|  |
| --- |
| 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

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer tincidunt arcu ullamcorper, imperdiet ante ac, finibus nisl. Donec pulvinar sapien enim, quis tincidunt libero tincidunt in. Integer blandit augue in nulla tempus, ut efficitur erat posuere. Suspendisse malesuada arcu sit amet gravida tincidunt. Nulla maximus ut velit eu pharetra. Nulla nec efficitur nulla. Suspendisse turpis felis, hendrerit vel augue sit amet, vestibulum convallis tortor. Sed commodo dolor in sem mollis, eget molestie elit bibendum.

Donec ultricies, risus tempor feugiat suscipit, dui neque aliquet enim, id mattis sapien leo et nisl. Aliquam at bibendum lorem, nec imperdiet eros. Aliquam eu augue sagittis, maximus ipsum ac, suscipit tellus. Integer libero odio, scelerisque nec condimentum quis, auctor et nulla. Ut cursus maximus erat quis pulvinar. Aliquam purus augue, maximus non malesuada ac, maximus a turpis. Duis non massa tristique, tempus justo nec, placerat odio. Vestibulum varius et nisl vitae rhoncus. Etiam pellentesque orci tempor leo consectetur, vitae placerat elit hendrerit. Quisque aliquet lacinia molestie. Nunc iaculis, mauris imperdiet auctor porta, ante lorem hendrerit erat, at molestie arcu lectus ac felis. Nunc mollis erat sit amet commodo suscipit. Morbi sollicitudin nisl nec suscipit vulputate. Nunc porta, urna sit amet semper molestie, libero urna ullamcorper enim, eget egestas massa orci in odio. Cras ultricies, enim ac consectetur ultricies, ligula leo ultricies lacus, eget sollicitudin sem dolor non enim.

Quisque aliquet facilisis laoreet. Pellentesque eget semper elit. Cras dignissim elementum massa, at tincidunt felis varius nec. Mauris eget magna maximus, convallis est eget, aliquet ipsum. Nulla vulputate leo ac ante lobortis eleifend. Nulla sit amet sem egestas, tincidunt velit in, vehicula urna. Nunc faucibus tincidunt risus non finibus. Maecenas at odio est. Vestibulum quis libero leo. Nunc et efficitur erat. Etiam tincidunt blandit risus nec vulputate. Suspendisse vestibulum magna in sapien pellentesque posuere.

# ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer tincidunt arcu ullamcorper, imperdiet ante ac, finibus nisl. Donec pulvinar sapien enim, quis tincidunt libero tincidunt in. Integer blandit augue in nulla tempus, ut efficitur erat posuere. Suspendisse malesuada arcu sit amet gravida tincidunt. Nulla maximus ut velit eu pharetra. Nulla nec efficitur nulla. Suspendisse turpis felis, hendrerit vel augue sit amet, vestibulum convallis tortor. Sed commodo dolor in sem mollis, eget molestie elit bibendum.

Donec ultricies, risus tempor feugiat suscipit, dui neque aliquet enim, id mattis sapien leo et nisl. Aliquam at bibendum lorem, nec imperdiet eros. Aliquam eu augue sagittis, maximus ipsum ac, suscipit tellus. Integer libero odio, scelerisque nec condimentum quis, auctor et nulla. Ut cursus maximus erat quis pulvinar. Aliquam purus augue, maximus non malesuada ac, maximus a turpis. Duis non massa tristique, tempus justo nec, placerat odio. Vestibulum varius et nisl vitae rhoncus. Etiam pellentesque orci tempor leo consectetur, vitae placerat elit hendrerit. Quisque aliquet lacinia molestie. Nunc iaculis, mauris imperdiet auctor porta, ante lorem hendrerit erat, at molestie arcu lectus ac felis. Nunc mollis erat sit amet commodo suscipit. Morbi sollicitudin nisl nec suscipit vulputate. Nunc porta, urna sit amet semper molestie, libero urna ullamcorper enim, eget egestas massa orci in odio. Cras ultricies, enim ac consectetur ultricies, ligula leo ultricies lacus, eget sollicitudin sem dolor non enim.

Quisque aliquet facilisis laoreet. Pellentesque eget semper elit. Cras dignissim elementum massa, at tincidunt felis varius nec. Mauris eget magna maximus, convallis est eget, aliquet ipsum. Nulla vulputate leo ac ante lobortis eleifend. Nulla sit amet sem egestas, tincidunt velit in, vehicula urna. Nunc faucibus tincidunt risus non finibus. Maecenas at odio est. Vestibulum quis libero leo. Nunc et efficitur erat. Etiam tincidunt blandit risus nec vulputate. Suspendisse vestibulum magna in sapien pellentesque posuere.

# 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. Papa, mama, saudara dan ibu baptis yang telah memberikan dukungan dan motivasi baik dalam penyelesaian Tugas Akhir ini.
2. Bapak Dr. 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.
3. 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.
4. 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. Sehingga kritik dan saran yang bersifat membangun sangat diharapkan demi kesempurnaan 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 Tugas Akhir ini dapat memberikan manfaat bagi para pembaca. Semoga Tuhan yang Maha Esa selalu memberkati dan menyertai kita semua. Amin.

Surabaya, Juni 2022

Penulis

# DAFTAR ISI

SURAT PERNYATAAN KEASLIAN 3

ABSTRAK 4

ABSTRACT 5

KATA PENGANTAR 6

DAFTAR ISI 8

DAFTAR TABEL 11

DAFTAR ALGORITMA 12

DAFTAR SEGMEN PROGRAM 13

DAFTAR RUMUS 14

BAB 1 PENDAHULUAN 15

1.1 Latar Belakang 15

1.2 Tujuan 17

1.3 Batasan Penelitian 17

1.4 Sistematika Pembahasan 18

BAB 2 TEORI PENUNJANG 20

2.1 Word Embedding 20

2.2 Named Entity Recognition (NER) 22

2.3 Transformer 23

2.3.1 Bagian Encoder 24

2.3.2 Bagian Decoder 28

2.4 Neural Network (NN) 30

2.4.1 Multi-Layer Perceptron (MLP) 31

2.5 Sequence to Sequence Models 33

2.6 Hungarian Match 35

2.7 BRAT 35

BAB 3 ARSITEKTUR SISTEM 41

3.1 Arsitektur Sistem Umum 41

3.2 Arsitektur Sistem dari Pra Proses 42

3.3 Arsitektur Sistem dari Proses Prediksi Nested NER 46

3.4 Arsitektur Sistem dari Proses Postprocess 48

# DAFTAR GAMBAR

Gambar Halaman

[2.1 Contoh Visualisasi (a) Bag of Words dan (b) Word Embedding 7](#_Toc103217260)

[2.2 Contoh Nilai Semantik Tersimpan dari Word Embedding 8](#_Toc103217261)

[2.3 Contoh Pengenalan Entitas Bernama 9](#_Toc103217262)

[2.4 Arsitektur Transformers 9](#_Toc103217263)

[2.5 Representasi Transformer bagian Encoder Decoder 10](#_Toc103217264)

[2.6 Contoh Isi dari Matriks Score 11](#_Toc103217265)

[2.7 Representasi Tiap Layer Dari Arsitektur Transformer 14](#_Toc103217266)

[2.8 Visualisasi Arsitektur MLP dengan Backpropagation 17](#_Toc103217267)

[2.9 Bentuk Ringkasan Arsitektur Seq2Seq 20](#_Toc103217268)

[2.10 Contoh Anotasi Text-Span pada BRAT 22](#_Toc103217269)

[2.11 Contoh Normalization Annotation 22](#_Toc103217270)

[3.1 Arsitektur Sistem 28](#_Toc103217271)

[3.2 Arsitektur Sistem Pra Proses 29](#_Toc103217272)

[3.3 Tampilan File Teks pada BRAT 29](#_Toc103217273)

[3.4 Tampilan Window Fitur Pelabelan 30](#_Toc103217274)

[3.5 Struktur Data (a) Data Anotasi BRAT (b) Data Input Model 31](#_Toc103217275)

[3.6 Arsitektur Sistem Proses Prediksi Nested NER 32](#_Toc103217276)

# DAFTAR TABEL

Tabel Halaman

2.1 Pola Ukuran 6

2.2 Paragraf 8

2.3 Contoh Penulisan 10

2.4 Kutipan Satu Paragraf 12

# DAFTAR ALGORITMA

Algoritma Halaman

2.1 Pola Ukuran 6

2.2 Paragraf 8

2.3 Contoh Penulisan 10

2.4 Kutipan Satu Paragraf 12

# DAFTAR SEGMEN PROGRAM

Segmen Program Halaman

2.1 Pola Ukuran 6

2.2 Paragraf 8

2.3 Contoh Penulisan 10

2.4 Kutipan Satu Paragraf 12

# DAFTAR RUMUS

Rumus Halaman

2.1 Pola Ukuran 6

2.2 Paragraf 8

2.3 Contoh Penulisan 10

2.4 Kutipan Satu Paragraf 12

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 entity, 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. Input kalimat hanya dalam 2 - 3 kalimat.
4. Dataset bersumber dari berita dari situs berita CNN Indonesia yang digunakan dalam tugas akhir Georgia Nikita (218116685).
5. Representasi kata Part-of-Speech Tagging tidak digunakan.
6. Jika ada, akan dilakukan penyesuaian terhadap metode yang digunakan dalam menyelesaikan permasalahan pada tugas akhir ini.
7. 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

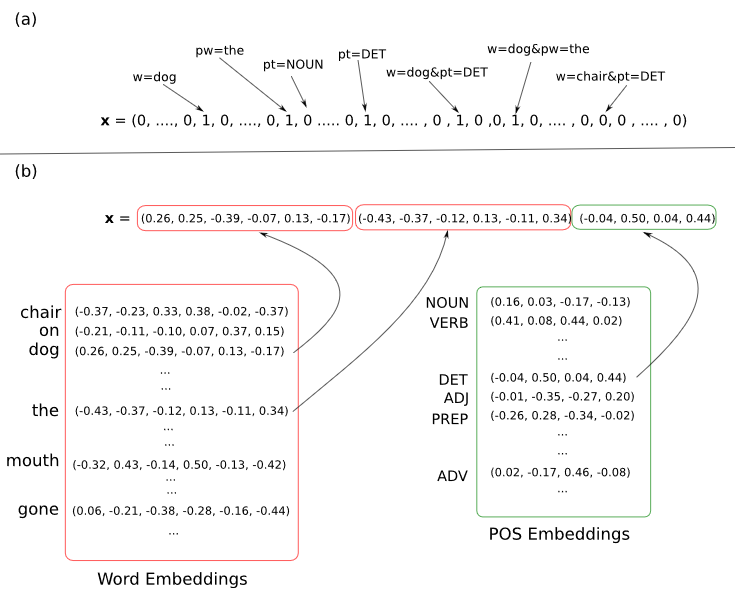
# TEORI PENUNJANG

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. [[4]](#footnote-4)

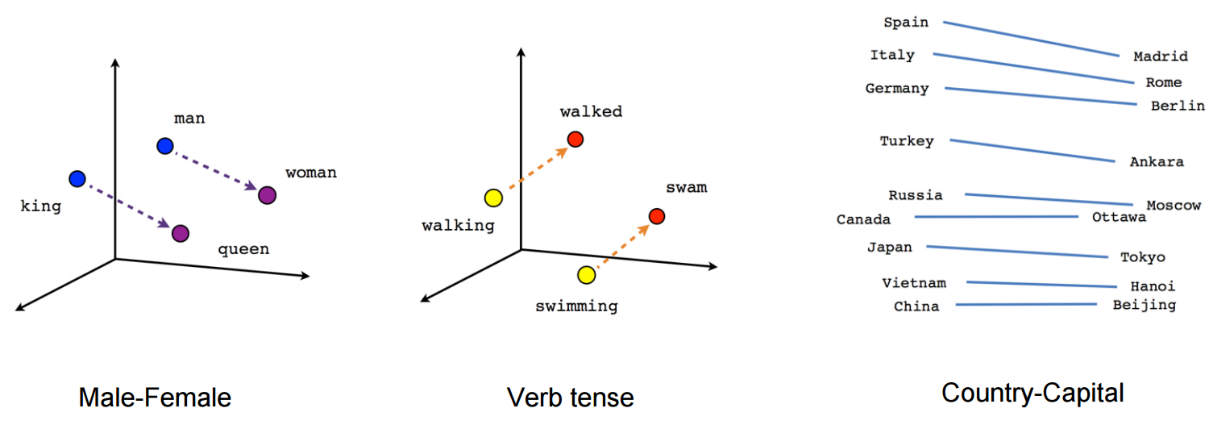
Secara visualisasi, hasil dari word embedding adalah sekumpulan vektor yang berisikan angka riil. Proses konversi kata menjadi vektor. Mengapa word embedding ini menjadi tren dalam penelitian NLP saat ini, salah satunya karena kemampuannya juga untuk memberikan representasi dalam bentuk *dense* (elemen *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 2.1  
Contoh Visualisasi (a) Bag of Words dan (b) Word Embedding

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.

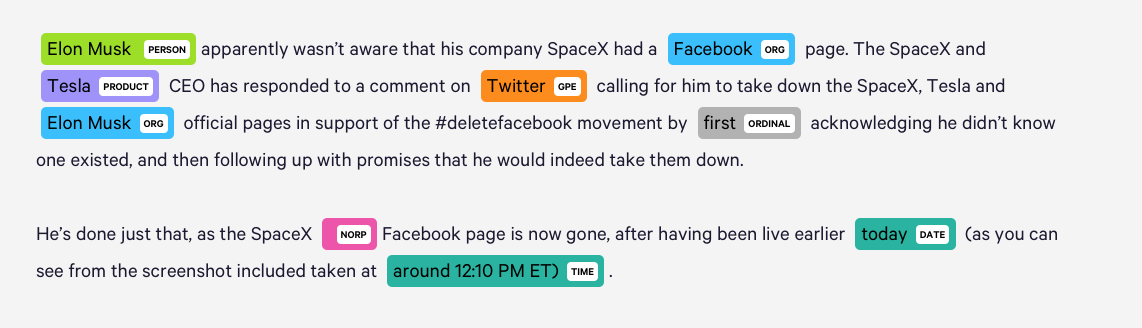
Dengan representasi dense dari word embedding, tiap nilai dari sebuah dimensi memiliki arti. Salah satu yang bisa dicontohkan adalah nilai semantik antar kata lainnya. Dari gambar di bawah menggambarkan visualisasi nilai semantik yang dapat direpresentasikan. Relasi antar kata yang bisa diberikan seperti 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 2.2   
Contoh Nilai Semantik Tersimpan dari Word Embedding

## Named Entity Recognition (NER)

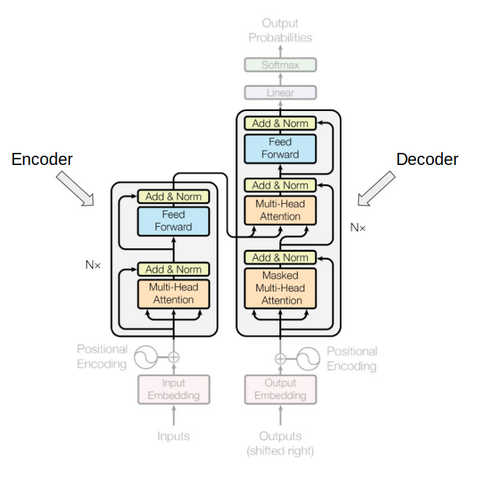
Named Entity Recognition (NER) merupakan task NLP untuk mencari entitas bernama dalam suatu kalimat kemudian menggolongkan entitas tersebut kepada kategori entitas yang ada dari sebuah dokumen/kalimat.[[5]](#footnote-5) Sebuah entitas bernama adalah istilah untuk menyebut sebuah entitas yang ada dalam 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.



Gambar 2.3   
Contoh Pengenalan Entitas Bernama[[6]](#footnote-6)

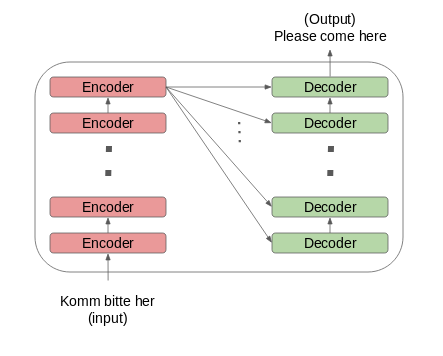
## Transformer

Konsep adanya Transformer 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. Penggunaan Transformer memberi solusi kepada kesulitan seq2seq. 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 2.4  
Arsitektur Transformers[[7]](#footnote-7)

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.



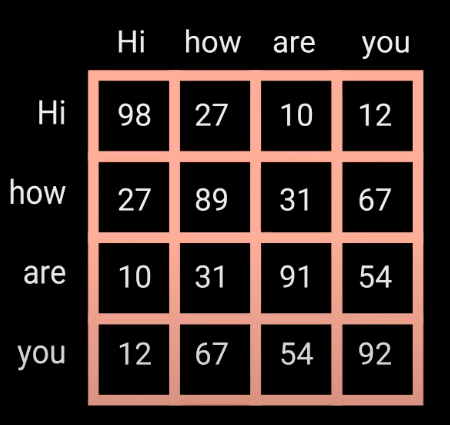
Gambar 2.5  
Representasi Transformer bagian Encoder Decoder

### 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). [[8]](#footnote-8) 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  
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. 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. 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). Dari penjelasan sebelumnya langkah-langkah self-attention didapatkan rumus sebagai berikut:

(2.1)

(2.2)

(2.3)

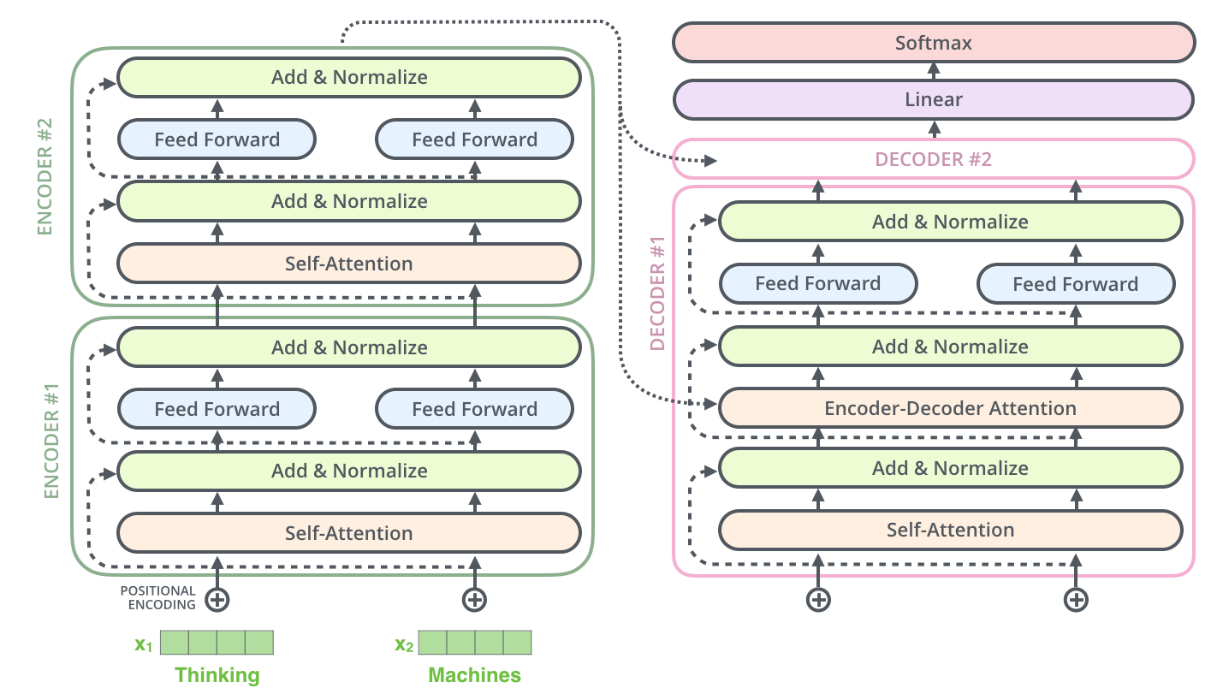
Sebelum lanjut kepada bagian kedua yaitu FFN, perlu diketahui adanya sublayer setelah multi-head attention dan FFN. Sublayer ini adalah *residual connection* dan normalisasi. Residual connection adalah proses menambahkan *positional encoding* dengan input. Selain di dua sublayer tersebut, positional 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.

(2.4)

(2.5)

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.

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  
Representasi Tiap Layer Dari Arsitektur Transformer[[9]](#footnote-9)

### 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-anttention, 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.

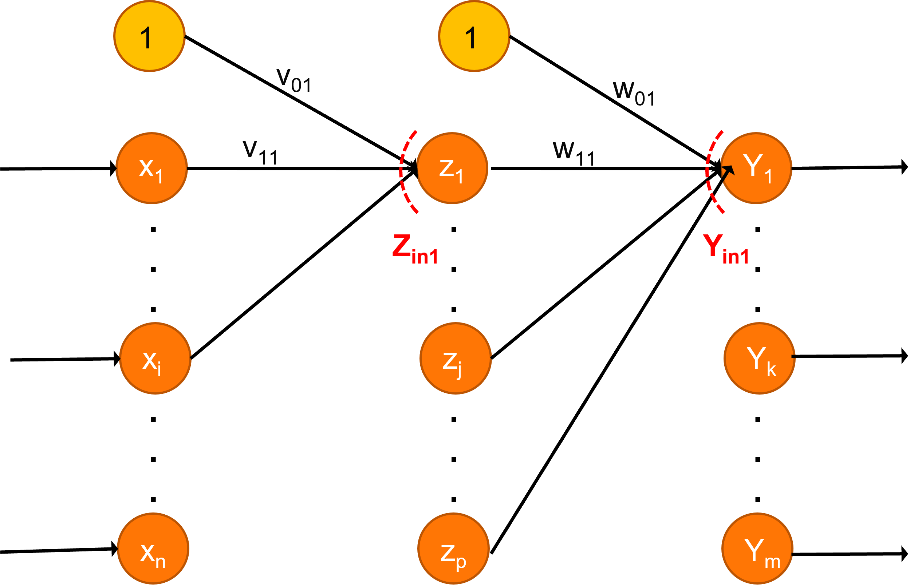
(2.6)

(2.7)

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.[[10]](#footnote-10) 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* (setiap node berhubungan dengan semua node sebelumnya dan setelahnya) dan dalam beberapa kasus setiap layer memiliki jumlah node yang sama.



Gambar 2.8  
Visualisasi Arsitektur MLP dengan Backpropagation

MLP adalah hasil perkembangan[[11]](#footnote-11) dari algoritma yang dibuat Rosenblatt yaitu *Perceptron*.[[12]](#footnote-12) 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[[13]](#footnote-13) 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.

(2.8)

(2.9)

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 diambil dari nilai–nilai sebelumnya yang telah dihitung atau ditemukan.

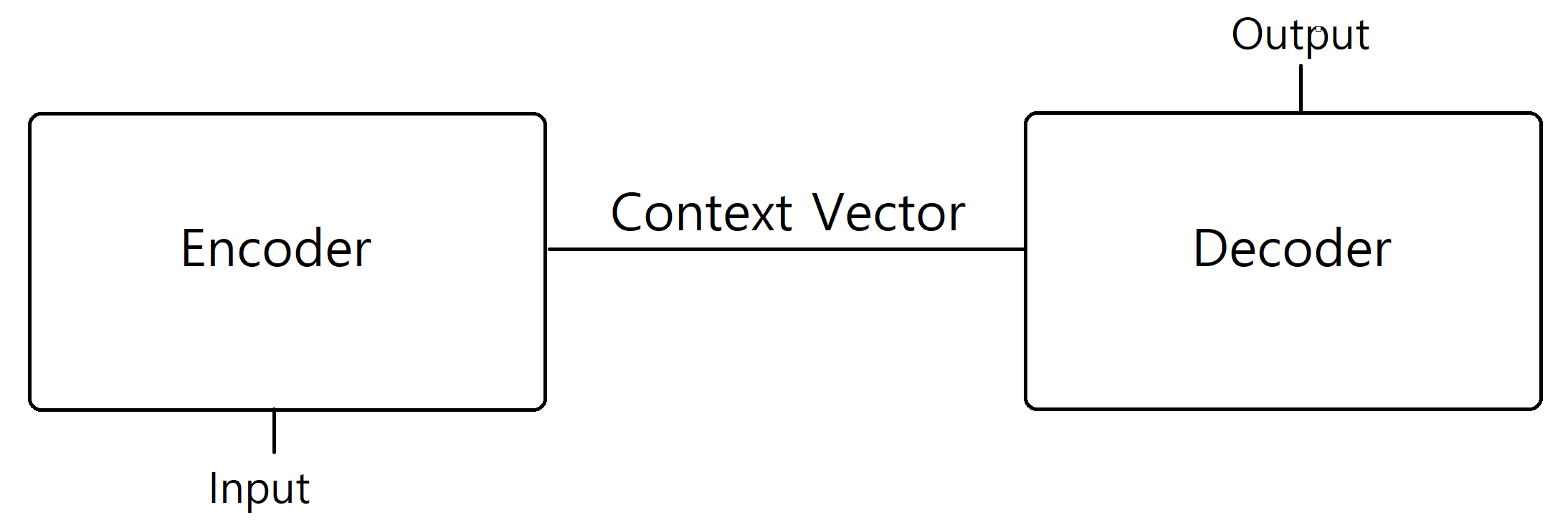
(2.10)

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[[14]](#footnote-14) 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.[[15]](#footnote-15) 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 2.9  
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

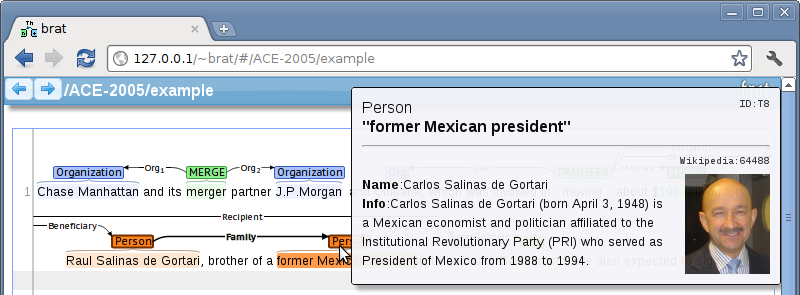
## BRAT

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 2.10  
Contoh Anotasi Text-Span pada BRAT

Adapun fitur *normalization annotations*, 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 2.11  
Contoh Normalization Annotation[[16]](#footnote-16)

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

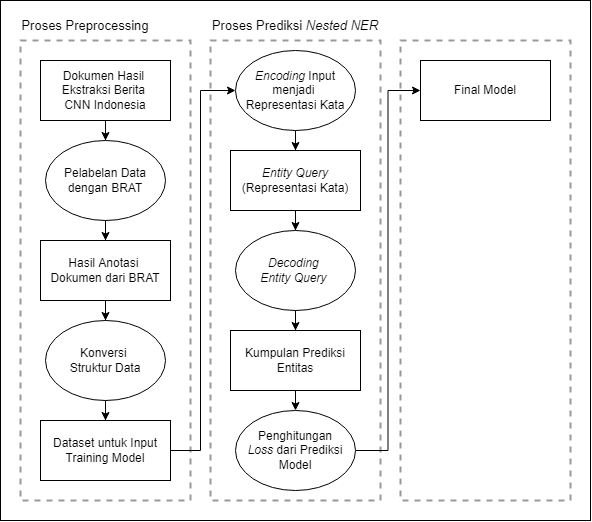
# ARSITEKTUR SISTEM

Bab ini akan membahas tentang arsitektur sistem keseluruhan tugas akhir ini. Arsitektur yang akan dibahas antara lain arsitektur sistem secara utuh, kemudian dibagi menjadi sub-proses tertentu. Pembagian sub-proses ini untuk membantu pengertian dan penjelasan lebih mudah dan detail kepada struktur arsitektur sistem. Pembahasan secara detail untuk masing-masing langkah / proses yang dilakukan dalam arsitektur pada sistem akan dibahas pada bab-bab berikutnya.

## Arsitektur Sistem Umum

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 umum akan membahas aliran data dari sub-proses pertama sampai akhir untuk mengetahui gambaran secara umum sistem ini. Visualisasi arsitektur sistem dapat dilihat dari gambar 3.1 dan dapat dilihat dari gambar tersebut terjadi kurang lebih \_\_\_ 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.

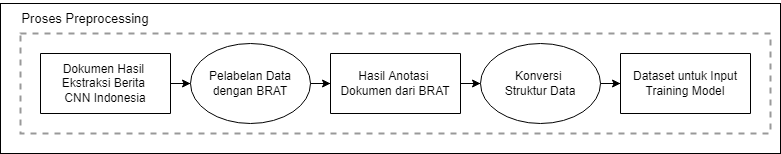
Gambar 3.1 memiliki pembagian tiga sub-proses, pra proses, proses prediksi nested NER, dan pasca proses. Pra proses adalah proses paling pertama, dan memiliki dua proses yang berhubungan dengan dataset. Proses ini fokus pada memroseskan dataset menjadi input yang sesuai untuk model program. Dari dataset 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 3.1  
Arsitektur Sistem

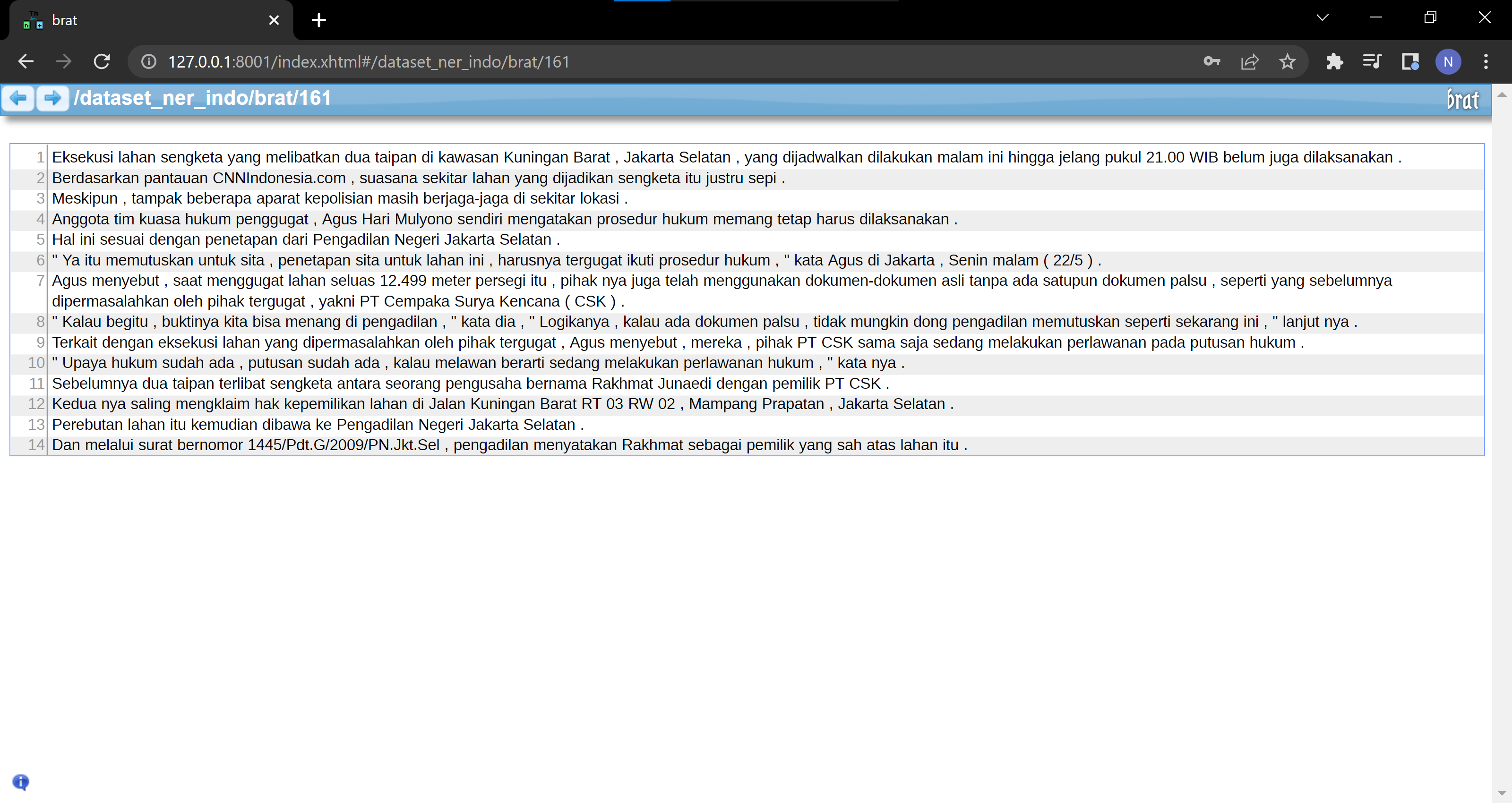
## Arsitektur Sistem dari Pra Proses

Subbab ini 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 3.2. 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.



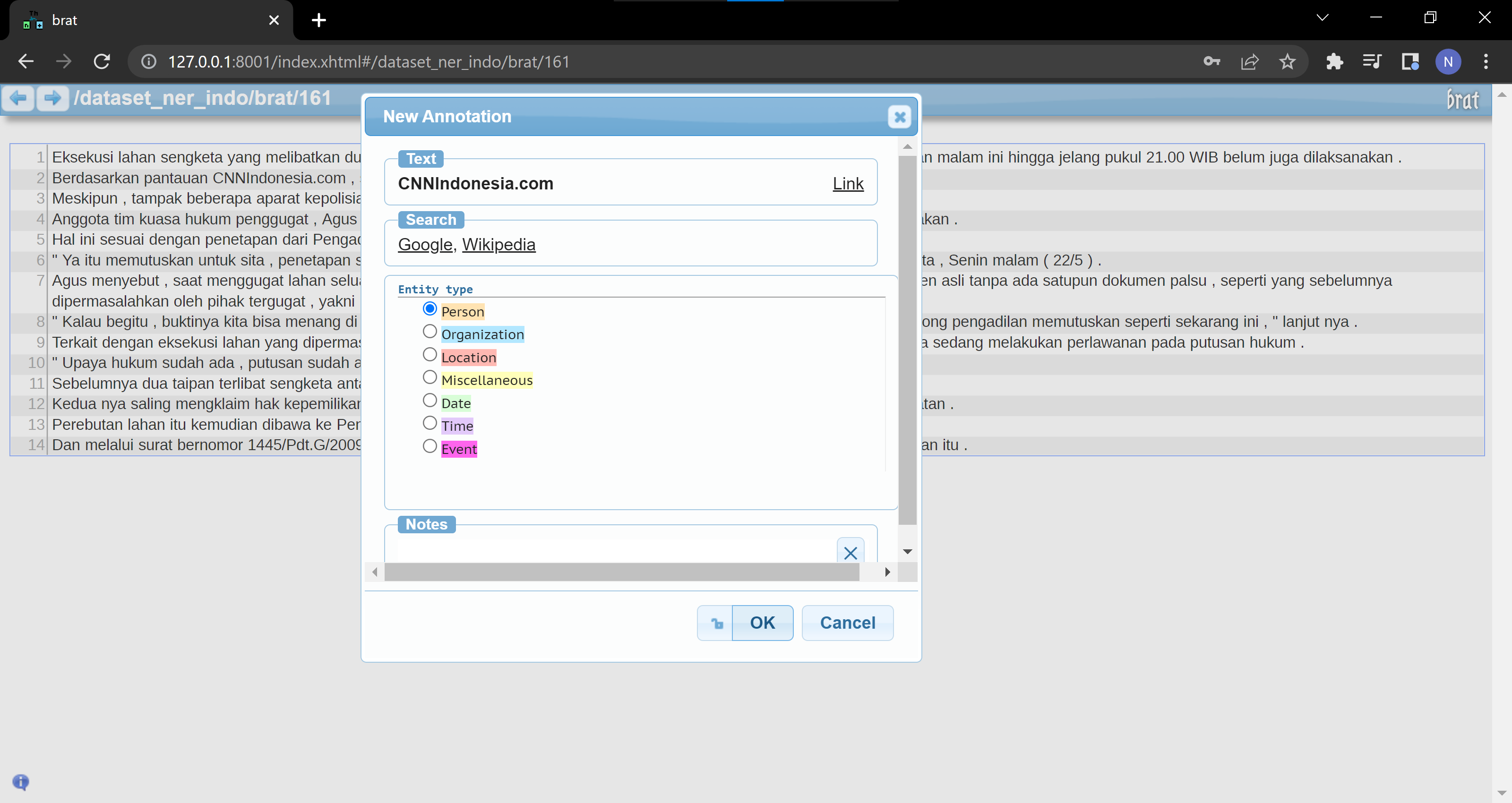
Gambar 3.2  
Arsitektur Sistem Pra Proses

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 3.3.



Gambar 3.3  
Tampilan File Teks pada BRAT

BRAT dipilih sebagai alat anotasi karena kemudahan kegunaannya dalam pelabelan. Karena penggunaan fitur yag 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 3.4.

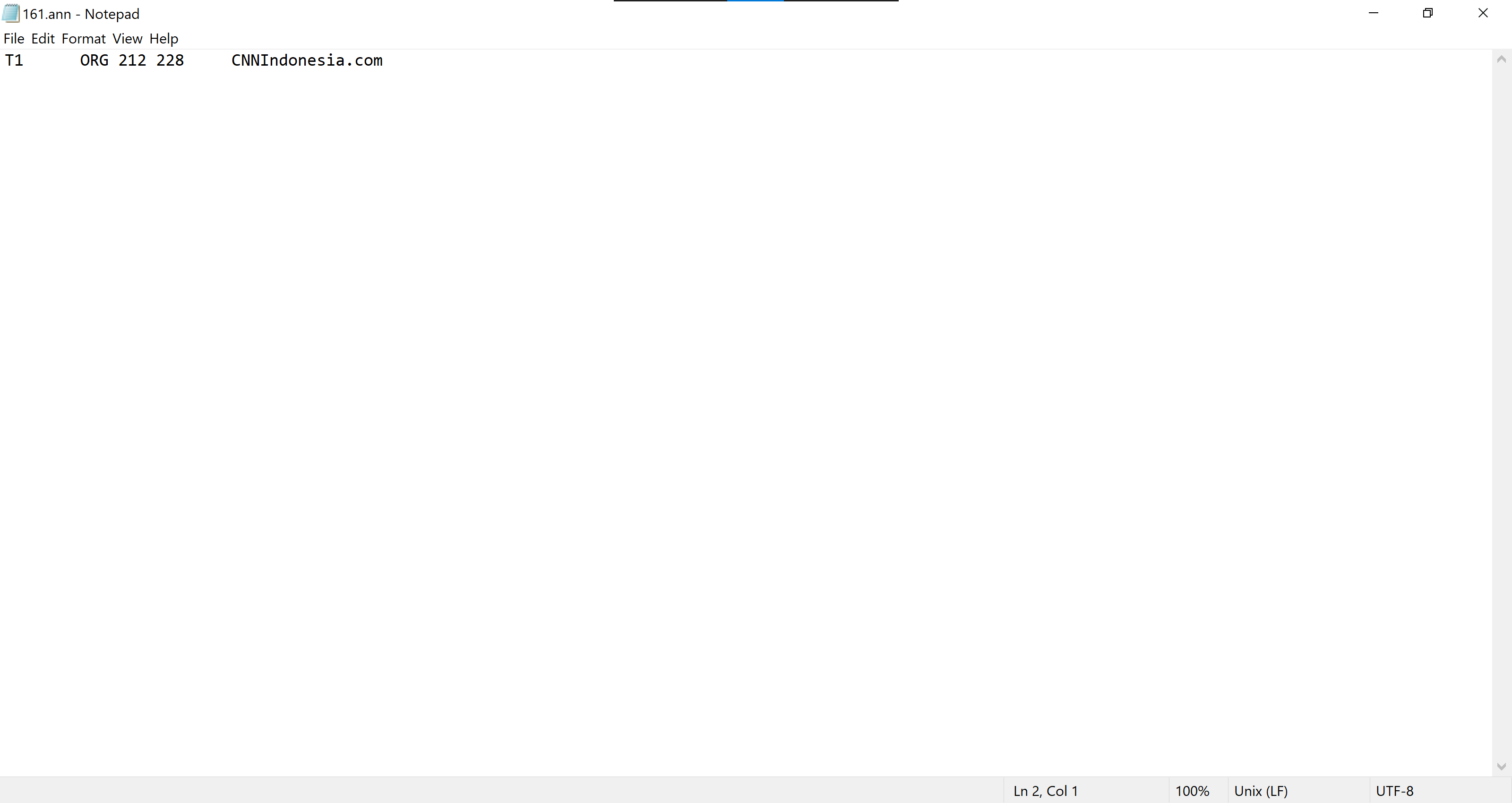


Gambar 3.4  
Tampilan Window Fitur Pelabelan

Contoh pada gambar 3.3 adalah satu file teks/dokumen dari dataset penelitian. Seiap 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 automatis 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 3.5.

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 3.5 (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)

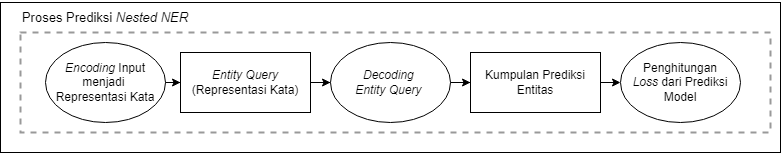
Gambar 3.5  
Struktur Data (a) Data Anotasi BRAT (b) Data Input Model

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.

## Arsitektur Sistem dari Proses Prediksi Nested NER

Subbab ini menjelaskan proses prediksi nested NER dengan melakukan training dengan metode Sequence-to-Set Network. Gambar 3.6 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 3.6  
Arsitektur Sistem Proses Prediksi Nested NER

Representasi kata tersebut merupakan output dan juga input pertama dari sub-proses ini, dimana representasi kata itu memiliki nama atau sebutan *Entity Query/Queries*. Entity Queries ini, seperti yang dijelaskan sebelumnya, merupakan representasi kata-kata dari input dataset dalam bentuk angka agar lebih mudah diproses oleh model. Secara teori, isi dari entity queries 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 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 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 3.6. 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.

## Arsitektur Sistem dari Proses Postprocess

Pada subbab ini akan dibahas mengenai tahap post processing yang akan dilakukan pada penelitian ini. Pada tahap postprocessing, akan dilakukan editing dengan menggunakan editor berbasis visualisasi. Detail arsitektur sistem dari tahapan post processing ini dapat dilihat pada gambar 3.19.

Pada postprocessing dengan editor ini akan digunakan editor yang\_ telah dikembangkan sebelumnya oleh Jessica Felani Wijoyo dalam Tugas Akhinya. Tujuan dari dilakukan editing ini adalah untuk menunjang agar hasil dari penelitian ini dapat dihasilkan sebuah akurasi yang sesuai dan tidak terjadi kekeliruan. Tampilan dari editor yang akan digunakan dapat dilihat pada gambar 3.20

Dengan menggunakan editor tersebut, data hasil induksi yang ada dapat diedit sebab hasil yang diperoleh merupakan hasil yang diekstrak secara otomatis sehingga mungkin saja terjadi kesalahan pada proses ekstraksi relasi yang ada. Untuk detail penggunaan dari tools tidak akan dijabarkan di dalam buku ini sebab telah dijabarkan pada buku tugas akhir dari Jessica Felani Wijoyo. Contoh pra proses yang dilakukan dapat dilihat pada gambar 3.2 1 dan 3.22.

Pada gambar 3.21, hasil relasi yang ada nampak bahwa dalam synset dengan nomer 69834 terdapat 3 buah kata yang salah, dimana salah satu katanya dalah salah. Dari informasi sense yang ada didalam synset tersebut, I yang tertera merupakan nountop yang sebenarnya salah sebab synset tersebut nya berada di dalam area lexnames hewan. Informasi gloss da tersebut ternyata kosong dan harus disi. Dari beberapa informasi yang salah t, maka tahap postprocessing digunakan untok memperbaiki bebe agar informasi yang akan diubah menjadi lexical database adalah benar.

Dari gambar 3.22 tersebut nampak bahwa isi dari sinonim set yang salah tersebut dibuang sehingga informasi yang berada di dalam synset dengan nomer 69384 adalah benar. Sedangkan untuk lexnames tersebut, akan diubah menjadi noun.animal sehingga synset ini nantinya akan ditempatkan dalam domain noun.animal pada lexicographer file. Untuk gloss yang kosong agar ti lak terjadi error dalam pembuatan lexical database maka akan disi dengan kata gloss untuk menggantikan isi dari gloss tersebut. Hal lain yang akan dilakukan dalam tahapan adalah selain melakukan beberapa hal diatas juga akan dilakukan proses lain seperti pembuangan karakter selain alphanumerik dan tanda baca tertentu antara lain, tanda titik dan tanda tanya. Beberapa relasi kembar karena kesalahan input pada saat melakukan induksi juga akan dibuang agar didapatkan relasi yang unik saja. Dari tahapan postprocessing ini lah data yang valid akan didapatkan untuk dibentuk menjadi lexical database WordNet Bahasa Indonesia.

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. 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-4)
5. 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-5)
6. Plato, *“Cara Menggunakan Pengenalan Entitas Bernama (NER) Untuk Ekstraksi Informasi”*, (https://zephyrnet.com/id/cara-menggunakan-nama-pengenalan-entitas-untuk-ekstraksi-informasi/) [↑](#footnote-ref-6)
7. Ashish Vaswani, dkk, Attention Is All You Need, 2017. [↑](#footnote-ref-7)
8. Michael Phi, “*Illustrated Guide to Transformers- Step by Step Explanation*”, https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0 [↑](#footnote-ref-8)
9. Jay Alammar, “*The Illustrated Transformer*”, <http://jalammar.github.io/illustrated-transformer/> diakses, 4 Mei 2022, 12:32 [↑](#footnote-ref-9)
10. Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117 [↑](#footnote-ref-10)
11. Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press [↑](#footnote-ref-11)
12. Frank Rosenblatt. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory 85, 460–461 (1957) [↑](#footnote-ref-12)
13. Prof. Dr. Ir. Kuswara Setiawan, M.T., Buku Paradigma Sistem Cerdas, (Malang : Bayu Media, 2003) [↑](#footnote-ref-13)
14. 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-14)
15. Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to Sequence Learning with Neural Networks, 2014 [↑](#footnote-ref-15)
16. BRAT, mini-introduction to brat, https://brat.nlplab.org/introduction.html [↑](#footnote-ref-16)