.000, Undang-Undang |

Definisi untuk entitas organisasi adalah sekumpulan orang yang bekerja sama untuk mencapai tujuan tertentu dan biasanya terpimpin atau terkendali. Organisasi yang mudah untuk diketahui adalah sebuah usaha atau perusahaan (Shopee, Pertamina). Adapun organisasi politik yang akan sering ditemukan dalam dataset seperti TNI, partai politik, badan legislatif. Organisasi formal yang tidak berhubungan dengan bisnis dan profesi juga dapat ditemukan seperti persekolahan, universitas, perseroan. Organisasi sosial seperti LSM, karang taruna, PMI juga diikut sertakan sebagai entitas. Untuk kumpulan orang namun tidak ada penamaan yang jelas seperti para murid, para relawan, seluruh anak, tidak dinyatakan sebagai entitas jika tidak ada konteks lebih. Contoh entitas yang dapat digolongkan sebagai organisasi dari konteks seperti para anggota DPR.

Entitas location akan digolongkan untuk suatu daerah apa pun (daratan, perairan) yang dapat ditunjuk secara akurat dengan konteks yang memadai. Contoh umum dapat diambil dari penamaan daerah seperti kota, provinsi, negara. Bisa juga perairan atau bentuk lahan yang memiliki nama yang resmi seperti Danau Toba, Laut Sulawesi, Bukit Bintang. Bisa juga lokasi suatu gedung, tempat apa pun yang memiliki penulisan nama yang jelas seperti jalan, daerah, arah mata angin. Semisal Jawa Barat, Rumah Sakit Mata Undaan (Undaan juga dapat digolongkan sebagai location sendiri), Jl. Ngagel Jaya Tengah No. 73-77. Objek wisata dan fasilitas umum seperti Trans Studio Bandung atau Bandara Soekarno-Hatta juga termasuk dalam entitas location. Beberapa dari dokumen dataset memiliki penulisan lokasi dengan penulisan bujur dan lintang, 10.59 Lintang Selatan (LS) dan 119.42 Bujur Timur (BT) juga termasuk location. Entitas location yang ada dari dataset adalah daerah jalan yang ditentukan arah dan tujuan (KM 14 Tol Jagorawi arah menuju Cawang).

Selanjutnya entitas date cukup singkat dan jelas, yaitu penulisan tanggal baik tanggal hari, tanggal bulan, tanggal tahun. Juga penulisan nama bulan dapat dihitung sebagai entitas date. Contoh yang dapat diberikan, bulan Januari, 17/08/1945, tahun 2022, dan kombinasi penulisan tanggal lainnya. Entitas date sering dipasangkan dengan entitas time, dimana entitas time akan memiliki ambiguitas yang banyak. Ambiguitas ini sering ditemukan karena entitas time didefinisikan dengan periode waktu yang disebut. Beberapa contoh yang jelas sebagai time adalah nama hari, waktu, zona waktu, jumlah waktu, bagian hari. Juga ada contoh seperti periode, masa, dan lainnya. Semisal, hari Senin, 17:00 WIB, 1 jam 30 menit, enam hari, malam hari, tahun ajaran 2022 – 2023, era Soeharto.

Event dalam bahasa Indonesia adalah peristiwa, kejadian. Dalam tugas akhir ini, ditentukan adanya entitas ini untuk menyesuaikan dengan kebutuhan dataset yang menyebut entitas event cukup sering. Untuk mempersingkat, entitas ini dianggap jika kata-kata yang disebut menyebut suatu nama kejadian/peristiwa yang yang sudah dirancang secara sengaja untuk terjadi. Semisal untuk kejadian pemilihan kepala daerah adalah suatu entitas event yang telah dirancang dan memiliki jadwal yang pasti tiap beberapa tahun. Bisa juga kejadian yang formal yang bukan diadakan beberapa hari seperti Pilkada, rapat paripurna atau suatu persidangan bisa juga acara yang diadakan oleh seorang organisasi seperti acara festival atau lomba.

Miscellaneous jika ditranslasi dalam bahasa Indonesia berarti aneka ragam atau bermacam-macam. Tujuan dengan adanya jenis entitas ini untuk menandakan kata-kata yang bisa termasuk named entity, namun tidak temrasuk dalam jenis yang telah di tentukan dalam named entity yang sudah ada (dari keenam jenis named entity lainnya). Beberapa contoh kata-kata yang ditemukan dalam dataset untuk named entity miscellaneous adalah entitas jenis uang (Rp. 100.000, 10 ribu rupiah, USD 100), entitas jenis dokumen/perhukuman (Pasal 107 KUHP, Undang-Undang, Peraturan Pemerintahan/PP), entitas jenis karya seperti buku novel, lukisan, lagu dan lain-lainnya (seni batik, Indonesia Raya, Ada Apa dengan Cinta), entitas jenis produk (Toyota, Suzuki APV).

### Statistika Dataset

Dataset bahasa Indonesia pada tugas akhir ini berasal dari CNN Indonesia dengan domain berita politik. Dataset ini sebelumnya sudah pernah dipakai untuk tugas akhir Christian Nathaniel Purwanto (214116299) dan juga tugas akhir Georgia Nikita (218116685). Berjumlah 2000 dokumen, isi dari tiap dokumen merupakan berita politik Indonesia. Mayoritas berita akan menyinggung politik seperti kasus korupsi, kenaikan jabatan menteri, pelaksanaan atau kampanye Pilkada, berita kemacetan lalu lintas. Tabel 3.4 adalah tabel yang menunjukkan statistika named entity dataset pada saat tugas akhir ini dikerjakan. Statistika yang ditunjukkan adalah jumlah entitas yang bersarang dalam dataset secara keseluruhan dalam satuan persentase dengan jenis entitas/tagset yang telah dibahas pada bab sebelumnya. Detail secara keseluruhan akan dibahas di akhir bab ini.

Tabel .   
Spesifikasi Dataset NER Bahasa Indonesia

|  |  |  |  |
| --- | --- | --- | --- |
| **Statistika** | **Jumlah Entitas** | **Jumlah Entitas Bersarang** | **Persentase Jumlah** |
| Person (PER) | 34.720 | 8.488 | 24% |
| Organization (ORG) | 20.857 | 8.627 | 41% |
| Location (LOC) | 20.004 | 8.578 | 43% |
| Date (DATE) | 5.720 | 3.548 | 62% |
| Time (TIME) | 7.679 | 1.292 | 17% |
| Event (EVENT) | 3.103 | 609 | 20% |
| Miscellaneous (MISC) | 6.774 | 1.345 | 20% |

## Preprocessing

Pra proses dalam tugas akhir ini sempat disinggung dalam bab arsitektur sistem. Proses ini dilakukan kepada dataset tugas akhir sebelum menjadi input untuk training model nya. Pada bab arsitektur sistem, pembahasan yang diberikan lebih difokuskan kepada alur sistem dan data yang digunakan sebagai input dan output. Pada subbab ini akan dibahas lebih mendetail tiap tahap pra proses yang dilakukan pada dataset tugas akhir. Terdapat tiga sub bab yaitu cara pelabelan dataset, konversi struktur dataset dan struktur akhir dataset.

### Struktur dan Pelabelan Dataset

Dataset tugas akhir ini terbantu dengan bantuan tugas akhir yang juga mengambil topik NLP. Beberapa tugas akhir pembantu telah disebut sebelumnya dan ada yang belum disebut. Tiap proses yang dilewatkan dataset dari tiap tugas akhir akan dibahas satu per satu. Alur besar dari penjelasan adalah sumber utama isi dataset dari website yang diambil dan juga bantuan *crawler* website. Juga format dataset yang sudah pernah dibuat, dan bentuk-bentuk akhir dataset yang bisa menjadi file input output kepada program masing-masing. Adapun library yang membantu pra proses dataset ini akan dijelaskan. Penjelasan dan sumber dataset didapatkan dari Amelinda Tjandra Dewi (214116288)[[34]](#footnote-34), Christian Nathaniel Purwanto (214116299), Georgia Nikita (218116685).

Topik/domain dari dataset adalah berita politik yang berhubungan dengan politik Indonesia, ada pun beberapa berita lainnya seperti berita lalu lintas namun mayoritas dari dataset berisi domain politik. Hal ini karena sumber berita diambil dari dua website berbeda yang menyajikan berita Indonesia yaitu CNN Indonesia[[35]](#footnote-35) dan Liputan 6[[36]](#footnote-36). CNN Indonesia dan Liputan 6 tidak hanya memiliki website portal berita tetapi juga siaran televisi yang sudah mulai dari tahun 2015 dan Liputan 6 pada 2000. Kedua situs berita menyediakan berita dalam bahasa Indonesia dengan tema yang sebagian besar sejenis. Berita yang disediakan seperti berita umum, nasional, lokal, bisnis, teknologi dan hiburan, dan untuk tiap portal memiliki fokus berbeda. Untuk CNN Indonesia memiliki fokus terhadap berita nasional, internasional, politik, olahraga dan Liputan 6 memiliki beberapa fokus berita yang khusus seperti meskipun ada jenis berita bisnis, terdapat juga bagian yang fokus terhadap berita *cryptocurrency*, saham, adapun bagian berita untuk foto.

Total berita yang diambil adalah 2000 berita dari kedua website dan pembagian dibagi sesuai dengan kebutuhan masing-masing tiap tugas akhir (contohnya pada tugas akhir Amelinda pembagian training dan test bergantung pada kasus uji coba). Untuk tugas akhir ini pembagian dataset akan dibagi dalam persentase 90 untuk training dan development dan presentase 10 untuk testing. Spesifikasi detail mengenai pembagian dataset akan dibahas pada subbab berikut ini.

Pengambilan dataset dari tiap website dilakukan dalam tugas akhir Amelinda menggunakan library crawler bahasa pemrogramman Python bernama Beautiful Soup. Secara singkat, cara library tersebut mengambil data dengan library ini adalah melihat struktur HTML dari halaman web yang diminta. Alur program crawler yang dibuat adalah mengambil link berita dari website dan dikumpulkan. Kemudian link tersebut akan diakses dan konten dari halaman tersebut akan dicrawl dari elemen-elemennya. Contoh pengambilan judul dari elemen H1, isi dari berita adalah elemen div dengan class bernama article-content-body\_item-content. Hasil *crawling* ini dimasukkan ke dalam suatu database untuk memudahkan pengumpulan data dan proses berikutnya untuk pra proses.

Input untuk tugas akhir Amelinda sama seperti dengan tugas akhir ini yaitu berupa token yang *word-level*. Karena itu hasil crawling akan berupa file ekstensi txt yang akan dilewatkan tokenisasi dan menghasilkan file ekstensi txt yang berbeda. Contoh kalimat untuk file pertama akan berisi “oleh pihak tergugat, Agus.”. Sedangkan contoh hasil kalimat setelah tokenisasi pada file lainnya adalah “oleh pihak tergugat , Agus . ”. File kedua (yang telah ditokenisasi) adalah file yang akan digunakan pada seluruh tugas akhir termasuk tugas akhir ini. Dan file berikutnya yang juga digunakan semua tugas akhir ada file ekstensi ann. File ini telah dijelaskan isi, struktur dan proses pembuatan file pada subbab 2.1.1 sistem arsitektur pra bagian proses. Dan file tersebut akan diubah menuju struktur data yang diperlukan model tiap tugas akhir.

Namun untuk struktur input data training model dari ketiga tugas akhir tersebut (selain tugas akhir ini), format yang dibutuhkan adalah format BIO. BIO adalah format tag dengan keterangan tag B untuk Beginning, I untuk Inside, O untuk Outside. B menyatakan bahwa token tersebut adalah awalan untuk sebuah entity, I untuk menyatakan token tersebut berada didalam suatu entity yang sudah diawali sebelumnya dengan tag B. O adalah untuk token yang tidak termasuk entity apapun. Dan seluruh tugas akhir tersebut (kecuali Georgia Nikita dan tugas akhir ini), juga akan memiliki POS tag selain BIO tag untuk setiap kata. Isi dari struktur data tugas akhir Christian dan Amelinda dapat dilihat sebagai berikut:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | Menteri | JJ | B-JOB | O | -1 |
| 1 | Luar | JJ | I -JOB | O | -1 |
| 2 | Negeri | NN | I -JOB | O | -1 |
| 3 | Iran | NNP | B-PER | Corefer-to | 0 |
| 4 | Javad | NNP | I-PER | O | -1 |

Berikut adalah isi dari file dengan ekstensi .bio untuk input training model. Kolom pertama menyatakan kode untuk tiap kata, kolom kedua adalah isi dari kata yang dilabelkan, kolom kedua sebagai isi dari POS tag, berikutnya adalah tag named entity dalam format IOB, dan dua kolom terakhir digunakan untuk mengetahui informasi *coreference* yang ada. Apabila kata tersebut memiliki nilai coreference terhadap kata lain. Kolom kedua terakhir adalah jenis relasi dan kolom terakhir ada kode kata yang kata tersebut memiliki coreference. Coreference dapat ditemukan dalam dataset ini karena tujuan tugas akhir tersebut meneliti kombinasi NER dan coreference.

Untuk tugas akhir ini, memiliki struktur akhir dataset (data input) mengikuti struktur yang telah diberikan dari penelitian metode Sequence-to-Set Network. File input akan berupan ekstensi .json, satu file adalah sekumpulan JSON objek dengan atribut yang sudah ditentukan. Sebelum menjelaskan isi dari dataset perlu diketahui bahwa dokumen yang disebut adalah satu teks berita dengan berbagai kalimat. File input untuk tugas akhir ini hanya tunggal karena seluruh dokumen akan digabung menjadi satu. Struktur secara keseluruhan untuk satu JSON objek dapat dilihat pada Gambar 3.10.



Gambar .  
Contoh Struktur Dataset

Terdapat tujuh atribut yang diperlukan untuk input training model metode ini. Setiap JSON objek dari dalam dataset harus memiliki ketujuh atribut, meskipun beberapa atribut tidak akan digunakan sebagai parameter training dalam penelitian tugas akhir ini. Nama-nama atribut secara urut adalah tokens, entities, relations, org\_id, pos, ltokens, rtokens. Tokens adalah kalimat utama dari JSON objek tersebut. Isi dari tokens diambil dari dokumen yang dipecahkan berdasarkan kalimatnya, kemudian kalimat-kalimat tersebut akan dipecah lagi untuk tiap kata. Karena itu isi dari tokens adalah *array of word-level* *token* (word-level artinya dipecahkan berdasarkan kata-kata). Entities cukup jelas yaitu array dari named entity untuk kalimat itu saja. Penyimpanan named entity nya menggunakan JSON objek juga dengan tiga atribut start, end dan type. Start dan end adalah indeks, namun berbeda dengan BRAT, indeks ini akan mengikuti bentuk tokens yaitu menjadi indeks untuk kata keberapa (word-level bukan *character-level*). Dan type adalah jenis named entity untuk kata tersebut.

Kedua atribut berikutnya, relations dan org\_id, tidak akan digunakan sebagai input ke dalam training model, karena untuk relations digunakan untuk coreference yang bisa dijadikan tambahan informasi untuk model namun tidak digunakan dalam penelitian tugas akhir ini. Untuk org\_id akan berisi kode untuk tiap asal dokumen untuk kalimat tersebut, sebagai bantuan dokumentasi saja. Ltokens dan rtokens dapat dijelaskan bersamaan karena memiliki kegunaan yang sama. Isi dari kedua atribut tersebut sama dengan atribut tokens, tetapi ltokens akan mengambil kalimat sebelum kalimat utama dan rtokens akan mengambil kalimat setelah kalimat utama. Huruf l dalam ltokens adalah *left*/kiri, huruf r dalam rtokens adalah *right*/kanan, mereferensikan kanan dan kiri kalimat utama.

### Konversi dan Statistika Dataset

Dari subbab struktur dan pelabelan dataset, diketahui bahwa tidak semua metode menerima struktur data yang sama, karena itu dari struktur data alat pelabelan menuju model perlu konversi. Tugas akhir ini telah menyediakan konversi dataset dari file BRAT menjadi file JSON dataset. Beberapa bagian penting untuk konversi dataset akan dijelaskan dalam subbab ini, perlu diketahui bahasa pemrogramman untuk konversi dataset menggunakan Python. Yang akan dijelaskan adalah segmen program untuk preprocessing file txt berisi dokumen teks berita yang sudah ditokenisasi, segmen program preprocessing file ann berisi tag named entity untuk tiap dokumen, juga preprocssing untuk penyesuaian indeks BRAT dengan indeks word-level.

Segmen Program . Preprocessing TXT File

1. # Preprocessing TXT File
2. batas\_files = 2000
3. ctr\_files = 0
4. for filename in filenames :
5. ctr\_files = ctr\_files + 1
6. if (ctr\_files <= batas\_files) :
7. with open(str(filename)+".txt", 'r') as f:
8. t\_doc = []
9. temp = f.read().split(' \n')
10. first\_sen= True
11. last = 0
12. for idx, t in enumerate(temp) :
13. t\_first\_idx = -1
14. t\_tokens = t.split(' ')
15. if first\_sen == True :
16. t\_first\_idx = 0
17. first\_sen = False
18. else :
19. t\_first\_idx = last + 2
20. t\_last\_idx = len(t) + t\_first\_idx
21. last = t\_last\_idx
22. t\_ltokens = []
23. t\_rtokens = []
24. if idx-1 >= 0 :
25. t\_ltokens = temp[idx-1].split(' ')
26. if idx+1 <= len(temp)-1 :
27. t\_rtokens = temp[idx+1].split(' ')
28. t\_pos = ['<UNK>' for i in t\_tokens]
29. t\_obj = {
30. "tokens" : t\_tokens,
31. "first\_idx" : t\_first\_idx,
32. "last\_idx" : t\_last\_idx,
33. "entities": [],
34. "pos": t\_pos,
35. "ltokens": t\_ltokens,
36. "rtokens": t\_rtokens
37. }
38. t\_doc.append(t\_obj)
39. documents.append({
40. "id\_doc" : str(filename),
41. "t\_doc" : t\_doc
42. })
43. len(documents)

Input Segmen Program 3.1 adalah txt file dokumen berita yang telah ditokenisasi tiap kata dan juga tanda baca (petik, koma, titik, garis miring). Output dari segmen program ini adalah variabel *documents* yang merupakan kumpulan objek oleh tiap dokumen dengan atribut nama dan detail dokumen. Baris 13 – 22 adalah perintah program yang berperan untuk menghitung indeks pertama dan indeks terakhir untuk tiap baris di dokumen tersebut. Contohnya untuk kalimat pertama memiliki indeks awal dan akhir yaitu 0 dan 23, sedangkan mungkin untuk kalimat kedua memiliki indeks awal dan akhir yaitu 25 dan 50. Perlu diperhatikan dari kalimat pertama dan kedua terdapat jarak indeks sebanyak dua, hal ini karena indeks dari BRAT juga menghitung karakter spesial yaitu ‘\n’. Dengan ini indeks tiap awal akan ditambah sebanyak dua agar lebih tepat. Indeks awal dan akhir tiap kalimat akan dijelaskan perannya pada segmen program pra proses ANN file.

Untuk baris 23 – 28 akan melakukan pra proses untuk mendapatkan nilai ltokens dan rtokens. Baris selanjutnya mengisi variabel untuk POS tag, namun karena tugas akhir ini tidak menggunakan POS tag maka variabel ini akan diisi dengan tokens <UNK> diketahui sebagai token untuk kata yang tidak dikenal. Namun ini tidak akan mempengaruhi penelitian karena parameter untuk mengaktifkan penggunaan POS embedding dimatikan (nilai parameter diatur menjadi *false*). Dan baris 30 – 38 adalah deklarasi variabel objek berisi detail yang dibutuhkan untuk struktur dataset akhir nanti.

Segmen Program . Preprocessing ANN File

1. for idx, t in enumerate(temp) :
2. t\_splitted\_tab = t.split('\t')
3. if len(t\_splitted\_tab) > 1 :
4. is\_error = False
5. try:
6. t\_splitted\_space = t\_splitted\_tab[1].split(' ')
7. t\_splitted\_space[1] = int(t\_splitted\_space[1])
8. t\_splitted\_space[2] = int(t\_splitted\_space[2])
9. except:
10. is\_error = True
11. print('FILENAME ERROR : ', filename,
12. 'KODE ERROR : ', t\_splitted\_tab[0],
13. 'kalimat : ', t\_splitted\_space)

Segmen Program 3.2 dijalankan setelah mendapat isi dari dokumen, kemudian label data yang telah dilakukan akan dilewatkan pra proses juga agar dapat dimasukkan ke dalam variabel program. Perintah yang akan dipakai adalah *split* karena penulisan format dalam file ann sangat templat. Bentuk dan format telah dijelaskan di bab ketiga mengenai arsitektur sistem bagian pra proses. Secara singkat, baris 2 akan menerima satu baris dari file ann kemudian dipisahkan berdasarkan karakter spesial tab (\t), hasil dari pemisahan tersebut adalah mendapatkan kode label, indeks label dan jenisnya, yang terakhir adalah kata-kata yang dilabelkan. Baris 6 – 14 berusaha melakukan perintah split karena indeks label dan jenisnya bergabung menjadi satu dengan pemisahan spasi (‘ ‘). Apabila terjadi kegagalan artinya ada penulisan di file tersebut yang tidak sesuai format BRAT, sehingga file tersebut tidak akan dimasukkan ke dalam dataset dan akan dicetak nama file untuk mempermudah preprocess secara manual.

Segmen Program . Preprocess Indeks ANN

1. def convert\_index(entities\_coba, tokens):
2. entities\_converted = []
3. str\_coba = " ".join(tokens)
5. for idx\_e, e in enumerate(entities\_coba) :
6. start\_idx = e['start']
7. end\_idx = e['end']
8. ctr = -1
9. sum = 0
10. start\_entity = -1
11. end\_entity = -1
12. for kata in str\_coba.split(" "):
13. ctr = ctr + 1
14. sum = sum + len(kata) + 1
16. if sum > start\_idx and start\_entity == -1:
17. start\_entity = ctr
18. if sum > end\_idx and end\_entity == -1:
19. end\_entity = ctr
20. temp = {
21. 'start' : start\_entity,
22. 'end' : end\_entity+1,
23. 'type' : e['tag']
24. }
25. entities\_converted.append(temp)
26. # print(temp)
27. break
29. return entities\_converted

Segmen program terakhir nomor Segmen Program 3.3, digunakan untuk mengubah indeks awal dan akhir yang sesuai alat pelabelan yaitu BRAT menjadi indeks yang esuai dengan struktur data akhir untuk training model. Indeks yang digunakan BRAT menggunakan format indeks character-level (indeks melihat huruf keberapa) dan indeks dimulai dari angka nol untuk satu dokumen, bukan untuk tiap kalimat. Untuk Sequence-to-Set Model menggunajan format indeks word-level (indeks melihat kata keberapa) dimana tiap kalimat mulai dari 0. Cara program berjalan pada inti program yaitu baris 14 – 25. Dari indeks yang telah diberikan, tiap kalimat yang telah dibuat dari tokens yang dimiliki akan dihitung panjangnya. Penghitugan panjang dari kalimat akan dilakukan per kata dan apabila panjang tersebut lebih dikit daripada indeks label saat itu maka akan dicatat sebagai indeks pertama atau terakhir. Jika kedua indeks ditemukan maka langsung akan dibuatkan objek baru dengan konversi tersebut.

Tabel .  
Statistika Dataset Tugas Akhir

|  |  |
| --- | --- |
| **Statistika** | **Jumlah** |
| Kalimat | 32.355 |
| Kalimat dengan entitas bersarang | 12.147 |
| Rata-rata panjang kalimat | 19,04132 |
| Seluruh entitas | 98.857 |
| Seluruh entitas bersarang | 32.487 |
| Persentase entitas (%) | 32% |

Berikut adalah statistika dari dataset yang telah melewatkan pelabelan terbaru. Tabel 3.5 menampilkan beberapa detail dataset yang diambil dari tabel spesifikasi dataset oleh penelitian tugas akhir ini. Seluruh kalimat (kalimat yang memiliki atau tidak memiliki pelabelan terhitung semua) dari dataset telah dihitung dengan jumlah sebanyak 32.355, dan untuk kalimat yang memiliki entitas bersarang berjumlah 12.341 sebanyak 38% kalimat yang mengandung entitas bersarang. Sebagai informasi tambahan, panjang kalimat secara keseluruhan berkisaran 19. Dan keterangan mengenai entitas yang berada didalam dataset terdapat 98.857 entitas (termasuk entitas bersarang). Dan didalam jumlah seluruh entitas, yang mengandung entitas bernama sebesar 20.745, ini berarti 21% dari seluruh entitas memiliki entitas bersarang.

BAB IV

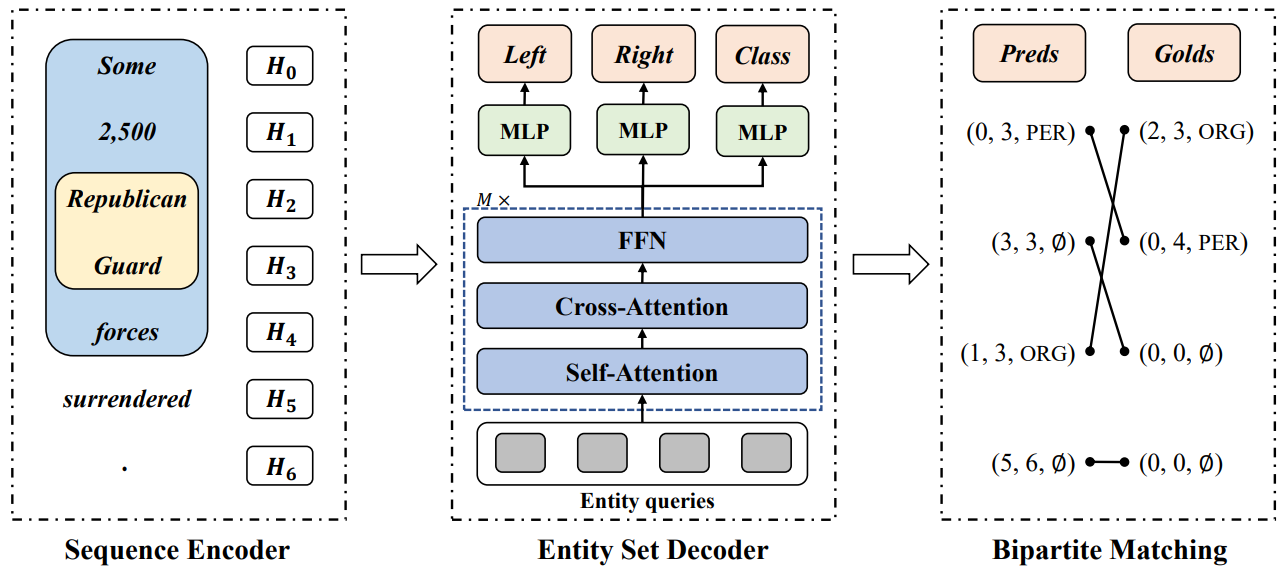
# SEQUENCE TO SET DALAM BAHASA INDONESIA

Sebelum memasuki uji coba tugas akhir ini, perlu diketahui seluruh isi dari arsitektur Sequence-To-Set Network secara detail. Sequence-To-Set Network memiliki tiga bagian besar dari arsitekturnya yaitu *encoder layer*, *decoder layer* dan *bipartite matching.* Bab ini akan dibagi menjadi tiga subbab juga, menjelaskan masing-masing tiga bagian besar itu, juga tiap subbab akan dibagi menjadi subbab kecil lagi sesuai dengan komponen/metode/fungsi yang digunakan untuk mencapai struktur akhir dari tiap subbab. Sebagai contoh, untuk encoder layer memiliki lima subbab kecil karena penggunaan empat jenis *word embedding* di mana tiap embedding memiliki cara tersendiri dan diperlukan penjelasan untuk tiap cara embedding tersebut. Namun, sebelum memasuki penjelasan tiap layer akan dijelaskan secara keseluruhan alur dari arsitektur sistem Sequence-To-Set Network.

## Sequence To Set Network

Sesuai pada gambar 4.1, alur dari Sequence-To-Set Network terdiri dari sequence encoder layer, entity set decoder layer dan bipartite matching dalam urutan yang telah disebut. Alur dari sistem ini adalah pertama kalimat input diterima oleh sequence encoder layer dalam bentuk struktur data yang telah dibahas sebelumnya mengenai pra proses dataset. Di mana output dari encoder ini memiliki tujuan untuk mendapatkan representasi dari kalimat tersebut untuk tiap kata dari kalimat tersebut dengan informasi dan konteks yang didapatkan.

Cara encoder mendapatkan informasi ini adalah dari beberapa metode word embedding yang telah ditentukan. Setiap dari embedding tersebut, seperti yang telah dijelaskan, akan dikenal lebih dalam pada subbab-subbab nya sendiri. Terdapat lima jenis word embedding yang disediakan, namun pada tugas akhir ini hanya digunakan empat (BERT, Word2Vec, Character-level LSTM, BiLSTM, POS Tag). Seluruh hasil embedding dari tiap metode akan kemudian digabung menjadi satu variabel dalam bentuk *vector*. Representasi kata terakhir ini Bersama dengan sebuah set vektor yang dapat dipelajari (dipanggil sebagai entity queries) dan menjadi output terakhir dari sequence encoder, juga menjadi input untuk bagian berikut yaitu set decoder layer.



Gambar .  
Arsitektur Sistem Sequence-To-Set Network[[37]](#footnote-37)

Entity set decoder layer, mengambil entity queries dan representasi token dari sequence encoder yang telah memberikan nilai-nilai konteks dan informasi yang penting untuk tiap kata agar decoder mengerti kata apa yang perlu diperhatikan untuk diprediksikan. Cara untuk memperhatikan dan melakukan proses decoding ini adalah dengan bantuan attention. Pada bab teori penunjang telah menjelaskan sekilas mengenai self-attention dan cross-attention masing-masing. Untuk subbab decoder di bab ini akan menjelaskan lebih rinci terhadap alur, input, output tiap layer mengingat bahwa beberapa dari layer decoder memiliki cara kerja dan peran berbeda.

Dan bagian terakhir ada bagian yang menghitung nilai error dari keseluruhan training yang telah dilakukan encoder dan decoder Sequence-To-Set Network ini. Cara untuk menghitung nilai error dari perbandingan set cukup menyusahkan, karena itu Sequence-To-Set Network memilih bipartite matching sebagai metode penghitungan loss function untuk menghitung pencocokan optimal dari prediksi dan *golden entity*. Nilai optimal dari pencocokan set prediksi dengan golden entity didapatkan dengan bantuan dari Algoritma Hungarian.

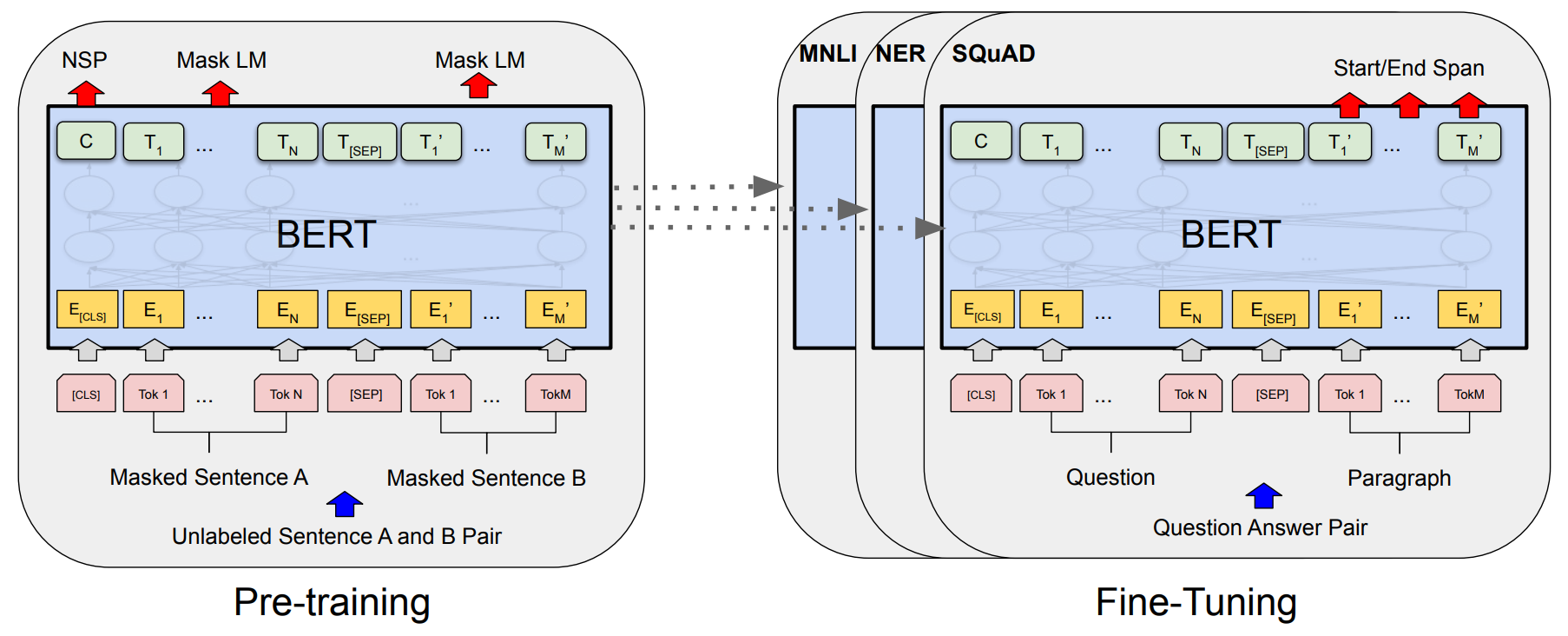
### Sequence Encoder

Sequence encoder adalah proses/bagian/layer pertama yang berada di Sequence-To-Set Network. Sama dengan penjelasan sebelumnya, tujuan dari proses ini adalah untuk mendapatkan informasi dan konteks kata-kata dari kalimat input yang didapatkan. Seluruh subbab dari bagian ini adalah semua word embedding yang digunakan pada tugas akhir ini. Penjelasan untuk word embedding yang tidak digunakan (POS tag) tidak akan dijelaskan dalam bab ini. Pada akhir subbab ini akan dijelaskan representasi kata, bentuk akhir yang dihasilkan dari proses encoder nanti.

#### BERT

Bidirectional Encoder Representations from Transformers (BERT) adalah metode yang telah diteliti dan *publish* oleh Google AI Language[[38]](#footnote-38). Metode BERT ini memiliki tujuan yaitu untuk menunjukkan pentingnya *bidirectional* pre-training dalam sebuah language model, karena pada saat itu metode Long Short Term Memory (LSTM) tidak dapat dinyatakan sebagai bidirectional (meskipun untuk BiLSTM juga tidak disebut mengambil informasi secara bidirectional. Karena ini, BERT dibuat untuk mendemonstrasikan hal tersebut. Secara struktur, BERT mengambil semua bagian dari encoder Transformers tanpa decodernya sekali pun. Memang Transformers mempunyai guna pada task yang paling cocok yaitu translasi bahasa. Namun untuk BERT, atau tumpukkan dari layer encoder dari Transformers, dapat memberikan solusi untuk beberapa pekerjaan seperti translasi, tanya jawaban, juga bisa analisa sentimen. Hal-hal ini hanya dapat dilakukan dengan BERT memahami informasi mengenai cara bahasa tersebut bekerja.

BERT akan dibagi menjadi dua tahap training, pretrain BERT untuk mengenal bahasa tersebut terlebih dahulu. Kemudian fine tune BERT untuk mengarahkan pemikiran solusi BERT tersebut kepada tugas yang ditujukan. Penjelasan berikut adalah mengenai proses yang dilewatkan untuk BERT.



Gambar .  
Visualisasi Arsitektur BERT QnA

Bagian pertama adalah pre-training (gambar 4.2 kotak kiri), pada gambar akan dijelaskan setiap arti dari kotak yang ada terlebih dahulu. Gambar alur dari pre-training berawal dari bawah menuju ke atas, dengan *unlabeled sentence A and B pair* adalah input dari kalimat A dan B yang kemudian akan diberikan token mask (hal ini akan dibahas dipararaf berikutnya), menghasilkan *Masked Sentence A* dan *Mask Sentence B*. Kalimat tersebut akan dibagi menjadi token yang word-level (kotak merah) yang selanjutnya akan dibuatkan representasi embedding (kotak kuning) untuk tiap token kata tersebut. Embedding akan dimasukkan ke dalam arsitektur BERT dan dari BERT menghasilkan output (kotak hijau). Output akan terdiri dari vektor yang disebut *word vector* dan juga satu output biner (kotak hijau paling kiri label C) untuk mengindikasi apakah kalimat B berhubungan dengan kalimat A. Semua penjelasan mengenai tiap bagian akan dijelaskan pada paragraf berikut ini.

Bagian pertama (pre-training) ini mengajar model untuk mengenal bahasa apa yang dipelajari, apa itu bahasa dan apakah bentuk konteks dalam bahasa tersebut. Cara untuk mencapai pengertian tentang bahasa dan konteks dengan baik adalah dengan dua cara, Masked Model Language (MLM) dan Next Sentence Prediction (NSP). MLM diaplikasikan pada alur pada input kalimat (Masked Sentence A dan Masked Sentence A) dan NSP pada output nanti (output pada kotak pertama label C).

MLM adalah cara komputer belajar bentuk dari kalimat sebuah bahasa menggunakan token *mask* yang akan digantikan beberapa kata dari suatu kalimat. Dari penutupan kata ini dan juga menebak kata apa yang sesuai, BERT dan memahami apa yang wajar untuk mengisi dengan cara melihat konteks juga dalam satu kalimat. Contoh yang dapat dimengerti untuk metode MLM adalah jika sebuah kalimat awalnya “The [MASK1] brown fox [MASK1] over the lazy dog. Kemudian cara satunya adalah NSP yang dapat dimengerti dari namanya sendiri, yaitu cara belajar komputer untuk mengerti kalimat yang dapat dihubungkan dengan kalimat lain. Hal ini juga dapat mengajar BERT untuk melihat konteks tiap kalimat untuk menyatukan mereka yang berhubungan. Contohnya, kalimat pertama adalah “Kampus ISTTS sangat ramai”. Kalimat keduanya, “Gedungnya sangat tinggi”. Dan kalau komputer yang sedang memelajari, artinya jawaban dari NSP yang benar seharusnya “benar kalimat pertama berhubungan dengan kalimat kedua”.

Setelah mengenal bentuk dari bahasa yang akan dipelajari, BERT perlu di fine tune agar sesuai dengan tujuan dari pembuatan model tersebut. Contoh dari penjelasan ini adalah untuk membuat model yang dapat melakukan tanya jawab dengan baik. Karena tujuan dari model harus menjawab dari sebuah pertanyaan, maka harus diberikan *supervised* dataset yang berisikan pertanyaan dan juga jawaban benar terhadap pertanyaan tersebut. Keunggulan BERT muncul di bagian ini. BERT yang telah pretrained telah mengenal bahasa tersebut, sehingga yang perlu diubah adalah beberapa parameter di dalam BERT tersebut, dan juga jenis output dari BERT harus diubah sesuai dengan yang diinginkan (yaitu output kata-kata). Dengan ini membuktikan BERT adalah metode yang dapat digunakan untuk memahami bahasa dan dengan cepat di fine tune untuk mendapatkan model yang bertujuan untuk suatu task dengan fine tune yang hanya mengubah beberapa dari isi BERT. Cara fine-tune dari arsitektur pre-training adalah input diubah, kalimat pertama adalah pertanyaan dan kalimat kedua adalah jawaban (gambar 4.2 kotak kanan/fine-tune, bagian Question dan Answer). Output juga diubah dari awalnya output untuk menebak kata yang di berikan mask, menjadi prediksi yang menjawab pertanyaan dalam bentuk jangkauan kata-kata dari kalimat jawaban (start/end span).

Yang belum dibahas pada bagian ini adalah apakah bentuk embedding dari input kalimat menuju BERT? Embedding terakhir untuk input direpresentasikan dengan gabungan antara tiga jenis embeddings yaitu token embeddings, segment embeddings, position embeddings. Token embeddings adalah token yang sudah dilatih sebelumnya (pretrained) dari metode yang ditentukan paper yaitu (WordPiece[[39]](#footnote-39)). Segment embedding adalah penanda untuk tiap token, bahwa token itu miliki kalimat apa (kalimat pertama atau kedua/ kalimat pertanyaan atau kalimat jawaban). Dan position embeddings diperlukan untuk menandakan token tersebut kalimat ke berapa, mengingat bahwa seluruh proses training akan berlangsung secara bersamaan.

Hal belum dibahas juga adalah konversi word vector dan juga penghitungan loss dari prediksi word vector tersebut. Cara untuk mendapatkan hasil output dalam kata-kata adalah word vector akan dilewatkan kepada sebuah softmax layer dengan 30.000 neuron (jumlah dari neuron bergantung pada jumlah kata dalam kosakata yang digunakan, dalam kasus ini WordPiece memiliki 30.00 kata). Softmax akan memberikan hasil akhir kata yang diprediksikan, kata-kata ini akan dibandingkan dengan kata sebenarnya yang akan direpresentasikan dalam bentuk *one hot encoded* dengan panjang 30.000 (sesuai jumlah kosakata).

Tugas akhir ini tidak akan menggunakan BERT namun BERT versi bahasa Indonesia yaitu IndoBERT[[40]](#footnote-40). Pembuatan IndoBERT oleh IndoLEM (Indonesia Language Evaluation Montage) menggunakan konfigurasi yang sama dengan BERT yang telah dijelaskan di atas (BERT base uncased). Isi dari IndoBERT adalah 12 hidden layer dengan jumlah 768, attention heads berjumlah 12, and feed-forward layer sejumlah 3.072. Dan untuk word embedding yang digunakan juga berasal dari WordPiece tapi versi Indonesia berukuran 31.293. IndoBERT telah dilewatkan berbagai sumber data yaitu Wikipedia Indonesia (74 juta kata), berita dari Kompas[[41]](#footnote-41), Tempo[[42]](#footnote-42), Liputan6[[43]](#footnote-43) (55 juta kata) dan Indonesian Web Corpus (90 juta kata).

#### Word2Vec

Awalnya untuk metode Sequence-to-Set, embedding yang kedua secara tertulis bukan Word2Vec yang digunakan namun GLoVE. Namun, penelitian ini juga menyediakan opsional untuk menggunakan representasi vektor antara GLoVE atau Word2Vec. Tugas akhir ini akan menggunakan Word2Vec yang disediakan oleh Dr. Ir. Joan Santoso dengan data bersumber dari Wikipedia Indonesia.

Tujuan adanya Word2Vec bermula karena pada saat penelitian Word2Vec, belum ada teknik yang cukup efisien untuk pengolahan representasi kata yang dapat menerima data yang berlebihan besarnya (sekitar ratusan juta kata). Word2Vec selain memberikan nilai kesamaan antar kata, teknik ini juga memberikan nilai kesamaan yang lebih dari satu (karena selain kemiripan, suatu kata bergantung pada konteks dapat memiliki arti kata lebih dari satu). Dalam penelitian ini juga ditemukan arti dari sebuah kata dapat berarti di luar cara sintaksis, tetapi juga dapat direpresentasikan dari nilai word vector secara hitungan. Contohnya vektor untuk kata “raja” - vektor untuk kata “laki-laki” + vektor untuk kata “perempuan” dapat mengarah pada word vector untuk kata “ratu”.

Word2Vec terinspirasi dari dua jenis arsitektur yaitu Feedforward Neural Net Language Model (NNLM)[[44]](#footnote-44) dan Recurrent Neural Net Language Model (RNNLM). NNLM terdiri dari layer input, *projection*, hidden and output. Secara alur, input kata-kata akan melewati proses encoding 1-of-V (V adalah panjang kosakata). Kemudian diproyeksikan (perkalian matriks seperti regular/dense/linear layer tetapi tanpa fungsi aktivasi diakhir layer seperti sigmoid/tanh) kepada layer projection (P) dengan dimensi *N x D*. Yang menjadi kekurangan dalam NNLM adalah nilai komputasi yang sangat tinggai akibat nilai dari layer proyeksi dapat memberikan matriks yang *dense* (matriks yang *non-zero*, matriks yang setidaknya satu yang tidak bernilai nol). *Q = N × D + N × D × H + H × V* adalah penilaian untuk komputasi NNLM, perlu diperhatikan bagian *N × D × H* adalah yang menitikberatkan nilai komputasi yang besar karena nilai proyeksi (*N x D*) dan nilai layer hidden (*H*) yang besar (layer proyeksi bisa 500 sampai 2000, dan hidden bisa 500 sampai 1000). Untuk meminimalkan nilai komputasi Word2Vec mengusulkan representasi kosakata diubahkan menggunakan *hierarchical softmax* dengan representasi Huffman binary (memberikan kode biner singkat kepada kata yang sering muncul) yang bisa mengurangi jumlah unit output yang perlu dievaluasikan. Terlepas dari cara baru ini bukan percepatan yang penting, Word2Vec akan mengusulkan juga arsitektur tanpa layer hidden yang kemudian akan membuat titik efisiensi berada pada normalisasi softmax.

Perbedaan NNLM dan RNNLM adalah penggunaan RNN yang hanya terdiri dari input, hidden dan output, tidak menggunakan layer proyeksi. RNN dikenal untuk informasi *short term* yang dapat disimpan. Artinya bahwa informasi dalam satu kalimat yang panjang atau kalimat sebelumnya dapat disimpan oleh RNN, membantu memberikan konteks lebih banyak. Juga komputasi yang diberikan oleh RNN adalah *Q = H × H + H × V*.

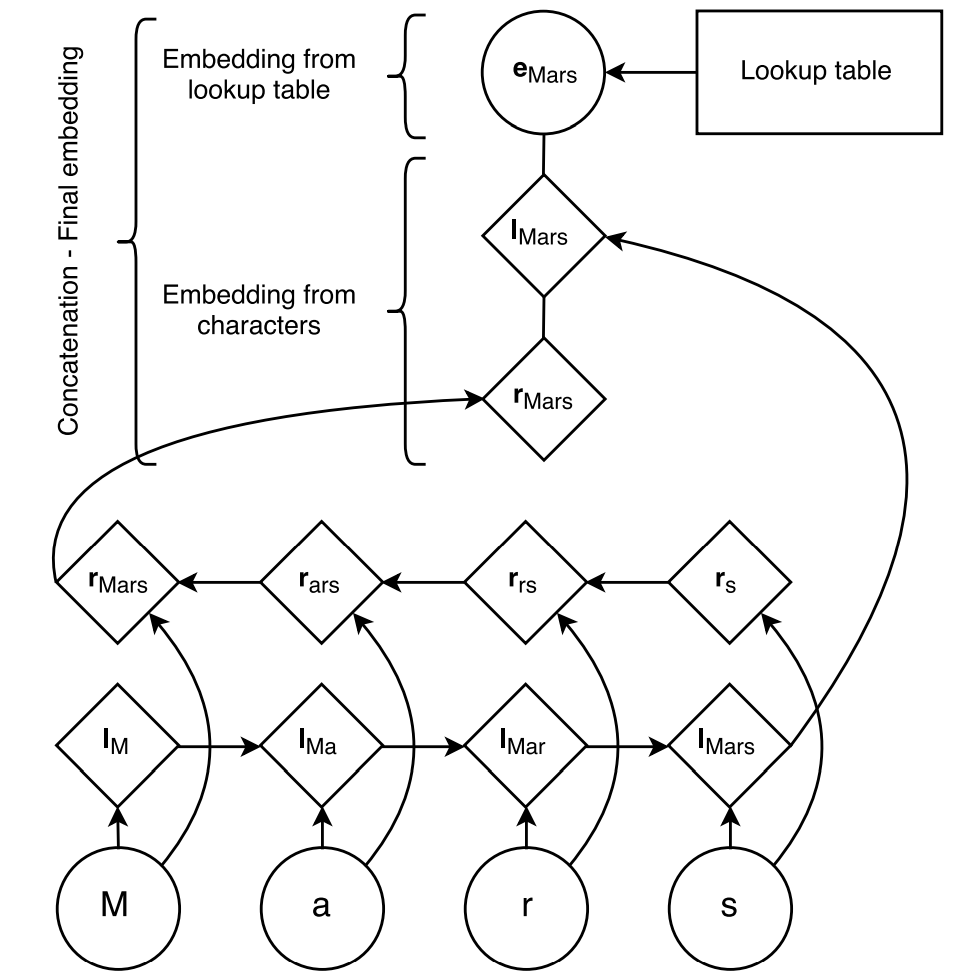
Peneliti Word2Vec menyatakan dari penelitian sebelum-sebelumnya menemukan kesimpulan bahwa NNLM dapat memberikan hasil training yang bagus dengan dua tahap yaitu *continuous* word vector dipelajari dengan model yang sederhana, kemudian dilakukan training N-gram NNLM diatas representasi kata-kata. Kedua tahap itu dapat dilakukan dengan arsitektur Continuous Bag-of-Words Model (C-BOW) dan Continuous Skip-gram Model (Skip-Gram). CBOW memiliki arsitektur yang mirip dengan NNLM namun layer proyeksi diberikan kepada semua kata-kata tidak hanya salah satu (menghasilkan nilai vektor rata-rata). Metode yang digunakan dalam C-BOW juga adalah melihat kata-kata di masa depan (di depan). Menghasilkan nilai komputasi *Q = N × D + D × log2(V)*. Berbeda dengan bag-of-words standar, C-BOW menggunakan representasi konteks yang terdistribusi secara terus menerus. Perlu diketahui juga, matriks weight antara input dan layer poroyeksi juga digunakan untuk semua posisi kata sama dengan cara NNLM.

Arsitektur Skip-Gram memiliki kemiripan juga dengan C-BOW, tetapi tujuan mereka berkebalikan. Di mana C-BOW bertujuan untuk menebak suatu kata berdasarkan konteks kata sekitarnya, Skip-Gram berusaha semaksimal mungkin untuk klasifikasi kata berdasarkan suatu kata lain dalam satu kalimat yang sama. Skip-Gram menggunakan setiap *current word* (kata yang difokuskan) sebagai input ke pengklasifikasi log-linear dengan *continuous projection layer*, dan memprediksi kata-kata dalam jangkauan tertentu sebelum dan sesudah center word tersebut. Ditemukan juga ternyata jika jangkauan melihat sebelum dan sesusah kata tersebut dapat menaikkan kualitas hasil dari word vector, perlu diperhatikan juga bahwa hal ini juga dapat menaikkan nilai komputasi. Namun ada juga pernyataan bahwa lebih banyak kata-kata yang jauh dari center word tidak memiliki relasi yang sedikit (secara konteks). Karena ini, nilai/weight untuk kata yang jauh (*distant words*) akan diperkecil. Untuk Skip-Gram akan memiliki nilai komputasi *Q = C × (D + D × log2(V))*, di mana C adalah jarak maksimal kata-kata sekitarnya.

#### Character-level BiLSTM

Character-level BiLSTM adalah metode embedding yang diambil dari penelitian mengenai Neural Architectures for Named Entity Recognition[[45]](#footnote-45)yang menginginkan input nya memiliki representasi yang sensitif terhadap penulisan karakter. Input word embedding dari penelitian tersebut tidak hanya memiliki representasi character-level, tetapi juga menggunakan pretrained embeddings dan juga menambahkan layer dropout untuk mendorong model untuk bergantung pada kedua representasi tersebut. Meskipun terdapat tiga proses pembuatan embedding dari penelitian tersebut, Sequence-to-Set Network hanya menggunakan metode character-level embedding, namun kedua tahap berikutnya (pretrained embedding dan layer dropout) akan tetap dibahas.

Penggunaan embedding secara character-level memiliki beberapa keuntungan, seperti nilai kegunaan yang tinggi terhadap beberapa tugas NLP seperti bahasa yang secara morfologi cukup beragam, tugas seperti POS Tagging dan language modelling juga untuk dependency parsing yang memiliki kesulitan dengan kata-kata yang tidak dikenal (biasa disebut out-of-vocabulary/OOV). Gambar 4.3 adalah penggambaran arsitektur dari character-level embedding untuk kata “Mars”. Pada awal pembuatan embedding akan diinisialisasi dengan nilai random yaitu *lookup table* embedding untuk tiap karakter. Embedding untuk sebuah kata yang telah dipecah menjadi tiap karakter adalah gabungan dari representasi *forward* (depan dari karakter) dan *backward* (arah ke belakang karakter) dari BiLSTM. Representasi kata kemudian akan digabungkan dengan representasi/embedding word-level nya dari sebuah lookup table word-level. Jika kata tersebut tidak memiliki word embedding, maka akan diberikan embedding token kata yang tidak kenal yang biasanya ditandakan UNK (token UNK kan diberikan probabilitas 0,5). Panjang dimensi hidden untuk character-level biLSTM adalah 25 untuk tiap arah, berarti 50 secara total (nilai ini juga menjadi nilai *default* untuk metode Sequence-to-Set Network).



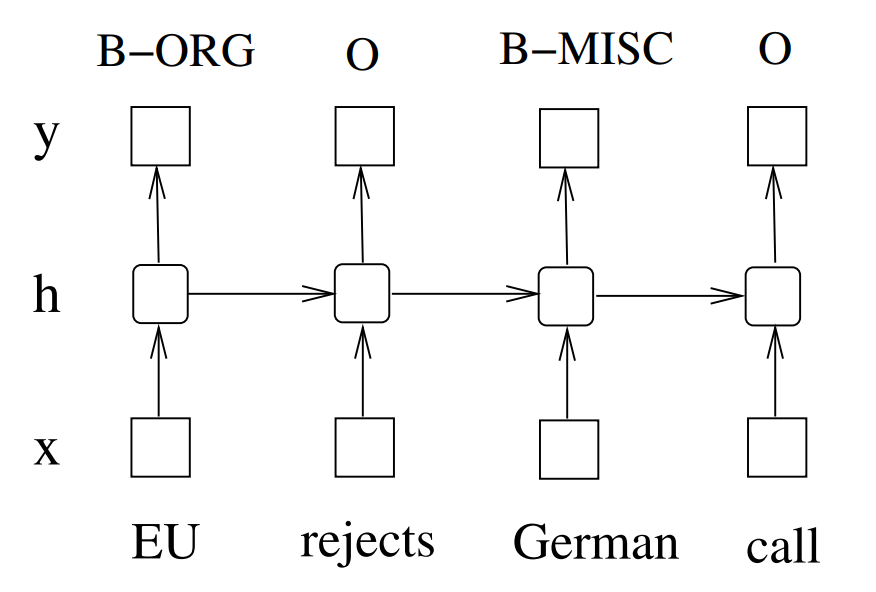
Gambar .  
Arsitektur Character-Level Embedding[[46]](#footnote-46)

Tahap berikutnya adalah menyediakan lookup table word-level dari sebuah pretrained model. Model ini adalah skip-n-gram, sebuah variasi dari Word2Vec dengan memerhatikan penulisan kata secara urutan. Untuk mendapatkan skip-n-gram tentu dilakukan fine tune kepada embedding saat proses training. Penentuan menggunakan word embedding lookup table yang pretrained adalah karena hasil penelitian mengalami peningkatan yang baik dibandingkan dengan word embedding yang random. Dimensi untuk embedding ini adalah 100 untuk bahasa Inggris dan 64 untuk yang lainnya (embedding ini dicoba dalam bahasa Spanyol). Dan minimum frekuensi kata adalah empat kali dan *window size* (jangkauan melihat berapa kata sebelum dan sesudah focus word) adalah delapan.

Dan terakhir adalah layer dropout yang ditambahkan di akhir embedding, sebelum input memasuki arsitektur sebenarnya. Dropout diberikan untuk menghindari model lebih bergantung pada salah satu representasi. Dengan dropout dialami peningkatan yang bagus juga dalam penelitian tersebut, karena model fokus terhadap kedua representasi, tidak salah satu saja.

#### BiLSTM

Bidirectional LSTM[[47]](#footnote-47) (BiLSTM) sendiri akan digunakan sebagai representasi akhir setelah ketiga jenis word embedding diatas digabungkan. Representasi akhir secara detail akan dibahas di subbab encoded tokens, untuk subbab ini akan fokus jalan kerjanya BiLSTM sendiri. Arsitektur BiLSTM terinspirasi dari Long Short Term Memory (LSTM) yang juga terinspirasi dari Recurrent Neural Network (RNN). RNN menjadi contoh dari LSTM karena RNN sebelumnya terbukti memberikan performa yang bagus terhadap tugas language model dan *speech recognition* karena cara penyimpinan informasi (bisa juga disebut sebagai fitur dari kata) waktu lampau, dan informasi ini dapat disimpan dalam jangka waktu yang lama. Gambar 4.4 menunjukkan struktur dari isi RNN dengan input kata-kata “EU rejects German call” dan tiga layer diatasnya yaitu input (*x*), hidden (*h*), dan output (*y*).



Gambar .  
Arsitektur Recurrent Neural Network (RNN)[[48]](#footnote-48)

Contoh dari gambar 4.4 adalah contoh RNN dengan tugas memberikan tag entitas dalam bentuk IOB. Layer input merepresentasikan fitur kata input dalam bentuk one-hot-encoding, vektor dense, atau vektor sparse (vektor dengan isi sebagian besar bernilai nol). Dan layer input merepresentasikan fitur (informasi) pada saat itu/time (dengan anotasi *t*), untuk layer output akan merepresentasikan probabilitas distribusi tag entitas pada time ke *t*. RNN memberikan koneksi antara hidden state sebelumnya dengan hidden state saat ini, karena ini informasi lama dapat selalu dilihat oleh RNN. Untuk nilai dari tiap layer dapat dilihat pada rumus (4.1) dan rumus (4.2), dimana *f* dan *g* adalah fungsi aktivasi *sigmoid* (rumus 4.3) dan *softmax* (rumus 4.4) dan nilai U, W, V adalah koneksi weight yang melewatkan training.

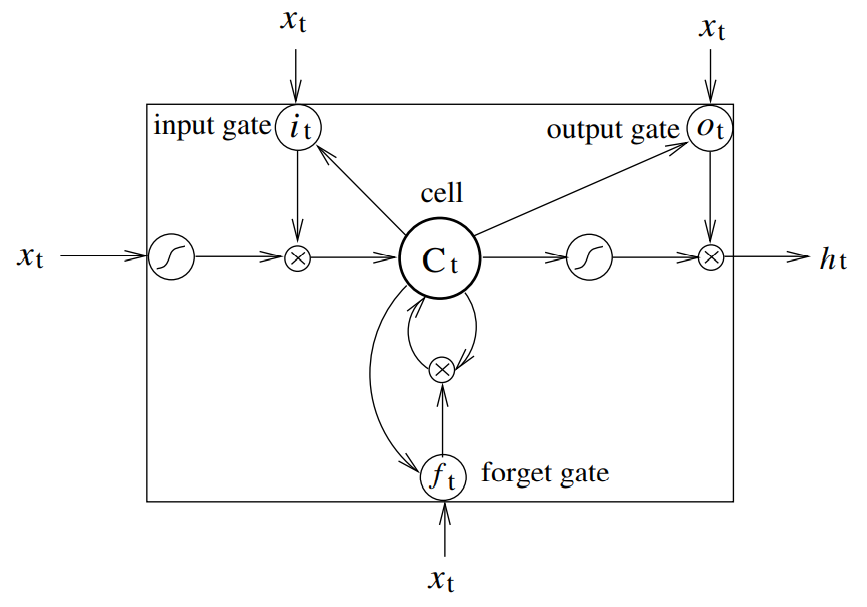
(.)

(.)

(.)

(.)

RNN memang memiliki kemampuan menyimpan informasi berjangka lama, namun RNN memiliki permasalah *vanishing gradient*. Selain keunggulan dalam komputasi, LSTM dapat menyelesaikan permasalahan vanishing gradient dari RNN. LSTM memiliki arsitektur yang sama dengan RNN dengan perbedaan hidden layer diubah dengan sebuah sel yang dinamakan *memory cell*. Dengan bantuan memory cell, LSTM mendapat sifat yang lebih bagus dalam mencari ketergantungan jarak jauh dalam data. Isi dari sebuah memory cell dapat dilihat pada gambar 4.5.



Gambar .  
(Satu) Memory Cell LSTM[[49]](#footnote-49)

Implementasi dari memory cell LSTM menggunakan rumus 4.5, 4.6, 4.7, dan 4.8. Dimana σ sebagai fungsi logistik sigmoid, kemudian *i, f, o* dan *c* adalah gate input, gate forget, gate output dan vektor sell. Setiap dari alur gate memiliki weight matriks dan berguna sesuai dengan namanya masing-masing. Contohnya adalah weight matriks untuk hidden-input gate, untuk weight matriks gate input-output.

(.)

(.)

(.)

(.)

Penelitian yang dirujuk untuk menggunakan BiLSTM[[50]](#footnote-50), memiliki fokus untuk penelitian berbagaim macam LSTM dalam tugas *sequence tagging*. Dan dalam tugas tersebut memberi akses untuk mengetahui fitur (informasi) lampau dan masa depan dalam waktu tertentu. Karena hal ini, dapat menggunakan BiLSTM untuk menggunakan fitur lampau kepada forward hidden state dan fitur masa depan kepada backward hidden state. BiLSTM menggunakan cara training back-propagation through time (BPTT)[[51]](#footnote-51). Backpropagation tetap dilakukan seperti biasanya disebuah network, namun setiap nilai hidden states dapat dilihat dari semua timestep (*t*) yang telah dilewatkan. Dan khusus untuk implementasi BiLSTM pada sequence tagging, akan dilakukan forward dan backward pass untuk satu kalimat utuh juga reset hidden state menjadi nol dilakukan disetiap awal tiap kalimat. Dan proses training kalimat akan dilakukan beberapa secara bersamaan karena implementasi training dalam bentuk *batch* (sekumpulan/sekelompok).

#### Encoded Tokens

Serangkaian input yang diberikan akan dilewatkan akan dibuatkan representasi token ke-i (dilambangkan sebagai ). Hasil dari adalah penggabungan (*concatenation*) dari embedding kontekstual oleh BERT (dilambangkan sebagai ), embedding Word2Vec (dilambangkan sebagai ), embedding character-level (dilambangkan sebagai ). Perlu diingat bahwa Sequence-to-Set Network sebenarnya menggunakan embedding GLoVe dan embedding POS Tag, namun penelitian tugas akhir ini memberi batasan POS Tag tidak diberikan dan GLoVE digantikan dengan embedding Word2Vec. Ketiga jenis embedding telah dijelaskan pada subbab sebelumnya dan juga dari penelitian yang telah dirujukkan. Kemudian dari semua representasi kata, akan berlaku sebagai input kepada BiLSTM (subbab 4.1.1.4) untuk mendapatkan serangkaian representasi yang final (dilambangkan ). Penulisan matematis untuk menghasilkan representasi token dapat dilihat pada rumus 4.9, 4.10, 4.11, dan 4.12. Dimana *l* adalah panjang dari input sequence, *d* hidden size dari LSTM, adalah lambang dari proses concatenation/penggabungan. Kemudian dan parameter untuk forward dan backward LSTM. dan adalah hidden state pada posisi ke *i* dari forward dan backward LSTM.

(.)

(.)

(.)

(.)

### Entity Set Decoder

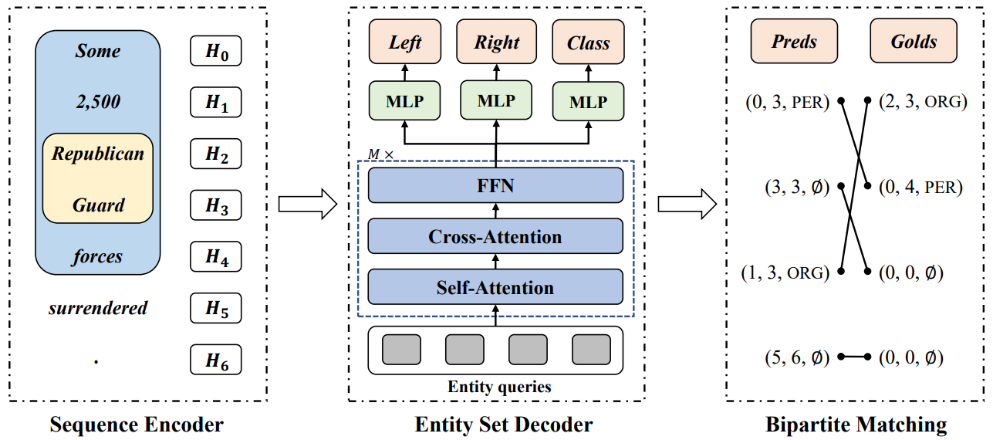
Entity Set Decoder adalah bagian berikutnya dalam Sequence-to-Set Network yang bertugas untuk melakukan prediksi dari kata-kata yang diberikan dari encoder dengan informasi dan kata-kata yang perlu difokuskan oleh decoder. Subbab ini akan dibagi menjadi dua subbab kecil sesuai dengan isinya Entity Set Decoder yaitu layer decoder dan *classification*. Dengan decoder terdiri dari tiga layer, yaitu self-attention, cross-attention, feed forward network. Untuk layer classification akan dijelaskan mengenai tiga Multilayer Perceptron yang digunakan untuk mengkalisifikasi output decoder layer menjadi prediksi akhirnya yaitu batasan awal dan akhir kata yang akan dilabel dan juga jenis label/entitas yang diberikan.

#### Decoder Layer

Bagian decoder dari Sequence-to-Set Network mengambil inspirasi dari Transformers, sehingga menggunakan self-attention dan cross-attention untuk mengubah sejumlah *N* entity queries. Namun ada perbedaan dengan layer decoder dari Transformers yaitu decoder untuk Sequence-to-Set Network dibentuk dengan sifat *non-autoregressive*. Sifat ini yang membentuk model untuk mendapatkan prediksi sebanyak *N* dalam sekali iterasi.

Sifat autoregressive ini didapatkan dari layer decoder Transformers sendiri dari satu layer yang dimiliki Transformer yaitu Masked Multi-Head Attention. Gunanya layer tersebut adalah untuk memberi mask kepada token berikutnya yang akan diprediksi. Contoh dalam kasus translasi bahasa Indonesia menuju bahasa Inggris dengan kalimat “Saya mencintai anda”. Token bahasa Inggris yang perlu diprediksikan adalah “I”, “love”, “you”. Misalkan pada saat ini didalam decoder Transformers telah memprediksikan sampai token kedua / “love”, artinya pada kasus ini yang diberikan mask adalah token ketiga. Hal ini dilakukan agar layer attention yang berada di decoder Transformers akan menekankan fokusnya kepada kedua token pertama (“I”, “love”).

Karena sifat dari decoder Sequence-to-Set Network adalah non-autoregressive, maka tidak akan menggunakan layer Masked Multi-Head Attention. Alasan juga diberikan sifat tersebut adalah keperluan informasi semantik kontekstual lengkap. Meskipun layer Masked Multi-Head Attention tidak dimasukkan dalam decoder ini, Sequence-to-Set Network tetap menggunakan layer-layer umum dari Transformers yaitu self-attention dan cross-attention bahkan juga Feed Forward Network. Dan layer decoder ini dapat ditentukan berapa jumlah tumpukkan layer decoder yang diinginkan (dilambangkan dengan *M*).

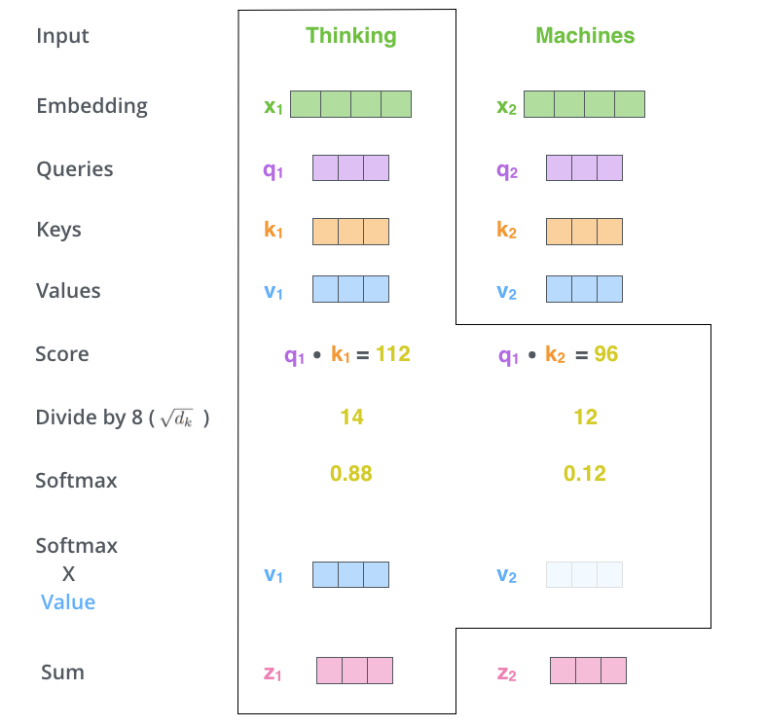


Gambar .  
Arsitektur Bagian Entity Set Decoder[[52]](#footnote-52)

Layer self-attention adalah layer pertama dari bagian encoder. Layer ini berguna untuk memberi fokus antar entitas dan mengambil nilai juga hubungan ketergantungan antar entitas. Dalam Sequence-to-Set Network, layer self-attention akan dibilang sebagai multi-head attention (sama dengan Transformers yang mengenalkan multi-head attention setelah self-attention dijelaskan. Secara singkat, multi-head attention adalah penggabungan dari penghitungan self-attention (dapat dipanggil sebagai attention head). Namun untuk mempermudah penjelasan, akan dijelaskan self-attention terlebih dahulu, kemudian dilanjutkan dengan multi-head attention.

Penggambaran yang mudah untuk ilmu self-attention adalah dari contoh kata “Kucing Andi tidak menyukai permen, ia menyukai susu”. Dari kalimat tersebut, model dapat memahami bahwa subjek “ia” adalah “Kucing Andi”. Untuk manusia tentu gampang untuk memahaminya, namun untuk komputer akan lebih susah. Cara self-attention mendapat informasi tersebut adalah, saat kata “ia” sedang diproses, self-attention memberi kemampuan model untuk melihat kata-kata disekitar sequence kalimat tersebut untuk mencari informasi tersebut.

Penjelasan penghitungan self-attention akan dijelaskan dalam bentuk vektor (implementasi aslinya menggunakan matriks). Tahap pertama untuk melakukan proses self-attention adalah kebutuhan tiga vektor bernama Query, Key dan Value. Ketiga vektor ini diinisialisasi sama dengan weight pada neural network biasa, dengan nilai random. Vektor-vektor ini akan berguna untuk menghitung *score* tiap kata, score merupakan nilai fokus yang perlu diberikan kepada kata tersebut menurut self-attention. Penghitungan score adalah dengan dot matriks antar vektor Query dengan vektor Key yang sedang dinilai. Contohnya jika sedang menghitung score kata pertama, maka score pertama adalah dikalikan dengan . Untuk score kedua maka perkalian dilakukan antara dan . Tiap dari score akan dibagi akar pangkat dimensi vektor didalam penelitian (. Contoh jika dimensi vektor adalah 64 maka tiap score akan dibagi 8, dan nilai baru tersebut dimasukkan kedalam fungsi softmax. Nilai dari softmax akan dikalikan dengan vektor Key dan langkah terakhir adalah menjumlahkan seluruh nilai vektor terbaru, ini menghasilkan output self-attention untuk posisi kata tersebut (dilambangkan vektor *z*). Gambar 4.7 dapat membantu pemahaman alur dari penghitungan self-attention.



Gambar .  
Alur Kalkulasi Self-Attention[[53]](#footnote-53)

Dari self-attention, dikembangkan menjadi multi-head attention dimana keunggulannya adalah attention dapat disebarkan lebih luas pada posisi kata berbeda-beda. Tiap “head” akan memiliki matriks query, key dan value sendiri. Pada Transformers (dan tugas akhir ini) akan menggunakan delapan attention-head sehingga terdapat delapan layer encoder dan decoder juga delapan matriks query, key dan value. Cara penghitungan multi-head attention sama dengan self-attention tetapi dilakukan delapan kali dengan 8 matriks berbedanya menghasilkan delapan matriks *z* yang berbeda-beda. Kedepalan matriks agar dapat diterima oleh layer-layer berikutnya perlu digabung, kemudian hasil dari matriks yang digabung perlu dikalikan dengan matriks (matriks ini telah melewatkan training bersama dengan modelnya). Hasil akhir ini yang akan diberikan kepada layer berikutnya yaitu cross-attention. Pada tugas akhir ini, ditentukan nilai dari query, key dan value akan sama dengan nilai entity queries ().

Layer cross-attention merupakan layer berikutnya setelah self-attention, layer ini dapat secara efektif mengambil informasi konstekstual dari kalimatnya/tokennya. Cross-attention juga akan dihitung secara multi-head attention, tetapi cara penghitungan cross-attention tidak ada perbedaanya dengan self-attention, yang berbeda adalah nilai dari matriks Query, Value dan Key. Dalam Transformers, cross-attention akan menerima matriks Query dan Key dari output encoder paling terakhir (berbeda dengan self-attention yang matriksnya akan dibuat bukan diterima dari layer lain). Nilai dari query, key dan value untuk cross-attention adalah query akan sama dengan nilai entity queries (). Dan nilai key dan value sama dengan representasi token dari encoder (). Secara keseluruhan rumus untuk self, cross, dan multi-head dapat ditulis seperti rumus 4.13, 4.14, dan 4.15. Dengan dan sebagai nilai proyeksi yang dapat diganti (*trainable*) dan R sebagai nilai akhir oleh cross-attention yang akan menjadi input untuk layer Feed Forward Network (FFN).

(.)

(.)

(.)

Layer FFN terdiri dari 3 buah layer perceptron dengan fungsi aktivasi ReLU dan layer proyeksi linear. Layer ini akan menghasilkan embedding final (dilambangkan sebagai ), embedding ini memiliki label tambahan () untuk membantu menandakan kata yang memang tidak memiliki entitas karena panjang dari hasil prediksi akan tetap dengan jumlah *N* entitas, dimana nilai *N* akan ditentukan lebih besar dari pada jumlah entitas yang ada dari sequence nya.

#### Classification Layer

Seperti namanya, layer ini bertugas untuk mengubah prediksi dari layer decoder dan diklasifikasikan sesuai dengan prediksinya. *Classifier* yang digunakan adalah Multi Layer Perceptron (MLP) berjumlah tiga, satu classifier untuk tiap prediksi yang perlu klasifikasikan (batasan kiri, batasan kanan dan jenis label/entitas). Tiap MLP akan juga memiliki fungsi aktivasi softmax didalamnya dan hasil dari tiap MLP adalah probabilitas klasifikasi () untuk tiap jenis label/entitas sendirim batasan kiri, dan batasan kanan. Perumusan untuk klasfikasi dapat dilihat pada rumus 4.16, 4.17, 4.18, dan 4.19, dengan *u* adalah entity query () dari U.

(.)

(.)

(.)

(.)

### Bipartite Matching

Untuk membuat sebuah model mengetahui nilai error nya (melakukan prediksi) saat training memerlukan fungsi menghitung loss. Menjadi kesulitan untuk melakukan penghitungan perbandingan nilai hasil prediksi dengan hasil sebenarnya (disebut juga *golden entities*). Metode *bipartite matching* adalah solusi yang dipilih untuk Sequence-to-Set Network. Sebelum menghitung nilai loss nya perlu dicarikan pasangan optimal diantara set prediksi sejumlah *N* () dan set golden (*y*) nya. Mengingat bahwa set prediksi memiliki jumlah tetap sebanyak *N*, maka set golden (*y*) akan diberikan lapisan pengisi (*padding*) dengan label entitas kosong (). Menghitung nilai *cost* dari penyocokan antara prediksi dengan golden akan menggunakan cara penghitungan algoritma Hungarian[[54]](#footnote-54) (). Rumus untuk algoritma Hungarian dapat ditemukan pada rumus 4.21. Dengan menemukan nilai cost penyocokan prediksi dengan golden, dapat dilakukan pencarian penyocokan paling optimal dari cost yang terrendah (rumus 4.20). Setelah mendapatkan pasangan optimal (), langsung dapat menghitung loss akhirnya yaitu rumus 4.22.

(.)

(.)

(.)

## Library dan Tools

Pembuatan program ini didukung dengan beberapa library dan tools yang membantu mempercepat beberapa tugas. Bagian dari bab ini akan menjelaskan dan mendeskripsikan library yang telah digunakan. Juga beberapa fungsi yang diaplikasikan ke dalam program Sequence-To-Set Network ini.

### PyTorch

PyTorch[[55]](#footnote-55) adalah framework *machine learning* yang bersifat *open source* (kode sumber program tersedia untuk dimodifikasi atau digunakan kembali) yang berdasarkan dari library Torch dan dikembangkan oleh Meta AI. Selain NLP, library ini dapat juga digunakan unutk tugas *computer vision*. Framework ini sudah membantu beberapa program ternama didunia NLP seperti Hugging Face's Transformers[[56]](#footnote-56), PyTorch Lightning[[57]](#footnote-57). Begitupun juga dengan program selain NLP seperti Catalyst[[58]](#footnote-58) (framework untuk Deep Learning) dan Tesla Autopilot.

Beberapa fitur yang sering digunakan pada pengaplikasian NLP dan juga pada tugas akhir ini adalah penggunaan jenis *class* PyTorch (yang juga berasal dari Torch awalnya) adalah Tensor. Mirip dengan NumPy array, PyTorch Tensor dapat dioperasikan dengan NVIDIA GPU yang menggunakan CUDA (tugas akhir ini juga menggunakan jenis GPU tersebut). Ada banyak jenis Tensor yang disediakan, namun tidak akan dijelaskan. Fitur *modules* yang disediakan juga menjadi fitur yang selalu dipakai dalam pembuatan program Sequence-To-Set Network. Beberapa modules dari PyTorch adalah autograd module (efisien dalam menghitung diferensiasi parameter di forward pass), optim module (modul yang memiliki berbagai algoritma optimasi yang digunakan untuk membangun neural network), dan nn module (memudahkan untuk membuat computational graphs dan mengambil nilai gradien).

### Hugging Face (Transformers)

Hugging Face[[59]](#footnote-59) adalah library Python yang menyediakan sekian ribu model yang sudah dilewatkan proses training (pretrained) sehingga dapat langsung dipake untuk pengaplikasian tugas NLP. Dan jenis tugas NLP yang disediakan modelnya beragam dari teks, visual dan audio. Tugas akhir ini menggunakan Hugging Face untuk mengambil library Transformers juga mengambil pretrained model IndoBERT. Beberapa arsitektur terbesar yang umum digunakan dalam penelitian disediakan dalam Hugging Face, contohnya BERT, GPT, GPT-2, RoBERTa, dan lain-lainnya. Hugging Face juga merupakan suatu platform yang sangat besar, memiliki fitur-fitur dalam websitenya seperti membuat model baru, mencari dataset, adapun untuk berkomunitas. Beberapa fungsi yang digunakan dari transformers oleh Hugging Face dalam tugas akhir ini seperti AdamW (optimizer), BertConfig, BertTokenizer .

### Google Colab

Google Colab[[60]](#footnote-60) adalah hasil dari Google Research, siapa saja dapat menulis dan mengeksekusi kode python melalui browser, dan sangat cocok untuk pembelajar interaktif mengenai machine learning, data analysis, dan lainnya. Colab bisa dikatakan sebagai *notebook* Jupyter yang dihosting. Karena tersedia secara online, fasilitas ini tidak membutuhkan setup apapun hanya memerlukan akun Google dan gratis tidak berbayar. Tetapi karena produk gratis, pasti ada beberapa batasan dari pihak Google yaitu penggunaan GPU tidak akan selalu tersedia, tiap notebook memiliki batas waktu *timeout* jika tidak melakukan apapun (tidak interaktif meskipun ada program yang sedang berjalan didalamnya) apabila melewati waktu itu maka koneksi notebook dengan *runtime* akan diputuskan. Dan apabila notebook tersebut ditutup saat sedang menjalankan suatu program, maka runtime akan terputus dan harus menyambungkan dari awal kembali.

Dengan banyaknya batasan terdapat beberapa akun *premium* yang disediakan oleh Google Colab namun berbayar. Terdapat dua jenis akun, Colab Pro dan Colab Pro +. Kedua dari akun tersebut akan mendapatkan GPU dan TPU yang lebih cepat, runtime lebih lama dan memory (RAM) lebih banyak. Untuk perbedaannya, Colab Pro + memiliki fitur eksekusi program dibelakang layer, sehingga user tidak perlu melakukan interaksi dan program dapat berjalan tanpa diputuskan oleh Google Colab sendiri. Dan untuk jumlah waktu runtime, jumlah memory dan GPU tentu lebih banyak daripada Colab Pro.

Tugas akhir ini akan menggunakan Google Colab Pro sebagai bantuan untuk melakukan training model. GPU yang diberikan untuk akun Colab Pro adalah NVIDIA Tesla P100 PCIe 16 GB atau NVIDIA Tesla T4 16GB. Adapun pilihan untuk eksekusi notebook dengan pilihan High-RAM. Namun pilihan itu hanya berlaku untuk satu notebook yang sedang online saja.

## Modifikasi Metode Sequence To Set Network

Penelitian metode pada tugas akhir ini melakukan beberapa modifikasi sesuai dengan fasilitas dan sumber daya yang digunakan. Pertama, bagian representasi kata yang dihasilkan oleh layer encoder merupakan gabungan dari beberapa jenis word embedding. Pada tugas akhir ini tidak akan menggunakan word embedding jenis POS Tag, selain karena dataset tidak disediakan POS Tag, tugas akhir ini mencoba untuk melakukan tugas NER tanpa bantuan POS Tag.

Kedua, adalah parameter freeze\_transformer akan diberikan nilai True, alias tiap kali model melakukan training Transformers tidak akan melakukan training ulang. Sehingga untuk weight yang digunakan adalah weight yang sudah pernah training sebelumnya dari Transformers. Hal ini dilakukan karena terbatasnya GPU yang ada untuk penelitian ini. Training model dilakukan di dua lingkungan (*environment*) yang berbeda. Yang pertama berada di Google Colab dengan GPU 16GB dan lainnya berada di OS Linux dengan GPU GeForce RTX 3070 dengan 8GB.

Untuk menyesuaikan dengan bahasa dan dataset yang ditujukan, Sequence To Set Network dalam bahasa Indonesia akan menggantikan pretrained model untuk word embedding. Yang pertama adalah dari BERT menjadi IndoBERT (dikembangkan oleh IndoLEM). Dan GLoVE akan digantikan dengan Word2Vec model yang disediakan dari bapak Dr. Ir. Joan Santoso, S.Kom, M.Kom.

Karena berada di dua lingkungan yang berbeda, ada beberapa modifikasi yang diberikan khusus untuk tiap lingkungan. Program pada Google Colab akan memiliki fitur *checkpoint* untuk tiap epoch yang dijalankan. Fitur ini disediakan karena durasi *runtime* yang ambigu dari Google Colab Pro. Untuk yang berada di Linux tidak akan diberikan checkpoint, namun modifikasi yang diberikan (karena hanya 8GB) adalah ukuran hidden layer dari arsitektur akan dikurang. Yang seharusnya FFN memiliki ukuran hidden 1024 dan modelnya memiliki ukuran hidden 768. Diubah kedua-duanya menjadi 504 agar dapat menjalankan training di dalam GPU Linux. Karena keterbatasan memory GPU, jumlah epoch yang disarankan dari penelitian Sequence-to-Set Network yang seharusnya 100 epoch akan diubah menjadi 30 epoch.

## Contoh Kasus Penggunaan Sequence To Set

Subbab terakhir ini akan memberikan penjabaran penghitungan matematis yang terjadi dalam serangkaian Sequence-To-Set Network. Untuk input kalimat akan diberikan “Jalan Ir. Soekarno” dan penjelasan akan berawal dari encoder sampai penghitungan loss.

BAB V

# UJI COBA

Bab kelima adalah uji coba yang dilakukan dalam tugas akhir ini. Dimulai dengan pembahasan cara hitung evaluasi dari Sequence-to-Set Network, yaitu Micro dan Macro dari F1 Score. Kemudian uji coba terhadap metode Sequence-to-Set Network. Uji coba akan dijelaskan dalam beberapa subbab bergantung pada konfigurasi yang diubah pada Sequence-to-Set Network. Kemudian akan dijelaskan metode pembanding dengan hasil Sequence-to-Set Network.

## Evaluator Performa Hasil dari Tugas Akhir

Evaluator adalah cara penghitungan performa dari tugas akhir ini, tepatnya metode Sequence-to-Set Network dalam bahasa Indonesia. Performa prediksi sebuah metode biasanya dihitung dengan akurasi prediksi dengan target output/prediksi sebenarnya. Penghitungan ini akan menggunakan library/tool yang membantu penghitungan yang akurat dan benar. Akan ada dua jenis evaluator yaitu Micro F1 Score dan Macro F1 Score. Keduanya akan dijelaskan pada subbab masing-masing.

### Macro dan Micro F1Score

Macro F1 Score adalah penghitungan F1 Score yang paling terus terang karena rumusnya. Macro F1 Score akan mengambil semua F1 Score yang telah dihitung, kemudian langsung dihitung nilai rata-rata tanpa ada perhitungan bobot lainnya. Contoh penghitungan akan diambil dari artikel “Micro, Macro & Weighted Averages of F1 Score, Clearly Explained”[[61]](#footnote-61). Disediakan hasil penghitungan True Positive (TP), False Positive (FP), and False Negative (FN) dari sebuah data klasifikasi gambar pesawat (dilambangkan A), kapal (dilambangkan B) dan mobil (dilambangkan C) sebagai tabel berikut (Tabel 5.1). Tabel tersebut akan menyediakan metrik precision (P), recall (R), dan F1 score untuk tiap label.

Tabel .  
Contoh Data Confusion Matriks

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Label** | **TP** | **FP** | **FN** | **Precision** | **Recall** | **F1 score** |
| Pesawat | 2 | 1 | 1 | 0,67 | 0,67 | = **0,67** |
| Kapal | 1 | 3 | 0 | 0,25 | 1,00 | = **0,40** |
| Mobil | 3 | 0 | 3 | 1,00 | 0,50 | = **0,67** |

Penghitungan precision, recall dan F1 score dapat dilihat dari rumus 5.1, 5.2, dan 5.3. Precision dan recall tidak dapat digunakan sebagai evaluasi suatu performa model karena itu nilai tersebut akan digunakan untuk membantu menghitung F1 Score. F1 Score adalah penghitungan yang dibuat agar dapat melihat precision dan recall seimbang dan penghitungan F1 Score terbukti nilai evaluasi yang bagus (meskipun data mungkin tidak seimbang). Dengan rumus F1 Score yang di tunjukkan pada rumus 5.3, membuktikan jika model mendapat nilai precision dan recall yang tinggi makan nilai F1 Score, begitupun untuk nilai rendah. Jika model memiliki nilai precision dan recall yang salah satunya nilai rendah dan salah satunya lagi nilainya tinggi, akan menghasilkan F1 Score yang rata-rata.

(.)

(.)

(.)

Untuk Macro F1 Score, pada tabel 5.2 menjelaskan bahwa seluruh F1 Score yang telah dihitung sebelumnya akan dijumlah dan dibagi sesuai jumlah label. Dengan kata lain, Macro F1 Score adalah penghitungan F1 Score rata-rata yang tidak berbobot. Artinya bahwa Macro F1 Score menganggap tiap label semua rata tanpa melihat jumlah *support* (jumlah kemunculan label dalam dataset) tiap label.

Tabel .  
Penghitungan Micro dan Macro

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Label** | **TP** | **FP** | **FN** | **F1 score** | **Macro** | **Micro** |
| Pesawat | 2 | 1 | 1 | 0,67 | = **0,58** | = **0,60** |
| Kapal | 1 | 3 | 0 | 0,40 |
| Mobil | 3 | 0 | 3 | 0,67 |

Sedangkan Micro F1 Score akan menghitung nilai rata-rata global. Penghitungan menggunakan True Positive (TP), False Negative (FN), and False Positive (FP) dari semua data. Contoh penghitungan akan diberikan pada tabel 5.2 dan rumus dapat dilihat pada dalam tabel tersebut . Tiap TP, FN dan FP dari seluruh label akan dijumlah dan digunakan dalam rumus Micro F1 Score. Micro F1 Score dapat juga dibilang akurasi (*accuracy*), karena pada dasarnya menghitung proporsi prediksi yang tepat dari semua prediksi. Dengan begitu, definisi tesrebut yang kita gunakan untuk menghitung akurasi secara keseluruhan.

Penggunaan Micro dan Macro perlu diperhatikan, untuk dataset yang tidak seimbang datanya namun tiap jenis data/label/*class* sejajar kepentingannya, maka dapat menggunakan Macro F1 Score. Jika dataset yang dimiliki dianggap cukup seimbang dan ingin nilai metrik yang dapat menyimpulkan performa secara keseluruhan label, lebih baik menggunakan Micro F1 Score. Pada Sequence-to-Set Network jenis F1 Score yang digunakan adalah Macro F1 Score dan Micro F1 Score. Namun untuk perbandingan F1 Score terbaik diambil dari Micro F1 Score.

Seperti yang telah disebut, penghitungan dalam program tugas akhir ini menggunakan bantuan library. Library tersebut adalah *sklearn*, secara khusus fungsi yang dibutuhkan adalah precision\_recall\_fscore\_support dari modul *metrics*. Dengan menggunakan fungsi tersebut, dengan singkat penulisan program langsung mendapat nilai precision, recall, F1 Score, bahkan juga support tiap label. Parameter dari fungsi tersebut yang digunakan adalah y­\_true dan y\_pred diisi dengan golden entites dan entitas yang telah diprediksikan. Kemudian dua parameter lainnya adalah *labels* untuk memberikan jenis class/label yang ada, dalam tugas akhir ini label yang digunakan adalah jenis-jenis entitas. Parameter terakhir yang digunakan adalah *average*, yang digunakan adalah micro dan macro namun selain itu ada beberapa jenis average yang disediakan sklearn seperti *weighted*, *samples*, *binary*.

## Uji Coba Sequence-to-Set Network pada Nested NER

Konfigurasi dari *hyperparameter* yang akan diubah untuk uji coba ini ada empat jenis yaitu *Batch Size*, *Learning Rate*, *Gradient Norm*, *Dropout*. Penilaian akurasi menggunakan Micro F1 Score dan diambil yang terbaik dari seluruh *epoch* yang dilewatkan. Pembagian dataset adalah 90% *training* dan *development* dan 10% untuk *testing*. Total ada 42 hyperparameter yang dapat dikonfigurasikan, namun akan disebut hyperparamter yang akan diubah untuk uji coba.

Nilai *default* untuk tiap hyperparameter adalah batch size sejumlah delapan, learning rate dengan nilai 2e-5, gradient norm dengan nilai 1.0 dan nilai dropout 0.1. Catatan untuk hyperparamter dropout, terdapat tiga hyperparameter yang dapat diubah nilai dropout nya yaitu prop\_drop, lstm\_drop dan char\_lstm\_drop. Hyperparameter tersebut masing-masing yaitu, prop\_drop untuk merubah nilai probabilitas dropout dalam training model Sequence-to-Set Network. lstm\_drop untuk merubah nilai probabilitas dropout untuk embedding akhir BiLSTM. char\_lstm\_drop untuk merubah nilai probabilitas dropout untuk embedding character-level BiLSTM. Uji coba dilakukan dengan cara untuk hyperparameter yang sedang diuji coba akan diganti, sedangkan hyperparameter lainnya mengikuti nilai default.

Untuk tiap gambar Secara kesimpulan dari uji coba Sequence-to-Set Network adalah F1 Score tertinggi adalah uji coba perubahan untuk batch size sebanyak dua dengan F1 Score 72.28%. Detail hasil uji coba untuk tiap jenis entitas, baik F1 Score, support, precision dan recall telah ditampilkan pada tabel 5.3

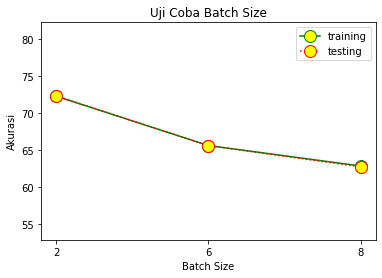
Tabel .  
Hasil Uji Coba F1 Score Terbaik

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis Entitas** | **Precision** | **Recall** | **F1 score** | **Support** |
| TIME | 67.21 | 75.21 | 70.98 | 714 |
| LOC | 71.33 | 70.88 | 71.10 | 1583 |
| EVENT | 67.73 | 48.53 | 56.55 | 307 |
| DATE | 88.87 | 91.36 | 90.10 | 463 |
| PER | 75.62 | 81.10 | 78.26 | 3212 |
| MISC | 49.27 | 56.16 | 52.49 | 479 |
| ORG | 67.10 | 68.04 | 67.57 | 2353 |
| MICRO | 71.00 | 73.60 | 72.28 | 9111 |
| MACRO | 69.59 | 70.18 | 69.58 | 9111 |

### Pengaruh Batch Size

Uji coba pertama adalah perubahan hyperparameter jumlah batch size terhadap nilai akurasi terbaik diakhir uji coba. Nilai untuk batch size yang di uji coba adalah batch size sejumlah 2, batch size sejumlah 6, batch size sejumlah 8 (nilai default). Hyperparameter lainnya akan mengikuti nilai default selama uji coba ini dilakukan. Gambar 5.1 menunjukkan hasil dari uji coba dengan merubah jumlah batch size.

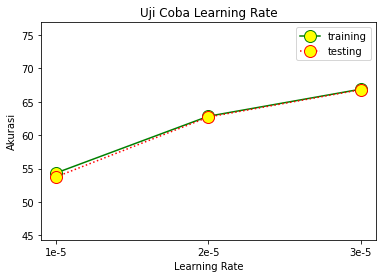
Dapat dilihat perubahan terjadi cukup signifikan untuk batch size jumlah dua menuju batch size jumlah 6, sebanyak 7% (72.28 dan 65.59). Namun untuk perubahan dari batch size jumlah 6 menuju batch size jumlah 8 hanya sebanyak sekitar 3% (62.82 dan 65.59). Untuk akurasi saat proses testing, hasil akurasi tidak memiliki perbedaan jauh, perbedaan ditemukan kurang dari satu (72.31, 65.59, dan 62.7). Kesimpulan sementara untuk pengaruh jumlah batch size adalah mempengaruh tidak terlalu signifikan (kurang lebih 3%). Ada pun kesimpulan berkurangnya nilai F1 Score semakin bertambah jumlah batch size.



Gambar .  
Hasil Uji Coba Jumlah Batch Size

### Pengaruh Learning Rate

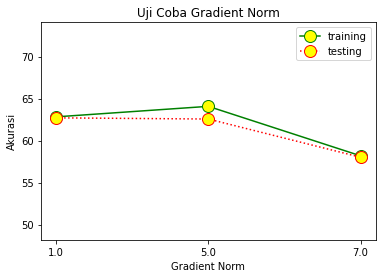
Uji coba berikutnya perubahan hyperparameter nilai learning rate. Nilai untuk learning rate yang di uji coba adalah 1e-5, 2e-5 (nilai default) dan 3e-5. Selain learning rate akan bernilai default selama uji coba ini dilakukan. Gambar 5.2 menunjukkan hasil dari uji coba dengan merubah learning rate. Perubahan yang muncul nilai learning rate 1e-5 menuju learning rate 2e-5, sebanyak 4% (54.34 dan 62.82). Jarak perubahan F1 Score dari uji coba kedua dan ketiga juga sama dengan sebelumnya yaitu 4% (62.82 dan 66.86). Akurasi proses testing tidak berbeda jauh (53.75, 62.7, dan 66.79). Kesimpulan dari uji coba ini tidak terlalu signifikan dan makin bertambah nilainya makin tinggi nilai F1 Score.



Gambar .  
Hasil Uji Coba Learning Rate

### Pengaruh Gradient Norm

Uji coba berikutnya adalah perubahan nilai gradient norm. Nilai yang akan di uji coba adalah 1.0 (nilai default), 5.0, 7.0 dan hyperparameter lainnya mengikuti nilai default pada uji coba ini. Gambar 5.3 menunjukkan hasil dari uji coba dengan perubahan yang muncul adalah dari nilai 1,0 menuju 5,0, terjadi sekitar 2% (62.82 dan 64.07). Tetapi saat nilai dari gradient norm dinaikkan, akurasi mengalami penuruan sebanyak 7% sehingga dapat dikatakan perubahan jika gradient norm dinaikkan mengalami perubaha signifikan namun penuruan.



Gambar .  
Hasil Uji Coba Gradient Norm

### Pengaruh Dropout

Uji coba terakhir adalah perubahan hyperparameter nilai dropout yang akan dilakukan kepada tiga variabel berbeda seperti yang dijelaskan (prop\_drop, lstm\_drop dan char\_lstm\_drop). Nilai drouput yang di uji coba adalah nilai 0.1 (nilai default), 0.3, 0.5. Hyperparameter lainnya akan mengikuti nilai default selama uji coba ini dilakukan. Gambar 5.4 menunjukkan hasil dari uji coba perubahan nilai dropout dan hasil dari perubahannya tidak signifikan dengan perubahan F1 Score yang tidak lebih dari 1% (akurasi training adalah 62.82, 62.46, 62.95). Perbandingan dengan akurasi test juga tidak beda jauh (62.7, 62.44, dan 62.6). Kesimpulan untuk pengaruh perubahan ketiga variabel dropout adalah mempengaruh tidak signifikan (kurang dari 1%).



Gambar .  
Hasil Uji Coba Dropout

## Perbandingan Metode Span-Based Method

Nested NER dari sebelumnya memiliki metode yang dianggap sudah cocok untuk digunakan sebagai penelitian, dan metode ini sudah berkembang banyak dalam Nested NER. Metode ini menganggap pengenalan entitas dengan klasifikasi span (rentangan, suatu bagian dengan batasan kiri dan kanan). Tetapi kekurangan dari metode span-based ini dengan konsep span, *search space* (luas pencarian) menjadi terlalu besar dan metode span-based tidak melihat nilai konteks satu entitas dengan entitas lain, yang sebenarnya dapat membantu model untuk menentukan entitas berdasarkan konteksnya. Karena itu, adanya Sequence-to-Set Network untuk memberikan solusi komputasi yang tidak seberat metode span-based tetapi memberikan akurasi yang tinggi.

Perbandingan dengan metode ini tidak dengan dataset tugas akhir ini namun dengan dataset GENIA. Alasannya karena metode pembanding membutuhkan komputasi yang besar dan resource dari tugas akhir ini tidak memadainya. Kedua, dengan dataset GENIA dan word embedding/model pretrained yang berhubungan dengan kata-kata biologi akan lebih stabil sebagai perbandingan.

Metode yang akan dibandingkan adalah metode Locate and Label[[62]](#footnote-62). Locate and Label adalah metode span-based terbaru pada saat tugas akhir ini dikerjakan. Tidak hanya sebagai metode span-based/klasifikasi span, metode ini menggunakan *identifier* entitas yang memiliki dua tahap. Pertama, akan diusulkan span dengan filter dan *boundary regression* pada *seed span* untuk menemukan entitas, kemudian diberikan label kepada span yang diusulkan dan disesuaikan batas dengan kategori yang sesuai. Perbandingan F1 Score untuk metode Sequence-to-Set dengan Locate and Label dapat dilihat dari tabel 5.4. Dari tabel tersebut dapat dilihat F1 Score yang dimiliki keduanya hanya berbeda 0,10%. Meskipun begitu, Locate and Label menang dengan persentase 80,54% sedangkan Sequence-to-Set dengan 80,44%. Keunggulan dari Locate and Label adalah keberhasilan metode tersebut dalam metrik Recall yang menjadi 80,89% sedangkan Sequence-to-Set dengan 78,66%. Yang unggul Sequence-to-Set adalah arsitektur sistem yang lebih hemat dibandingkan dengan Locate and Label.

Tabel .  
Tabel Perbandingan F1 Score

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Precision** | **Recall** | **F1 Score** |
| Lin et al. (2019) | 75.80 | 73.90 | 74.80 |
| Luo and Zhao (2020) | 77.40 | 74.60 | 76.00 |
| Wang et al. (2020b) | 78.10 | 74.40 | 76.20 |
| Strakova et al. (2019) | - | - | 78.31 |
| Wang et al. (2020a) | 79.45 | 78.94 | 79.19 |
| Yu et al. (2020) | 81.80 | 79.30 | 80.50 |
| Sequence-to-Set (2021) | 82.31 | 78.66 | 80.44 |
| Locate and Label (2021) | 80.10 | 80.89 | 80.54 |

## Catatan Uji Coba

Pelaksanaan uji coba tugas akhir ini memiliki beberapa kesulitan, kesulitan utamanya adalah kurangnya *resource* (fasilitas/sumber daya) secara memory untuk menjalankan arsitektur Sequence-to-Set secara keseluruhan. Penelitian pada awal dijalankan pada *environment* OS Linux (dapat dilihat dalam Lampiran B) dengan GPU GeForce Nvidia RTX 3070 8GB. Karena tidak memadai, arsitektur diupayakan di kecilkan dengan cara Autocast[[63]](#footnote-63) yaitu hidden size tidak dikurangi namun beberapa dari weight akan dibuang untuk meringankan komputasi, cara ini tidak dapat memenuhi kebutuhan dari arsitektur. Kemudian hidden size dari Transformer dan FFN dikurangi dari nilai awal 782 dan 1028 menjadi rata 504 untuk tiap dimensi. GPU dapat menjalankan metode namun mendapatkan akurasi yang kurang bagus karena pemotongan hidden size yang sekitar 50%.

Dengan ini, diputuskan membutuhkan resource yang lebih besar, sehinga penelitian menggunakan Google Colab Pro yang menyediakan GPU Nvidia Tesla T4 16GB. Tetapi GPU tersebut masih belum bisa memenuhi Transformers, sehingga penelitian dilanjutkan dengan Transformers tidak di training ulang (freeze), namun Autocast tidak digunakan, dan hidden size disesuaikan kembali 782 dan 1028. Ada pun beberapa halangan dalam Google Colab Pro karena beberapa fitur yang tidak dapat diakses. Idle runtime diberikan secara random, sehingga notebook tidak dapat ditinggalkan sepenuhnya, harus dipantau agar tidak terputus koneksi secara mendadak (unexpected error, dapat dilihat pada lampiran B). Ditengah pelaksanaan uji coba juga terjadi runtime disconnected, yang berarti Google Colab tidak mengijinkan komputasi berat tanpa aktifitas yang interaktif (terlalu idle). Karena penggunaan GPU akun tugas akhir ini terlalu berat dan tidak interaktif, maka Google Colab memberikan timeout selama 5 jam[[64]](#footnote-64).

BAB VI

# PENUTUP

Sebelum menutup pembahasan mengenai tugas akhir ini, bab terakhir menjelaskan mengenai poin-poin kesimpulan yang ditemukan saat pelaksanaan pengerjaan tugas akhir. Yang ditemukan selain kesimpulan dalam bab ini adalah saran-saran dari penulis untuk penelitian topik ini maupun topik lainnya yang menyinggung topik ini. Saran ini disediakan agar dapat dikembangkan dan dapat digunakan dikemudian hari. Berikut ini adalah kesimpulan dan saran yang telah ditemukan.

## Kesimpulan

Subbab pertama, akan menjelaskan mengenai kesimpulan-kesimpulan yang ditemukan selama mengerjakan dan menganalisa topik tugas akhir ini yaitu penggunaan Sequence-to-Set Network kepada Nested NER dalam bahasa Indonesia. Kesimpulan ini mencakup seluruh proses pengerjaan tugas akhir dari awal sampai akhir, juga kendala-kendala yang dilewatkan dan kesimpulan dari pengerjaan dan kendala yang diterima. Kesimpulan tersebut meliputi:

* Pelabelan dataset adalah proses yang perlu dilakukan dengan teliti dan dengan konsisten, dengan dataset yang konsisten model dapat mengetahui informasi dan konteks yang benar untuk mengetahui kata apa yang perlu diberikan label. Hal ini berlaku tidak hanya untuk Nested NER tetapi juga NER.
* Sequence-to-Set Network menerima bahasa selain bahasa yang ditelitinya (bahasa Inggris), namun untuk model pretrained/word embedding perlu disediakan yang berada dalam domain yang sama. Domain dari tugas akhir ini adalah berita politik, namun untuk pretrained yang digunakan memiliki mayoritas kata memiliki konteks/domain yang lain. IndoBERT berisi 74 juta kata dari Wikipedia Indonesia dan Indonesian Web Corpus sejumlah 90 juta kata. Sedangkan untuk berita yang berasal dari Kompas, Tempo, Liputan6 hanya berjumlah 55 juta kata.
* Hasil dari model Sequence-to-Set Network ini kurang bisa bekerja pada kalimat yang pendek, kemungkinan dampak adalah dataset berita yang terbiasa dengan kalimat panjang dan memiliki banyak konteks.
* *Resource* (fasilitas/sumber daya) untuk melakukan training metode ini tidak memadai dengan kebutuhan arsitektur, dengan hidden size Tranformers sebanyak 768 dan hidden size FFN 1024 tidak cukup dengan GPU 8GB dan 16GB.
* Uji coba yang memiliki F1 Score tertinggi adalah perubahan jumlah batch size sebanyak dua dengan akurasi 72.28% dan apabila jumlah tersebut ditingkatkan akurasi akan menurun.

## Saran

Pada bagian subbab ini akan menjelaskan mengenai saran atau pendapat yang didapat dari penelitian dan uji coba dari penulis untuk penelitian metode Sequence-to-Set Network, penggunaan Transformers, word embedding atau pun topik Nested NER. Saran ini diharapkan dapat digunakan untuk pengembangan tugas akhir ini untuk kedepannya. Berikut adalah saran yang didapat selama proses penelitian tugas akhir ini.

* Pelabelan dataset perlu ada persetujuan untuk kata-kata yang muncul keambiguan, kata yang perlu dilabelkan disarankan memberikan ambigu sesedikit mungkin. Contohnya untuk kata seperti “polisi itu” dengan “polisi” terlalu ambigu untuk dilabelkan.
* Pemilihan pretrained model word embedding yang digunakan yang akan digunakan perlu diperhatikan karena akan memberikan dampak terhadap representasi kata dan proses encoding untuk metode Sequence-to-Set Network.
* Penggunaan Transformers dengan GPU 8GB dan 16GB tidak memenuhi apa yang dibutuhkan. Untuk menanggulangi masalah ini perlu ada pengorbanan pengurangan arsitektur seperti pengurangan hidden size, menggunakan metode Autocast[[65]](#footnote-65).

DAFTAR PUSTAKA

Zeqi Tan, dkk, A Sequence-to-Set Network for Nested Named Entity Recognition, (2021).

Christopher Marshall, “What is named entity recognition (NER) and how can I use it?” (https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d)

Jana Strakova, Milan Straka, Jan Hajic. “Neural architectures for nested ner through linearization”, Proceedings of ACL 2019, 2019.

Mohammad Golam Sohrab and Makoto Miwa. “Deep exhaustive model for nested named entity recognition”, Proceedings of EMNLP 2018, 2018.

Georgia Nikita, Skripsi: “Service Oriented Nested NER untuk Ekstraksi Keyword Entitas di Portal Berita Bahasa Indonesia” (Surabaya: 2022).

Christian Nathaniel Puerwono, Skripsi: Ekstraksi Entity dan Relasi Dalam Bahasa Indonesia Menggunakan Bidirectional LSTM” (Surabaya: 2018).

Amelinda Tjandra Dewi, Skripsi:Named Entity Recognition dan Coreference Resolution Nama Orang untuk Teks Bahasa Indonesia dengan Menggunakan Conditional Random Fields. (Surabaya:2018).

Arliyanti Nurdin, dkk, Perbandingan Kinerja Word Embedding Word2vec, Glove, Dan Fasttext Pada Klasifikasi Teks, Jurnal TEKNOKOMPAK Vol. 14 No. 2 (2020), hal. 74—79.

Yoav Goldberg, Neural Network Methods in Natural Language Processing, (April 2017).

Xiaoya Li, dkk, A Unified MRC Framework for Named Entity Recognition, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Juli 2020), Hal 5849

Plato, “Cara Menggunakan Pengenalan Entitas Bernama (NER) Untuk Ekstraksi Informasi”, (https://zephyrnet.com/id/cara-menggunakan-nama-pengenalan-entitas-untuk-ekstraksi-informasi/)

Ashish Vaswani, dkk, Attention Is All You Need, 2017.

Michael Phi, “Illustrated Guide to Transformers- Step by Step Explanation”, https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Jay Alammar, “The Illustrated Transformer”, http://jalammar.github.io/illustrated-transformer/ diakses, 4 Mei 2022, 12:32

Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117

Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press

Frank Rosenblatt. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory 85, 460–461 (1957)

Prof. Dr. Ir. Kuswara Setiawan, M.T., Buku Paradigma Sistem Cerdas, (Malang : Bayu Media, 2003)

Wasserman, P.D., Schwartz, T., Neural networks. II. What are they and why is everybody so interested in them now?, IEEE Expert, 1988, Volume 3, Issue 1, Hal. 10-15

Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to Sequence Learning with Neural Networks, 2014

H. W. Kuhn, The Hungarian Method for The Assignment Problem

BRAT, mini-introduction to brat, https://brat.nlplab.org/introduction.html

Kripke, Saul, Identity and Necessity, M.K. Munitz (ed.). Identity and Individuation. New York: New York University Press, (New York, 1971), pp. 135–64

Kim Sang, Erik F. Tjong, Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition, COLING-02: The 6th Conference on Natural Language Learning 2002, (2002).

Chinchor, Nancy, MUC-7 Named Entity Task Definition, (1997).

Finkel, Jenny Rose, Manning, Christopher D., Nested Named Entity Recognition, (2009).

Byrne, Kate, Nested Named Entity Recognition in Historical Archive Text, ICSC ’07: Proceedings of the International Conference on Semantic Computing (2007), hal. 589– 596.

Kim, J.D., dkk., GENIA corpus—a semantically annotated corpus for bio-textmining, Bioinformatics (2003), Vol. 19 Suppl. 1 2003, hal. i180–i182

Shachi Language Research Search, ACE 2004 Multilingual Training Corpus, http://shachi.org/resources/593, 2017.

Linguistic Data Consortium, ACE 2005 Multilingual Training Corpus, http://catalog.Idc.upenn.edu/ldc2006t06, 2018.

National Institute of Standards and Technology, Text Analysis Conference (TAC) 2017, https://tac.nist.gov/2017/index.html, 2017.

Kim, J.D., dkk., GENIA corpus—a semantically annotated corpus for bio-textmining, Bioinformatics (2003), Vol. 19 Suppl. 1 2003, hal. i180–i182

CNN Indonesia, https://www.CNNIndonesia.com

Liputan 6, https://www.liputan6.com

Jacob Devlin, dkk, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, In Proceedings of NAACL 2019, pages 4171–4186, Minneapolis, Minnesota, June 2019

Yonghui Wu, dkk, Google’s neural machine translation system: Bridging the gap between human and machine translation, (2016). Fajri Koto, dkk, IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP, (2020).

Kompas, https://kompas.com

Koran TEMPO, https://koran.tempo.co

Bengio Y, Ducharme R., Vincent P., A Neural Probabilistic Language Model, Journal of Machine Learning Research, 3: hal.1137-1155, (2003).

Guillaume Lample, dkk, Neural Architectures for Named Entity Recognition. In Proceedings of NAACL 2016, hal. 260–270, (2016).

Zhiheng Huang , Wei Xu, Kai Yu, Bidirectional LSTM-CRF Models for Sequence Tagging, (2015).

Mikael Boden, A Guide To Recurrent Neural Networks And Backpropagation, In the Dallas Project, (2002)

GitHub - Pytorch (https://github.com/pytorch/pytorch)

Pytorch-Transformers, (https://pytorch.org/hub/huggingface\_pytorch-transformers/)

GitHub - PyTorch Lightning, (https://github.com/PyTorchLightning/pytorch-lightning/)

GitHub - Catalyst, (https://github.com/catalyst-team/catalyst)

Hugging Face, (https://huggingface.co)/

Google Colab, (https://colab.research.google.com/?utm\_source=scs-index).

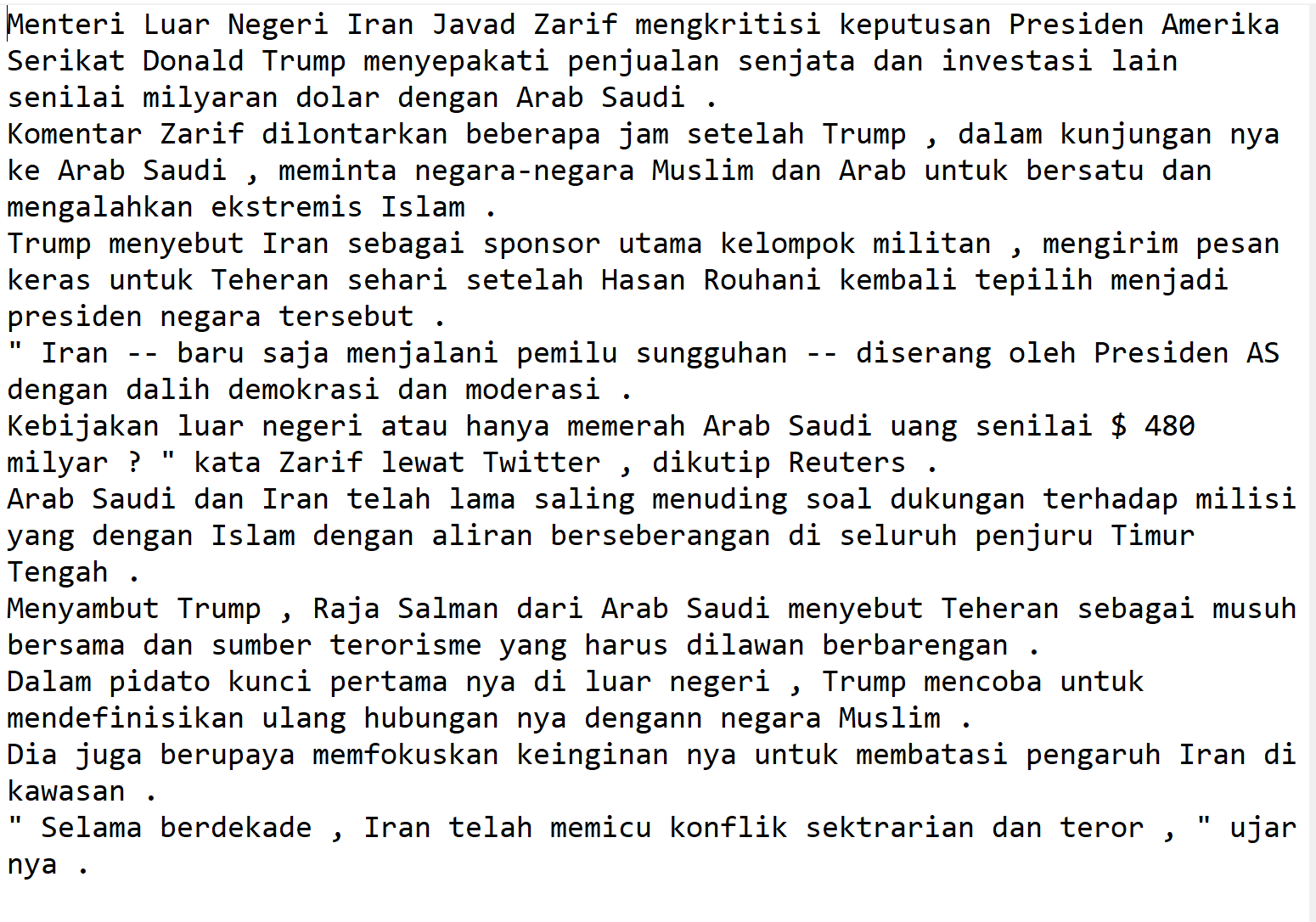
Kenneth Leung, Micro, Macro & Weighted Averages of F1 Score, Clearly Explained, (https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f#2f35)

/1901.08149.

|  |  |
| --- | --- |
| **RIWAYAT HIDUP** | |
|  | **Nama :** Nicoletta Valencia Halim  **Alamat Asal :** Citraland, Taman Internasional 1 B6 No 15  **Tempat/Tanggal Lahir :** Surabaya, 08-08-2000 |
| **Jenjang Pendidikan :**   * 2004 – 2006 * 2006 – 2012 * 2012 – 2015 * 2015 – 2018 * Sejak 2018 | TK Margie, Probolinggo  SD Margie, Surabaya  SMP Margie, Surabaya  SMA Kr. Petra 1, Surabaya  Institut Sains dan Teknologi Terpadu Surabaya |
| **Pengalaman Kerja :**   * 2018 – sekarang | Asisten Dosen Wakil Rektor 1 ISTTS |

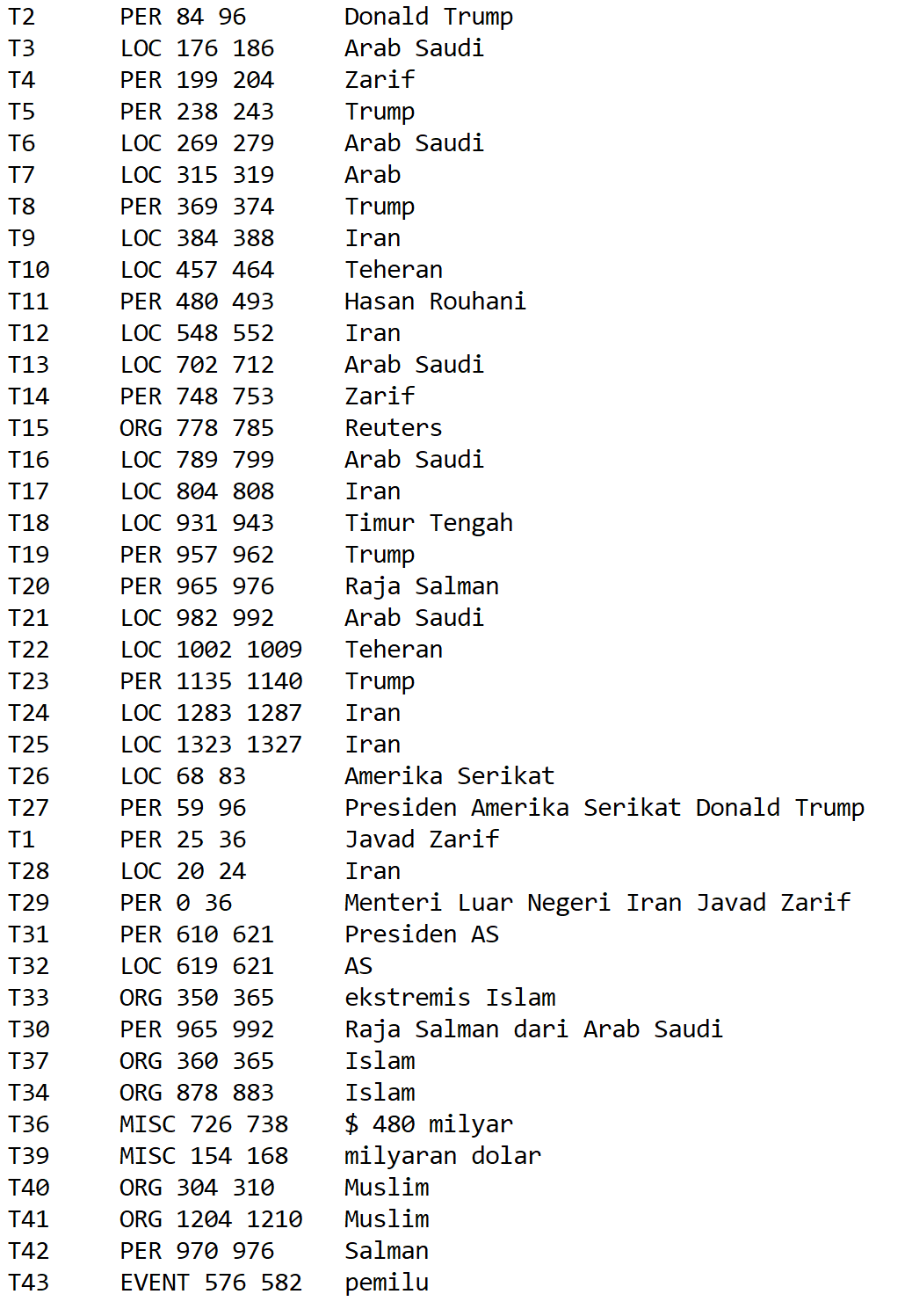
**LAMPIRAN A**

CONTOH DATASET NER



**Gambar A.1**

**File .TXT**



**Gambar A.2**

**File .ANN**

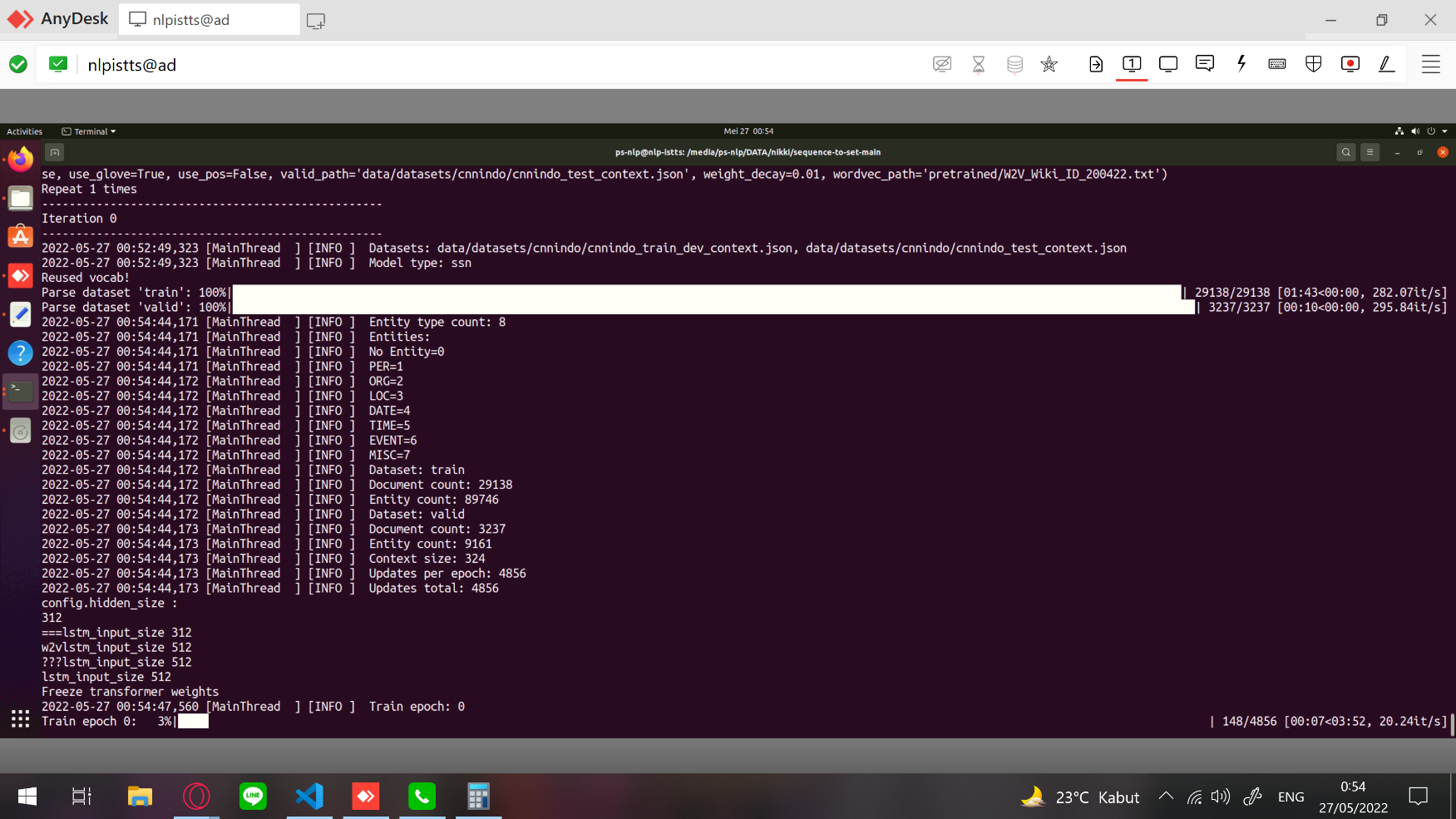


**Gambar A.3**

**Visualisasi BRAT**

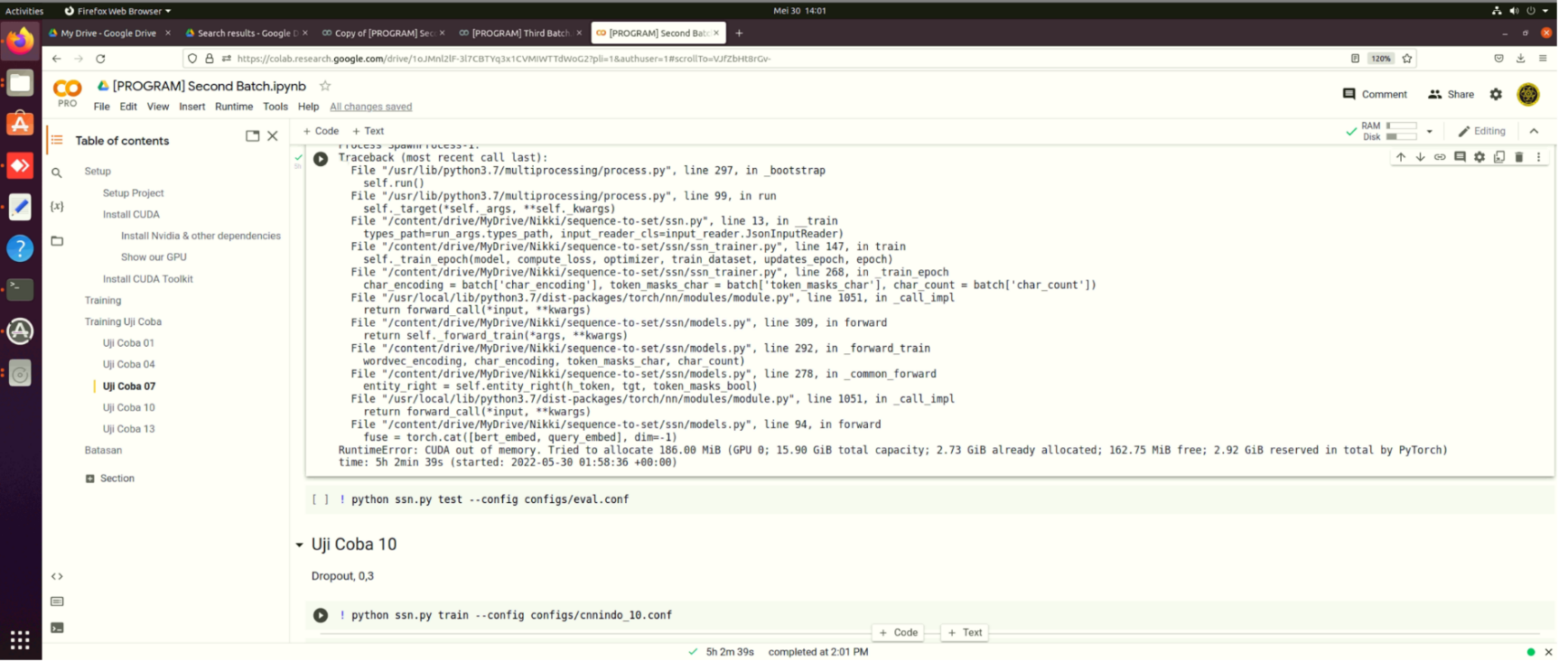
**LAMPIRAN B**

ENVIRONMENT UJI COBA



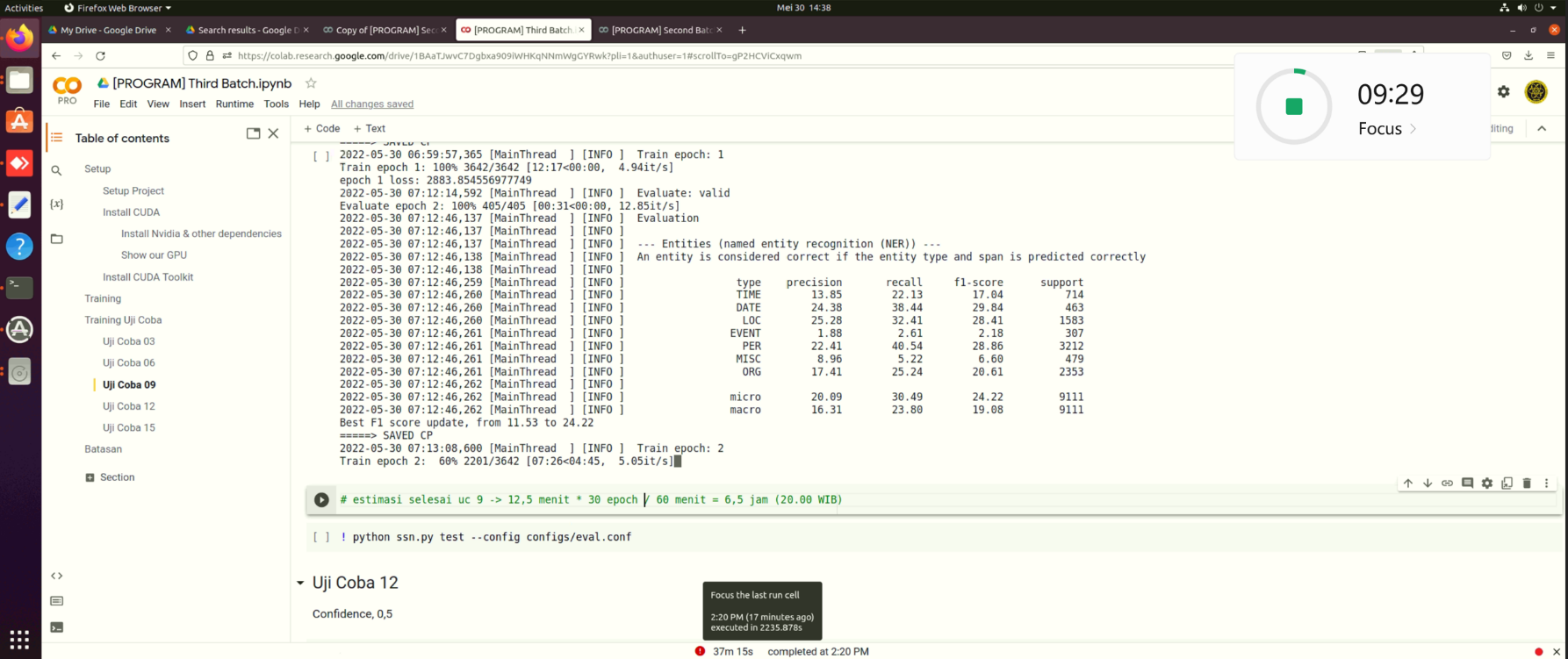
**Gambar A.1**

**Environment Linux**



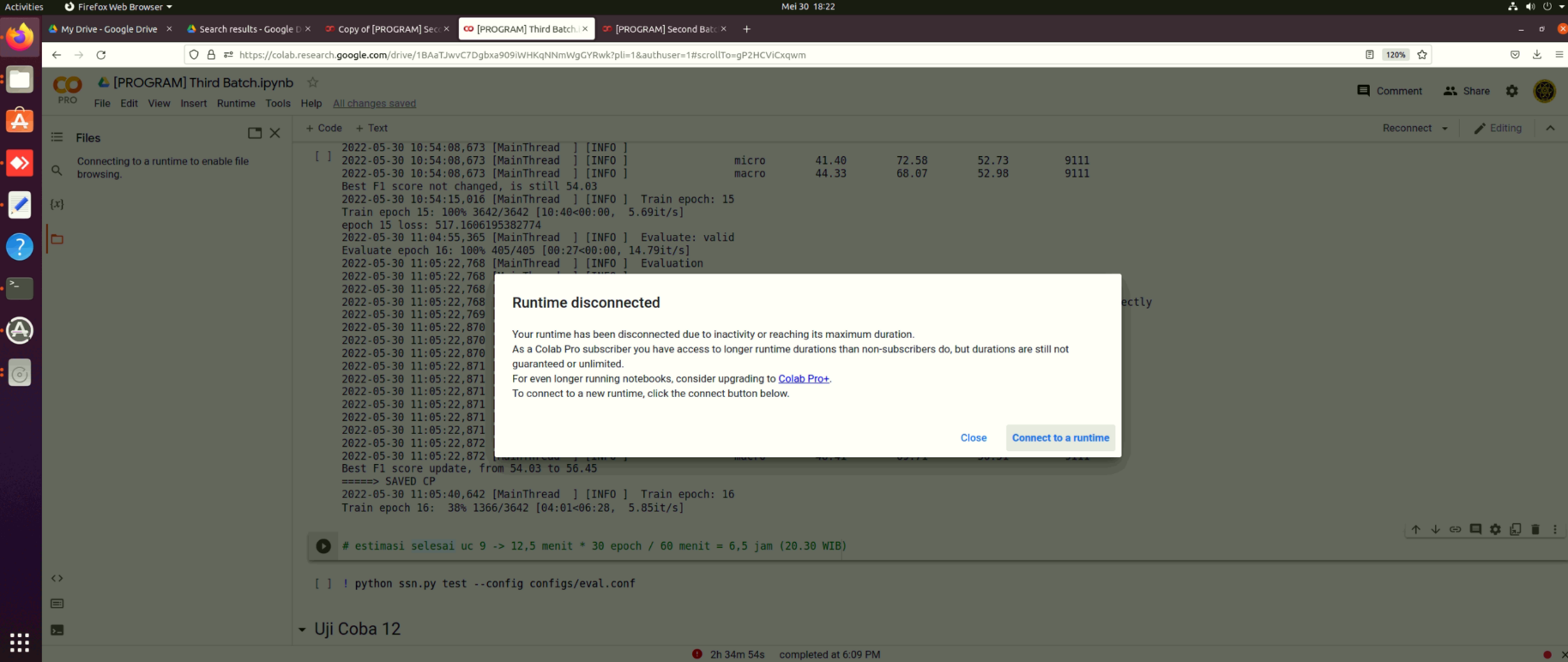
**Gambar A.2**

**Google Colab – Error: CUDA out of memory**



**Gambar A.3**

**Google Colab – Error: Unexpected error**



**Gambar A.4**

**Google Colab – Error: Runtime Disconnected**

1. Christopher Marshall, “What is named entity recognition (NER) and how can I use it?” (https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d) [↑](#footnote-ref-1)
2. Jana Strakova, Milan Straka, Jan Hajic. “*Neural architectures for nested ner through linearization*”, Proceedings of ACL 2019, 2019. [↑](#footnote-ref-2)
3. Mohammad Golam Sohrab and Makoto Miwa. “*Deep exhaustive model for nested named entity recognition*”, Proceedings of EMNLP 2018, 2018. [↑](#footnote-ref-3)
4. Georgia Nikita, Skripsi: “Service Oriented Nested NER untuk Ekstraksi Keyword Entitas di Portal Berita Bahasa Indonesia” (Surabaya: 2022). [↑](#footnote-ref-4)
5. Christian Nathaniel Puerwono, Skripsi: Ekstraksi Entity dan Relasi Dalam Bahasa Indonesia Menggunakan Bidirectional LSTM” (Surabaya: 2018). [↑](#footnote-ref-5)
6. Amelinda Tjandra Dewi, Skripsi:Named Entity Recognition dan Coreference Resolution Nama Orang untuk Teks Bahasa Indonesia dengan Menggunakan Conditional Random Fields. (Surabaya:2018). [↑](#footnote-ref-6)
7. Arliyanti Nurdin, dkk, Perbandingan Kinerja Word Embedding Word2vec, Glove, Dan Fasttext Pada Klasifikasi Teks, Jurnal TEKNOKOMPAK Vol. 14 No. 2 (2020), hal. 74—79. [↑](#footnote-ref-7)
8. Yoav Goldberg, Neural Network Methods in Natural Language Processing, (April 2017). [↑](#footnote-ref-8)
9. Xiaoya Li, dkk, A Unified MRC Framework for Named Entity Recognition, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Juli 2020), Hal 5849 [↑](#footnote-ref-9)
10. Plato, *“Cara Menggunakan Pengenalan Entitas Bernama (NER) Untuk Ekstraksi Informasi”*, (https://zephyrnet.com/id/cara-menggunakan-nama-pengenalan-entitas-untuk-ekstraksi-informasi/) [↑](#footnote-ref-10)
11. Ashish Vaswani, dkk, Attention Is All You Need, 2017. [↑](#footnote-ref-11)
12. Michael Phi, “*Illustrated Guide to Transformers- Step by Step Explanation*”, https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0 [↑](#footnote-ref-12)
13. Jay Alammar, “*The Illustrated Transformer*”, http://jalammar.github.io/illustrated-transformer/ diakses, 4 Mei 2022, 12:32 [↑](#footnote-ref-13)
14. Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117 [↑](#footnote-ref-14)
15. Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press [↑](#footnote-ref-15)
16. Frank Rosenblatt. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory 85, 460–461 (1957) [↑](#footnote-ref-16)
17. Prof. Dr. Ir. Kuswara Setiawan, M.T., Buku Paradigma Sistem Cerdas, (Malang : Bayu Media, 2003) [↑](#footnote-ref-17)
18. Wasserman, P.D., Schwartz, T., Neural networks. II. What are they and why is everybody so interested in them now?, IEEE Expert, 1988, Volume 3, Issue 1, Hal. 10-15 [↑](#footnote-ref-18)
19. Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to Sequence Learning with Neural Networks, 2014 [↑](#footnote-ref-19)
20. H. W. Kuhn, The Hungarian Method for The Assignment Problem [↑](#footnote-ref-20)
21. BRAT, mini-introduction to brat, https://brat.nlplab.org/introduction.html [↑](#footnote-ref-21)
22. Kripke, Saul, Identity and Necessity, M.K. Munitz (ed.). Identity and Individuation. New York: New York University Press, (New York, 1971), pp. 135–64 [↑](#footnote-ref-22)
23. Kim Sang, Erik F. Tjong, Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition, COLING-02: The 6th Conference on Natural Language Learning 2002, (2002). [↑](#footnote-ref-23)
24. Chinchor, Nancy, MUC-7 Named Entity Task Definition, (1997). [↑](#footnote-ref-24)
25. Finkel, Jenny Rose, Manning, Christopher D., Nested Named Entity Recognition, (2009). [↑](#footnote-ref-25)
26. Byrne, Kate, Nested Named Entity Recognition in Historical Archive Text, ICSC ’07: Proceedings of the International Conference on Semantic Computing (2007), hal. 589– 596. [↑](#footnote-ref-26)
27. Kim, J.D., dkk., GENIA corpus—a semantically annotated corpus for bio-textmining,  Bioinformatics (2003), Vol. 19 Suppl. 1 2003, hal. i180–i182 [↑](#footnote-ref-27)
28. Shachi Language Research Search, ACE 2004 Multilingual Training Corpus, http://shachi.org/resources/593, 2017. [↑](#footnote-ref-28)
29. Linguistic Data Consortium, ACE 2005 Multilingual Training Corpus, http://catalog.Idc.upenn.edu/ldc2006t06, 2018. [↑](#footnote-ref-29)
30. National Institute of Standards and Technology, Text Analysis Conference (TAC) 2017, https://tac.nist.gov/2017/index.html, 2017. [↑](#footnote-ref-30)
31. Kim, J.D., dkk., GENIA corpus—a semantically annotated corpus for bio-textmining,  Bioinformatics (2003), Vol. 19 Suppl. 1 2003, hal. i180–i182 [↑](#footnote-ref-31)
32. Georgia Nikita, Skripsi: “Service Oriented Nested NER untuk Ekstraksi Keyword Entitas di Portal Berita Bahasa Indonesia” (Surabaya: 2022). [↑](#footnote-ref-32)
33. Christian Nathaniel Puerwono, Skripsi: Ekstraksi Entity dan Relasi Dalam Bahasa Indonesia Menggunakan Bidirectional LSTM” (Surabaya: 2018). [↑](#footnote-ref-33)
34. Amelinda Tjandra Dewi, Skripsi:Named Entity Recognition dan Coreference Resolution Nama Orang untuk Teks Bahasa Indonesia dengan Menggunakan Conditional Random Fields. (Surabaya:2018). [↑](#footnote-ref-34)
35. CNN Indonesia, *https://www.CNNIndonesia.com* [↑](#footnote-ref-35)
36. Liputan 6, *https://www.liputan6.com* [↑](#footnote-ref-36)
37. Zeqi Tan, dkk, A Sequence-to-Set Network for Nested Named Entity Recognition, (2021). [↑](#footnote-ref-37)
38. Jacob Devlin, dkk, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, In Proceedings of NAACL 2019, pages 4171–4186, Minneapolis, Minnesota, June 2019. [↑](#footnote-ref-38)
39. Yonghui Wu, dkk, Google’s neural machine translation system: Bridging the gap between human and machine translation, (2016). [↑](#footnote-ref-39)
40. Fajri Koto, dkk, IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP, (2020). [↑](#footnote-ref-40)
41. Kompas, https://kompas.com [↑](#footnote-ref-41)
42. Koran TEMPO, https://koran.tempo.co [↑](#footnote-ref-42)
43. Liputan6, https://liputan6.com [↑](#footnote-ref-43)
44. Bengio Y, Ducharme R., Vincent P., A Neural Probabilistic Language Model, Journal of Machine Learning Research, 3: hal.1137-1155, (2003). [↑](#footnote-ref-44)
45. Guillaume Lample, dkk, Neural Architectures for Named Entity Recognition. In Proceedings of NAACL 2016, hal. 260–270, (2016). [↑](#footnote-ref-45)
46. Ibid [↑](#footnote-ref-46)
47. Zhiheng Huang , Wei Xu, Kai Yu, Bidirectional LSTM-CRF Models for Sequence Tagging, (2015). [↑](#footnote-ref-47)
48. Ibid [↑](#footnote-ref-48)
49. Ibid [↑](#footnote-ref-49)
50. Ibid [↑](#footnote-ref-50)
51. Mikael Boden, A Guide To Recurrent Neural Networks And Backpropagation, In the Dallas Project, (2002). [↑](#footnote-ref-51)
52. Zeqi Tan, dkk, A Sequence-to-Set Network for Nested Named Entity Recognition, (2021). [↑](#footnote-ref-52)
53. Jay Alammar, The Illustrated Transformer, (https://jalammar.github.io/illustrated-transformer/). [↑](#footnote-ref-53)
54. Harold W Kuhn, The Hungarian Method for The Assignment Problem. Naval research logistics quarterly, 2(1-2): hal.83–97, (1955). [↑](#footnote-ref-54)
55. GitHub - Pytorch (https://github.com/pytorch/pytorch) [↑](#footnote-ref-55)
56. Pytorch-Transformers, (https://pytorch.org/hub/huggingface\_pytorch-transformers/) [↑](#footnote-ref-56)
57. GitHub - PyTorch Lightning, (https://github.com/PyTorchLightning/pytorch-lightning/) [↑](#footnote-ref-57)
58. GitHub - Catalyst, (https://github.com/catalyst-team/catalyst) [↑](#footnote-ref-58)
59. Hugging Face, (https://huggingface.co)/ [↑](#footnote-ref-59)
60. Google Colab, (https://colab.research.google.com/?utm\_source=scs-index). [↑](#footnote-ref-60)
61. Kenneth Leung, Micro, Macro & Weighted Averages of F1 Score, Clearly Explained, (https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f#2f35) [↑](#footnote-ref-61)
62. Yongliang Shen, Locate and Label: A Two-stage Identifier for Nested Named Entity Recognition, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), (Agustus : 2021). [↑](#footnote-ref-62)
63. Automatic Mixed Precision Package, (https://pytorch.org/docs/stable/amp.html). [↑](#footnote-ref-63)
64. Google Colab FAQ, https://research.google.com/colaboratory/faq.html [↑](#footnote-ref-64)
65. Automatic Mixed Precision Package, (https://pytorch.org/docs/stable/amp.html) [↑](#footnote-ref-65)