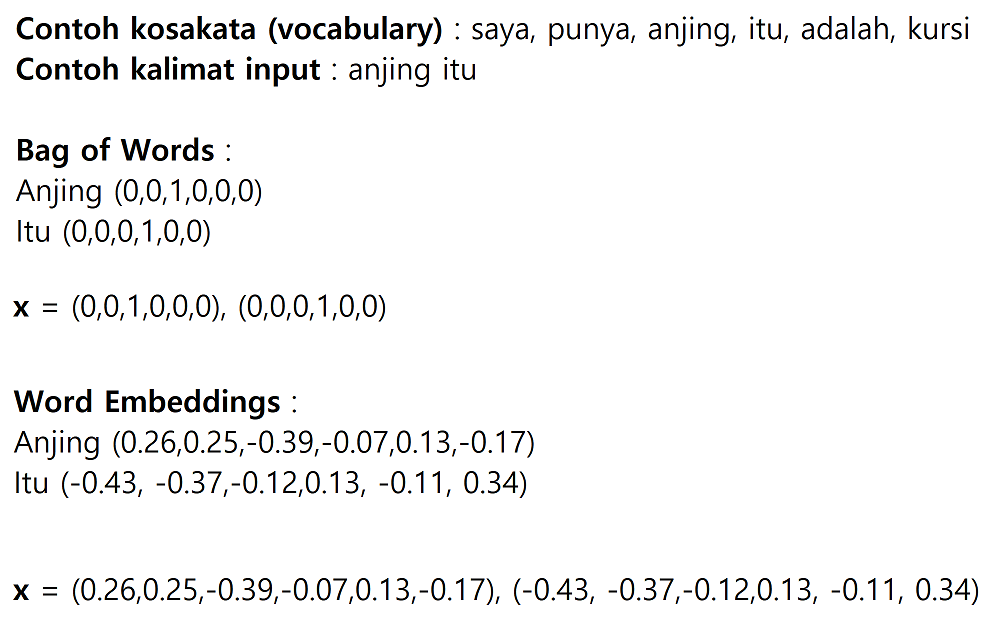
BAB II

# 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.

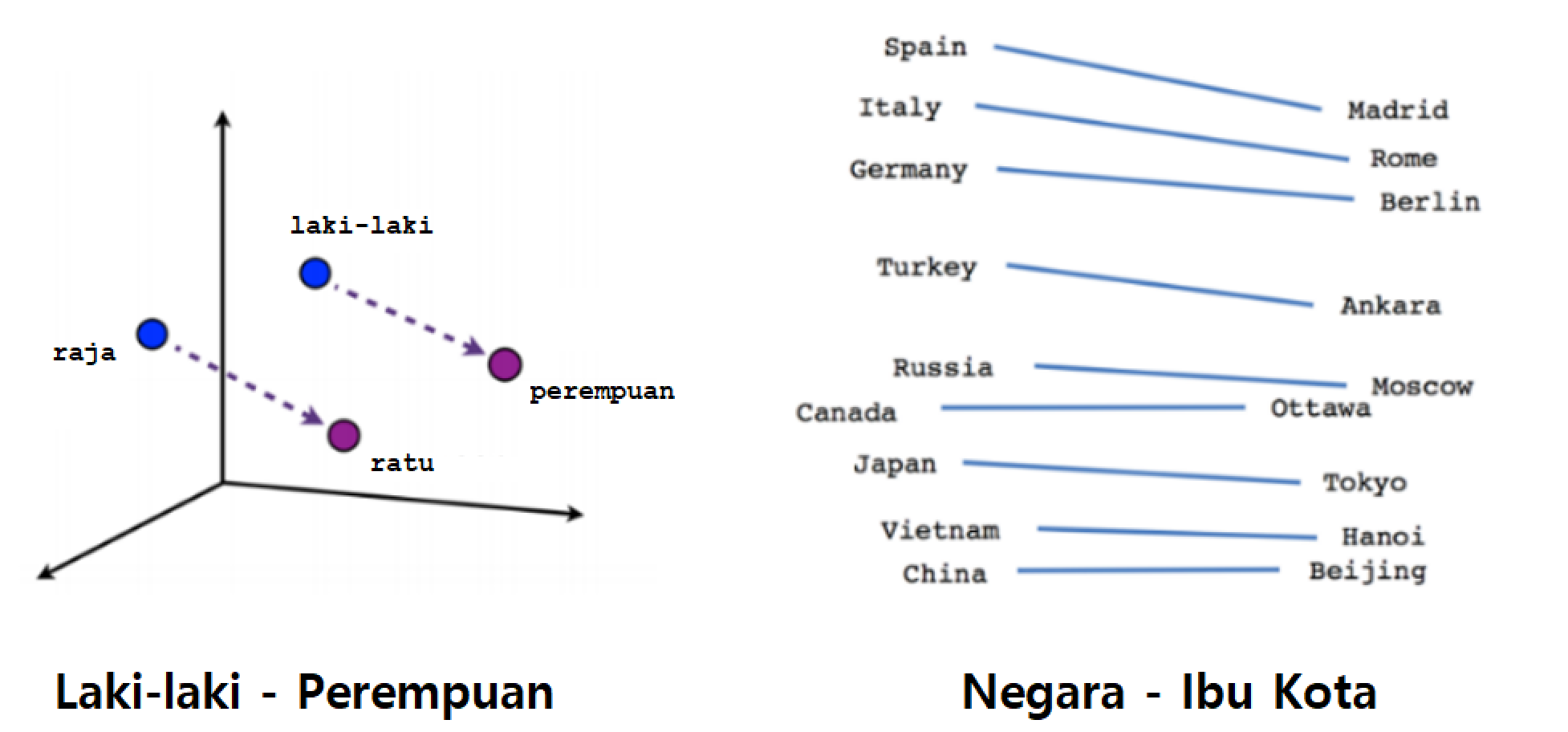
## Word Embedding

Word embedding adalah teknik yang saat ini menjadi teknik yang sering digunakan dalam NLP karena kemampuannya untuk menyimpan nilai semantik dari satu kata dengan kata lainnya/kata di sekitarnya.[[1]](#footnote-1) Word embedding membantu komputer untuk menerima kata-kata/huruf menjadi angka agar dapat diolah dengan mudah. Hasil dari word embedding adalah sekumpulan vektor yang berisikan angka riil. Word embedding menjadi teknik yang sering digunakan dalam penelitian NLP saat ini 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 Bag of Words dan 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 “anjing itu”. Jika direpresentasikan tiap kata dengan satu dimensi dan memiliki panjang sebanyak kata yang ada dalam satu kosakata dataset tersebut (sering kali berjumlah banyak), ukuran untuk tiap representasi kata secara keseluruhan akan menjadi besar dan memakan nilai komputasi yang besar. Jika menggunakan representasi yang umum seperti *One Hot Encoding*, maka dimensi yang besar akan memiliki nilai nol yang banyak. Sebuah vektor yang mayoritas adalah bernilai nol adalah representasi sparse. Sedangkan word embedding akan memiliki panjang dimensi yang tidak bergantung dengan jumlah kata, karena itu tentu akan lebih efisiensi secara komputasi, memori dan informasi karena nilai nol tidak sering digunakan.

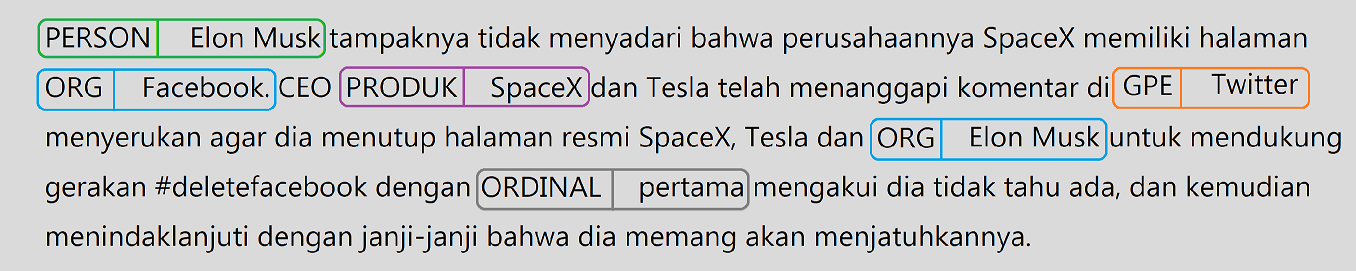


Gambar 2.2   
Contoh Nilai Semantik Tersimpan dari Word Embedding

Isi tiap nilai dalam sebuah dimensi dari representasi dense dari word embedding memiliki arti dan informasi yang berbeda-beda. Salah satu yang menjadi contoh 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 contohnya “laki-laki” dan “perempuan”. Contoh lain adalah hubungan secara semantik seperti nama negara dengan nama ibu kotanya.

## Named Entity Recognition (NER)

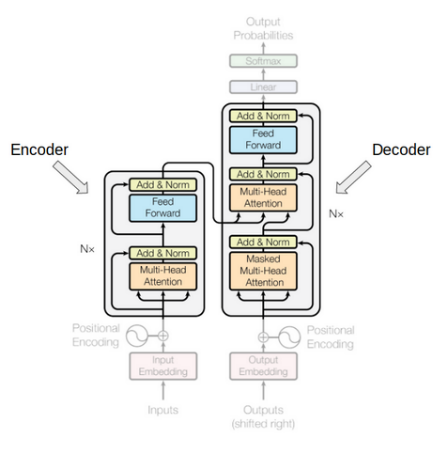
Named Entity Recognition (NER) merupakan task NLP untuk mencari entitas bernama dalam suatu kalimat kemudian menentukan jenis entitas tersebut berdasarkan kategori entitas yang ada dari sebuah dokumen/kalimat.[[2]](#footnote-2) Sebuah entitas bernama adalah istilah untuk menyebut sebuah entitas yang 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. Contoh hasil dari task NER dapat dilihat dari gambar di bawah (gambar 2.3).



Gambar 2.3   
Contoh Pengenalan Entitas Bernama[[3]](#footnote-3)

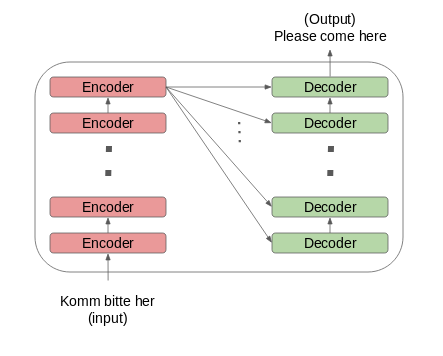
## Transformer

Konsep adanya Transformer (Gambar 2.4) muncul dari model *sequence-to-sequence* (*seq2seq*), di mana model tersebut memiliki tujuan untuk menerima sebuah sequence (urutan) dan memberikan hasil output sebuah sequence juga. 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 yang menggunakan urutan selaras RNN, Transformer bergantung pada penggunaan *attention* untuk menghitung representasi input dan output nya.



Gambar 2.4  
Arsitektur Transformers[[4]](#footnote-4)

Arsitektur dari Transformer berorientasi hanya pada encoder dan decoder, encoder dan decoder tersebut dapat diatur jumlahnya. Isi dari bagian encoder dan decoder Transformer adalah encoder dan decoder yang berjumlah banyak (yang merupakan parameter yang bisa diatur, jumlah yang ditetapkan pada paper “Attention is All You Need” adalah enam) dan bertumpukan (dapat dilihat pada Gambar 2.5). Secara detail, satu bagian Encoder memiliki Multi-Head Attention kemudian diikuti Feed Forward Neural Network. Bagian dari satu 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.



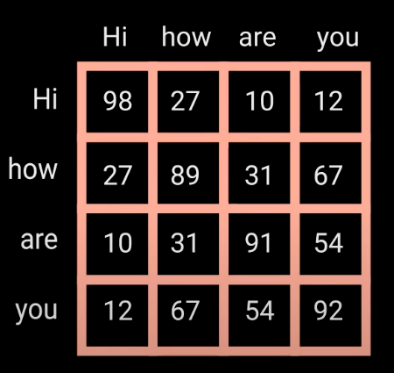
Gambar 2.5  
Representasi Transformer bagian Encoder Decoder

### 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-head 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). [[5]](#footnote-5) 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 sistem pengambilan data. Contoh kasus pada website *Youtube*, jika ingin mencari suatu video user akan memasukkan sebuah query dan sistem 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 2.6  
Contoh Isi dari Matriks Score

Selain ketiga vektor itu, ada matriks *score* yang kegunaannya adalah untuk memberi skor/nilai terhadap tiap kata. Skor tersebut menandakan seberapa besar fokus pada suatu kata dengan kata-kata disekitar kata yang difokuskan tersebut. Semakin tinggi nilainya, semakin besar fokus yang diberikan. Dan matriks score didapatkan dari perkalian *dot product* antara vektor query dan vektor key.

Matriks score yang dibiarkan dengan nilai yang relatif tinggi akan menyebabkan permasalahan seperti *exploding gradients* (permasalahan saat training model, di mana 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 diperlukan penurunan nilainya dengan dibagi dengan akar pangkat panjang dimensi vektor key. Nilai matriks score yang baru 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. Tujuan fungsi aktivasi adalah memuncakkan nilai yang tinggi dan merendahkan nilai yang kecil, dengan representasi tiap nilai antara 0 dan 1. Hasil akhir dari softmax akan dipanggil sebagai *attention weights*. Attention weights tersebut akan dikali 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 satu vektor 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. Dapat disimpulkan multi-head attention adalah self-attention yang dilakukan sebanyak N kali (nilai N adalah parameter yang dapat ditentukan sendiri), multi-head dapat dilihat pada rumus 2.3. Dari penjelasan sebelumnya langkah-langkah self-attention didapatkan rumus sebagai di bawah ini.

(2.1)

(2.2)

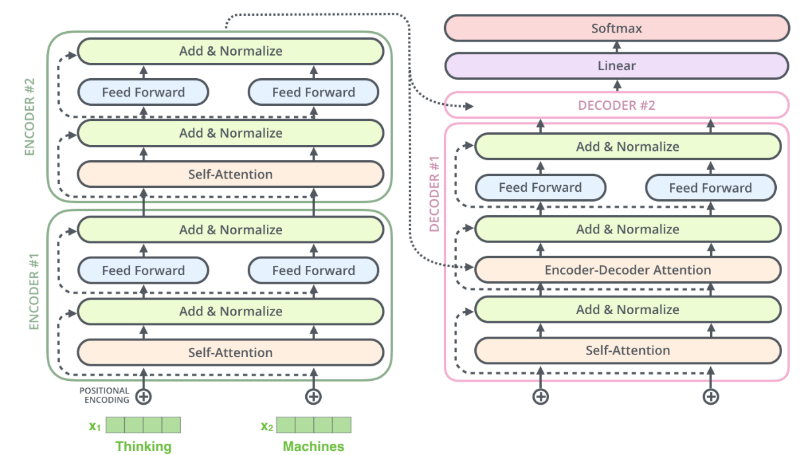
(2.3)

Sebelum penjelasan bagian FFN, perlu diketahui adanya sublayer setelah multi-head attention dan setelah FFN yaitu sublayer *residual connection* dan normalisasi. Residual connection adalah proses menambahkan *positional encoding* dengan input. 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, rumus untuk indeks yang ganjil akan dibuatkan vektor 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 multi-head attention disambungkan dengan layer berisi residual connection dan normalisasi, dan selanjutnya adalah layer FFN. Input dari FFN didapatkan dari residual connection dan LayerNorm. Struktur FFN ini adalah dua layer linear dengan fungsi pengaktifan ReLU diantarnya. Setelah FFN, output dari layer tersebut dilewatkan proses residual connection dan LayerNorm sekali lagi.



Gambar 2.7  
Representasi Tiap Layer Dari Arsitektur Transformer[[6]](#footnote-6)

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 bantuan penghitungan attention, residual connection membantu training model agar gradien dapat melewati network dengan tanpa memberikan efek samping seperti exploding gradient. LayerNorm digunakan untuk menormalisasikan nila-nilai agar network menghitung nilai-nilainya dapat bernilai stabil, dan FFN melakukan proyeksi/analisa terhadap input nya sehingga berpotensi untuk memberikan representasi dengan informasi yang banyak. Gambar 2.7 adalah penggambaran isi layer encoder dan decoder secara detail.

### Bagian Decoder

Struktur decoder dengan encoder memiliki kemiripan namun diantara multi-head attention dengan FFN terdapat tambahan satu multi-head attention tetapi dengan mekanisme *masking* ditambahkan, sublayer residual connection dan normalisasi juga tetap diberikan seperti sebelumnya. Dan diakhir decoder bagian FFN dan sublayer setelahnya, diberikan layer linear sebagai *classifier* dan fungsi aktivasi jenis softmax untuk mendapatkan probabilitas kata yang diprediksikan. Dalam Transformer, bagian decoder ini bersifat *autoregressive*, artinya decoder tersebut menggunakan metode 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 kata akan diubah menjadi word embedding, yang akan ditambahkan dengan positional embedding. Embedding tersebut akan dimasukkan ke layer multi-head pertama dan dilakukan pembuatan vektor query, key dan value juga matriks score. Kemudian matriks score dilewatkan fungsi pengaktifan softmax, hasil tersebut akan dikalikan dengan matriks value dan menghasilkan output terakhir multi-head attention yang pertama.

Sebenarnya dalam paper “Attention is All You Need”, 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 decoder nya, sehingga membutuhkan solusi untuk menghindari informasi yang bocor (*leakage*) sehingga dapat dilihat decoder untuk membantu dirinya sendiri dalam prediksi. Karena paper yang dirujuk pada tugas akhir ini membuat decoder dengan sifat non-autoregressive decoder, maka kebocoran informasi tidak dipermasalahkan, sehingga decoder Sequence-to-Set dilakukan tanpa masking decoder sehingga bisa mendapatkan nilai ketergantungan antar entity nya.

Setelah multi-head yang pertama, 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 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 (seperti pada gambar 2.7). Dan vektor valuenya diambil dari output multi-head attention decoder yang pertama. Dengan cara ini, cross-attention dapat 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.

Tahap terakhir adalah layer linear dan layer softmax di akhir decoder. Kedua layer ini ditempatkan setelah seluruh tumpukan decoder telah dilewatkan. Layer linear tersebut berperan sebagai *classifier* sebesar *class* yang ditentukan dalam penelitian tersebut. Contohnya dalam kasus jika ada 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 total dari output ini jika dijumlahkan hasilnya sama dengan 1. Nilai probabilitas tertinggi adalah kata yang telah diprediksikan. 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.

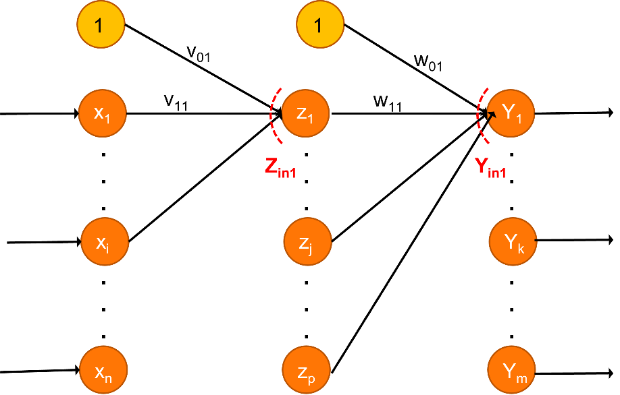
(2.6)

(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*. Penjelasan dari ketiga layer tersebut dapat dilihat pada subbab berikut mengenai Multi-Layer Perceptron (MLP).

### Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) memiliki struktur yang mirip dengan struktur *Feed Forward Neural Network* (FFN). FFN adalah bentuk neural network yang paling pertama dan paling sederhana dibandingkan dengan NN yang sudah berkembang saat ini.[[7]](#footnote-7) 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*). Alur dimulai dengan 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 2.8  
Visualisasi Arsitektur MLP dengan Backpropagation

MLP adalah hasil perkembangan[[8]](#footnote-8) dari algoritma yang dibuat Rosenblatt yaitu *Perceptron*.[[9]](#footnote-9) 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[[10]](#footnote-10) 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. Loss Calculate akan melakukan penghitungan jauhnya perbedaan antara nilai output saat ini dengan output sebenarnya. Ada banyak jenis 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 fungsi loss yang didapatkan. Nilai gradien adalah penghitungan nilai untuk mengevaluasikan nilai fungsi loss, 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 fungsi loss 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.

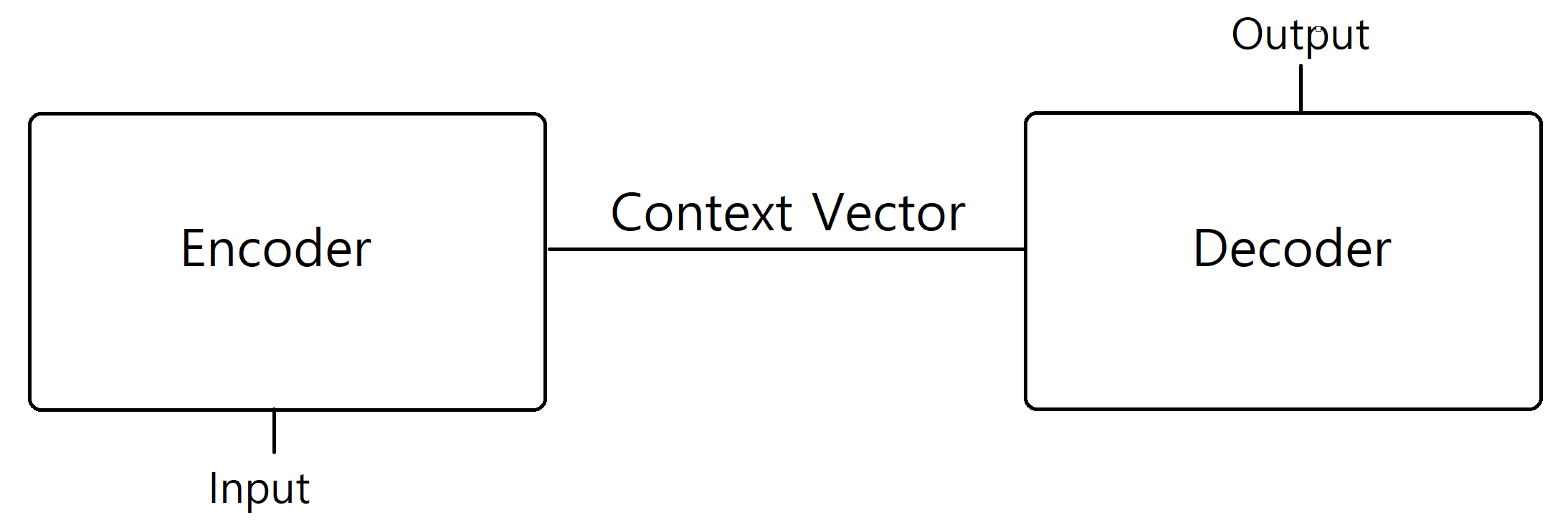
(2.10)

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. Contohnya adalah penggunaan MLP pada bidang pengenalan suara, pengenalan gambar, dan perangkat lunak terjemahan mesin[[11]](#footnote-11) meskipun pada akhirnya *support vector machines* (SVM) pada akhirnya lebih direkomendaiskan untuk digunakan sebagai classifier.

## Sequence to Sequence Models

Subab ini akan menjelaskan mengenai seq2seq, suatu metode yang menjadi inspirasi untuk metode sequence-to-set dari 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) dengan kemampuannya untuk mengingat informasi beberapa timestep/kata/huruf sebelumnya.

Metode seq2seq dikenalkan oleh Google dan kemudian metode tersebut sering digunakan dalam kasus translasi bahasa, contoh dari salah satu penelitian dari Google sendiri mengenai penggunaan metode seq2seq untuk meningkatkan kualitas translasi bahasa.[[12]](#footnote-12) 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 sebuah 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 vektor 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.

## Hungarian Match

Hungarian match[[13]](#footnote-13) adalah algoritma yang akan membantu permasalahan penghitungan loss saat training model untuk Sequence-to-Setdengan kemampuannya untuk mencari *optimal* *assignment* (pencocokan/penyesuaian optimal) antara prediksi dengan golden entity. Kata *hungarian* dipilih 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 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 dan 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). Perlu diketahui istilah yang akan digunakan untuk 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. Lemma pertamamenyatakan 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.

|  |
| --- |
|  |

Adapun *corollary* (bukti yang mendukung) untuk lemma pertama 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 :

Teorema 1: 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 :

Teorema 2: 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 (satu buah) 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 :

Teorema 3: 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 :

Teorema 4 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.

(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:

Teorema 5: nilai 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.

(2.12)

(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:

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.

(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 .

(2.15)

Pernyataan corollary 1 dapat mendefinisikan perbaikan budget dapat dinyatakan:

(2.16)

(2.17)

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 apakah budget terbaru ini memadai/adequate dan total alokasi anggaran telah berkurang dari sebelumnya. Perubahan dianggap gagal apalah bila telah berkurang dan tidak berganti, dengan kegagalan 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.

(2.18)

Dapat dibuktikan juga bahwa jumlah alokasi anggaran telah berkurang karena *n* – *r* dan bertambah sebanyak *s*, dapat diringkas bahwa telah berkurang sebanyak *n* – (*r + s*) = *n – m > 0*. Semua ini menyimpulkan teorema 7 yaitu :

Teorema 7: 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

Algoritma Hungarian mendapatkan inspirasi dan menggabungkan dari dua permasalahan dan teorema Simple dan General Assignment Problem. Bagian dari subbab ini akan lebih fokus dalam menjelaskan bagaimana cara algoritma ini berjalan bukan pembahasan secara teorema. 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.

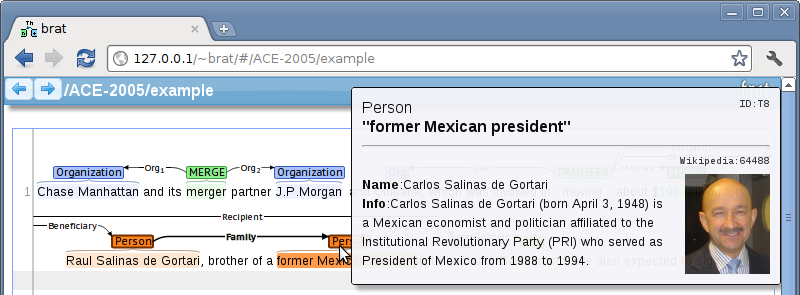
## BRAT

Subbab BRAT 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 adalah 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* (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 2.11  
Contoh Normalization Annotation[[14]](#footnote-14)

Banyak penelitian task dalam NLP yang dilakukan dengan bantuan BRAT, dan dalam web BRAT telah 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. 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, fitur yang digunakan secara umum dari BRAT juga banyak dan berguna dalam perannya sebagai alat anotasi. Tugas akhir ini membatas penggunaan fitur sesuai yang dibutuhkan yaitu anotasi text-span saja. Juga 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 BRAT 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.

## Macro dan Micro F1 Score

Macro F1 Score adalah penghitungan F1 Score yang 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”[[15]](#footnote-15). 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 2.1). Tabel tersebut akan menyediakan metrik precision (P), recall (R), dan F1 score untuk tiap label.

Tabel 2.1  
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 2.19, 2.20, dan 2.21. 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 2.21, 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.

(2.19)

(2.20)

(2.21)

Untuk Macro F1 Score, pada tabel 2.2 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 2.2  
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 |

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 2.2 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 disebut sebagai akurasi (*accuracy*) juga, karena hasil penghitungan rumus Micro F1 Score dengan hasil penghitungan rumus akurasi memiliki nilai yang sama.

Penggunaan Micro dan Macro perlu diperhatikan dengan cara evaluasi metode yang diinginkan, 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 tugas akhir ini, jenis F1 Score yang digunakan adalah Macro F1 Score dan Micro F1 Score. Namun untuk perbandingan F1 Score terbaik diambil dari Micro F1 Score.

1. 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-1)
2. 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-2)
3. Plato, *“Cara Menggunakan Pengenalan Entitas Bernama (NER) Untuk Ekstraksi Informasi”*, (https://zephyrnet.com/id/cara-menggunakan-nama-pengenalan-entitas-untuk-ekstraksi-informasi/, diakses pada 27 Maret 2022) [↑](#footnote-ref-3)
4. Ashish Vaswani, dkk, Attention Is All You Need, 2017. [↑](#footnote-ref-4)
5. Michael Phi, “*Illustrated Guide to Transformers- Step by Step Explanation*”, (https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0, diakses pada 1 Mei 2022) [↑](#footnote-ref-5)
6. Jay Alammar, “*The Illustrated Transformer*”, http://jalammar.github.io/illustrated-transformer/ diakses, 4 Mei 2022, 12:32 [↑](#footnote-ref-6)
7. Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117 [↑](#footnote-ref-7)
8. Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press [↑](#footnote-ref-8)
9. Frank Rosenblatt. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory 85, 460–461 (1957) [↑](#footnote-ref-9)
10. Prof. Dr. Ir. Kuswara Setiawan, M.T., Buku Paradigma Sistem Cerdas, (Malang : Bayu Media, 2003) [↑](#footnote-ref-10)
11. 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-11)
12. Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to Sequence Learning with Neural Networks, 2014 [↑](#footnote-ref-12)
13. H. W. Kuhn, The Hungarian Method for The Assignment Problem [↑](#footnote-ref-13)
14. BRAT, mini-introduction to brat, (https://brat.nlplab.org/introduction.html, 27 April 2022) [↑](#footnote-ref-14)
15. 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-15)