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.

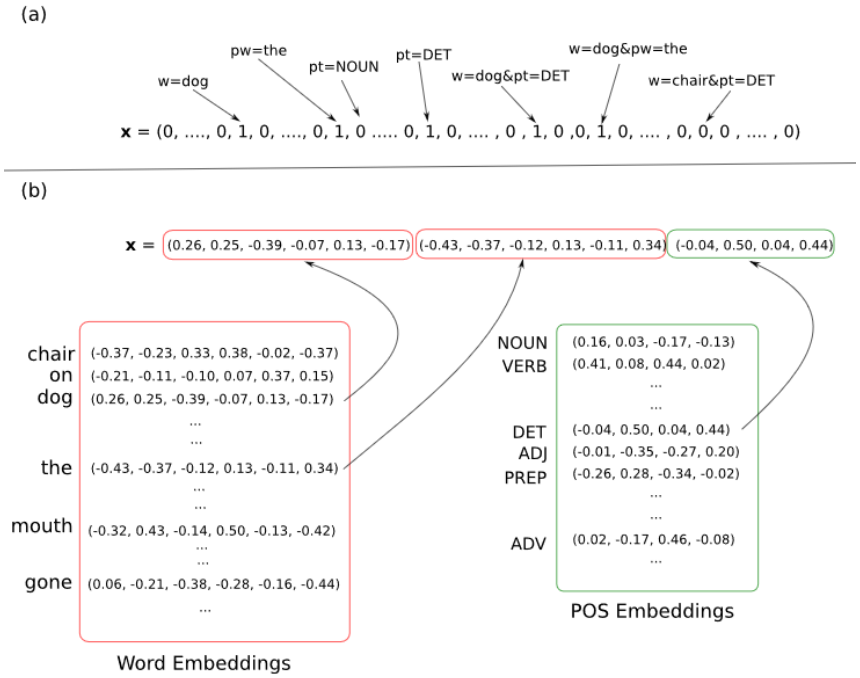
2.1 Word Embedding

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. [1]

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

[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.

menghasilkan representasi kata *sparse* (elemen yang sebagian besar mengandung nilai nol dan sedikit elemen *non-zero*).



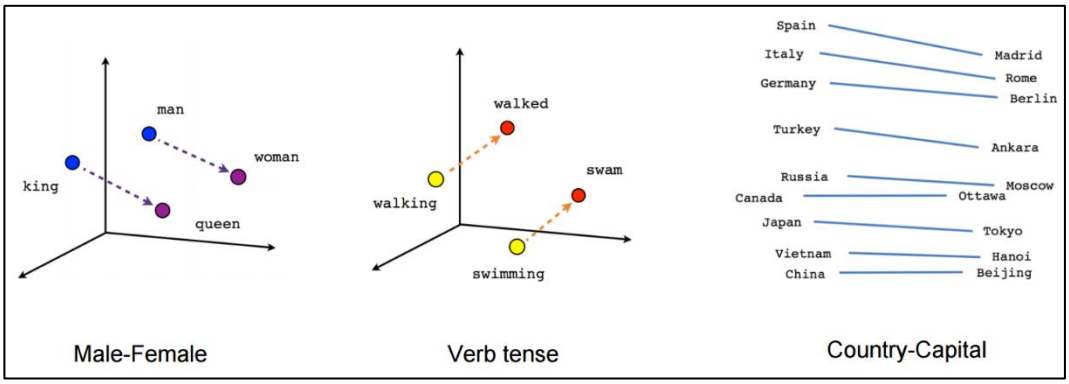
Gambar 2.1
Contoh Visualisasi (a) Bag of Words dan (b) Word Embedding[2]

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

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

kata lainnya. Dari Gambar 2.2 menggambarkan visualisasi nilai semantik yang dapat direpresentasikan. Relasi antar kata yang bisa diberikan seperti lawan kata (Male-Female), kalau dalam bahasa Inggris mampu memberikan nilai relasi bentuk kata kerja yang berbeda (Verb Tense), bahkan juga hubungan secara semantik seperti nama negara dengan nama ibu kotanya (Country-Capital).



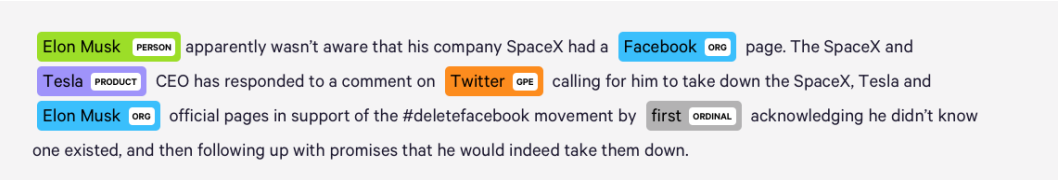
Gambar 2.2
Contoh Nilai Semantik Tersimpan dari Word Embedding

2.2 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.[3] 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

[3] 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

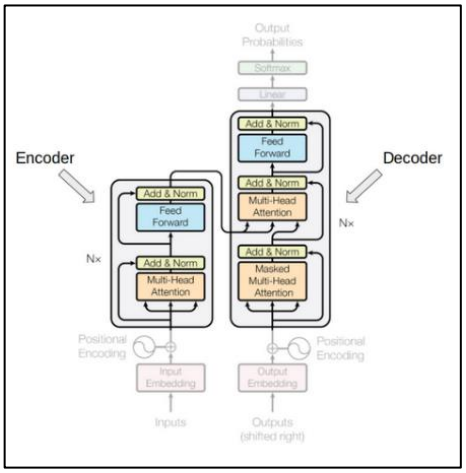
untuk keperluan analisis lebih lanjut. orang) adalah Valencia. Penggambaran hasil dari task NER dapat dilihat dari gambar dibawah (Gambar 2.3).



Gambar 2.3
Contoh Pengenalan Entitas Bernama[4]

2.3 Transformer

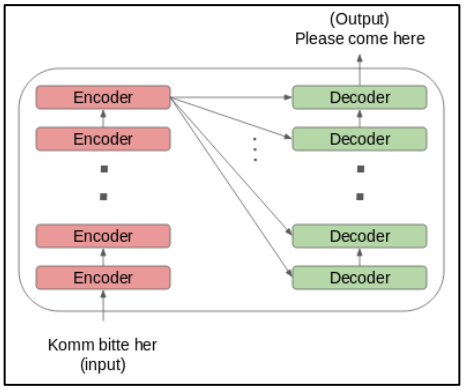
Konsep adanya Transformer (Gambar 2.4) muncul dari model *sequence-to-sequence* (*seq2seq*), di mana model tersebut memiliki tujuan untuk mengerti konversi sebuah sequence, contohnya adalah penerjemahan bahasa dari Inggris ke Indonesia. Model ini melakukan tugasnya dengan baik tetapi ada kesulitan dalam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



Gambar 2.4
Arsitektur Transformers[5]

[4] Plato, *"Cara Menggunakan Pengenalan Entitas Bernama (NER) Untuk Ekstraksi Informasi"*, (https://zephyrnet.com/id/cara-menggunakan-nama-pengenalan-entitas-untuk-ekstraksi-informasi/)
[5] Ashish Vaswani, dkk, Attention Is All You Need, 2017.

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

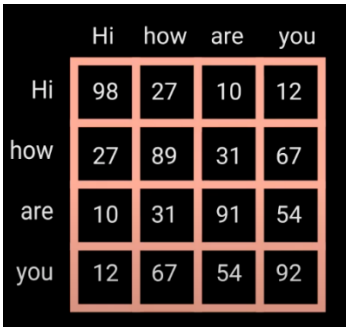
2.3.1 Bagian Encoder

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). [6] 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 ($\sqrt{d_k}$)
4. Matriks score dilewatkan pada fungsi aktivasi *softmax*
5. Mengalikan matriks score dengan vektor value
6. Menjumlahkan seluruh matriks yang dihitung

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



Gambar 2.6
Contoh Isi dari Matriks Score

[6] Michael Phi, "*Illustrated Guide to Transformers- Step by Step Explanation*", https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

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

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

Agar dapat diimplementasikan kepada multi-head attention, tiap operasi self-attention ini (yang disebut juga sebagai *head*) akan terjadi sebanyak N kali dan untuk tiap head mendapatkan vektor query, key dan value yang telah dibagi sebanyak N vektor dan memiliki rumus yang ditulis pada rumus $Head(Q, K, V) = softmax\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V$ (2.2. Output untuk setiap head akan disambung menjadi 1 vector yang sama. Secara teori, dengan tiap head/self-attention melakukan perhitungan yang berbeda, maka informasi yang didapatkan untuk

melakukan *decoding* akan menjadi lebih banyak/besar. Dan dapat disimpulkan multi-head attention adalah self-attention yang dilakukan sebanyak N kali (nilai N adalah parameter yang dapat ditentukan sendiri), multi-head memiliki rumus yang mudah dimengerti dapat dilihat pada rumus $Multi(Q, K, V) = Concat(Head_1, \dots Head_N)\, W^O$ (2.3. Dari penjelasan sebelumnya langkah-langkah self-attention didapatkan rumus sebagai berikut:

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_i)} \qquad (2.1)$$

$$Head(Q, K, V) = softmax\left(\frac{Q\, K^T}{\sqrt{d_k}}\right) V \qquad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (2.2)$$

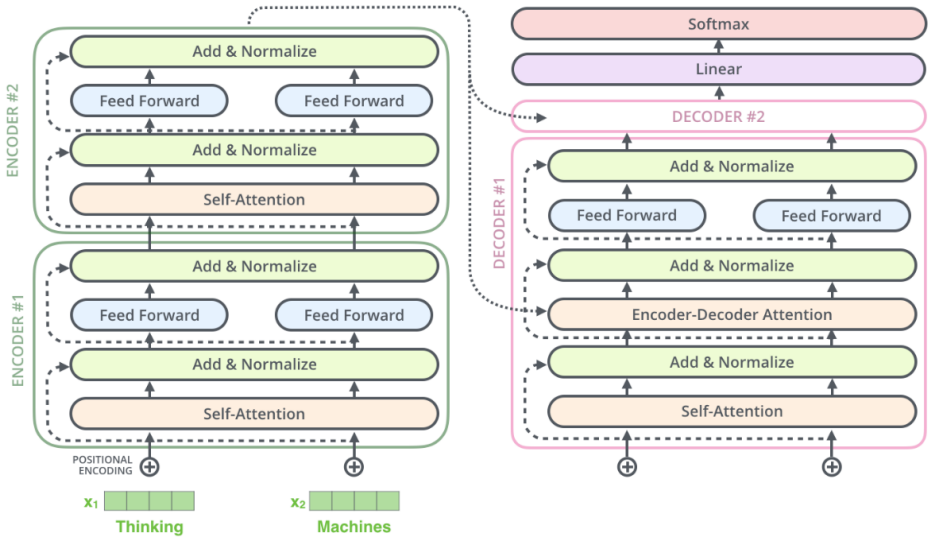
$$Multi(Q, K, V) = Concat(Head_1, \dots Head_N)\, W^O \qquad \dots\dots\dots\dots \quad (2.3)$$

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 $PE(pos, 2i) = sin\left(pos/10000^{2i/d_{model}}\right)$ (2.4 dan $PE(pos, 2i + 1) = \cos\left(pos/10000^{2i/d_{model}}\right)$ (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.

$$PE(pos, 2i) = sin\left(pos/10000^{2i/d_{model}}\right) \qquad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (2.4)$$

$$PE(pos, 2i + 1) = \cos\left(pos/10000^{2i/d_{model}}\right) \qquad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (2.5)$$

Setelah



Gambar

Sebagai

2.3.2 Bagian Decoder

Decoder

Bagian

Sebenarnya

Setelah

Dan

2.4 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

d

i

g

u

n

a

k

yang melewati sebuah *threshold*, dan menurunkan nilai yang tidak melewati *threshold* tersebut), contohnya pada rumus $y_k = a(y\_in_k)$ (2.7 dengan a sebagai activation function.

$$y\_in_k = z_j w_k + \cdots + z_n w_n \quad \text{..... (2.6)}$$

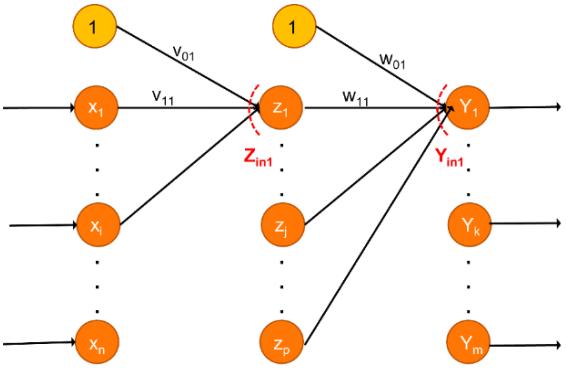
$$y_k = a(y\_in_k) \quad \text{..... (2.7)}$$

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).

2.4.1 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.[8] Contoh alur ini dapat dilihat dari Gambar 2.8. Di mana koneksi antar node di FFN tidak membentuk siklus, namun informasi atau nilai yang diberikan mengarah ke depan (*forward*). Dari input layer, melewati hidden layer dan berakhir pada output layer. FFN perlu diketahui terlebih dahulu karena MLP memiliki struktur yang mirip dengan FFN, tetapi memiliki perbedaan di mana MLP memiliki tiap layer adalah *fully connected layer* dan dalam beberapa kasus setiap layer memiliki jumlah node yang sama.

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



Gambar 2.8
Visualisasi Arsitektur MLP dengan Backpropagation

MLP adalah hasil perkembangan[9] dari algoritma yang dibuat Rosenblatt yaitu *Perceptron*.[10] 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[11] 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 $y\_in_k = z_j w_k + \cdots + z_n w_n$

(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 $\delta_k = (t_k - y_k)\, a'(y\_in_k)$ (2.8) untuk loss function nya. Dengan variabel δ_k adalah nilai loss function, t_k sebagai target output yang diinginkan, f_k nilai node saat itu, a' sebagai turunan pertama activation function.

---

[9] Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press
[10] Frank Rosenblatt. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory 85, 460–461 (1957)
[11] Prof. Dr. Ir. Kuswara Setiawan, M.T., Buku Paradigma Sistem Cerdas, (Malang : Bayu Media, 2003)

$$\delta_k = (t_k - y_k)\, a'(y\_in_k) \quad \text{..... (2.8)}$$

$$\Delta w_k = \alpha\ \delta_k\ z_j \quad \text{..... (2.9)}$$

Rumus $\Delta w_k = \alpha\ \delta_k\ z_j$ (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. Δw_k sebagai nilai gradien, α merupakan nilai *learning rate* dan e_j sebagai nilai node sebelumnya. Tahap yang terakhir adalah tahap backward pass, tahap yang akan melakukan perubahan pada weight sesuai dengan loss function dan gradien yang telah hitung. Untuk penjelasan ini, rumus yang digunakan untuk dapat dilihat pada rumus. Rumus tersebut (dilihat pada rumus $w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_k$ (2.10) diambil dari nilai–nilai sebelumnya yang telah dihitung atau ditemukan.

$$w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_k \quad \text{..... (2.10)}$$

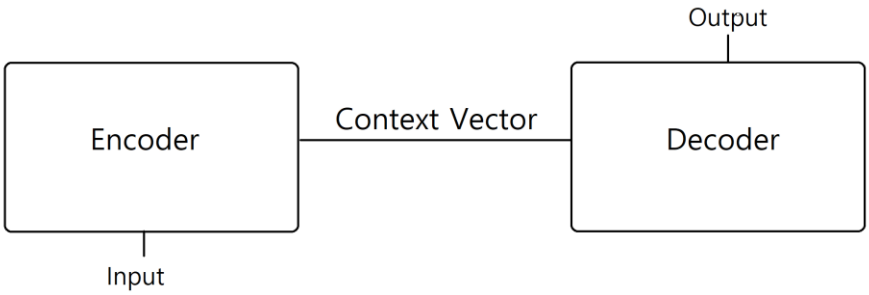
Rumus-rumus yang telah disebut dihitung untuk tiap node yang ada dalam arsitektur MLP, dan tahap-tahap tersebut dilakukan berulang kali sampai iterasi yang telah ditentukan atau saat perubahan nilai gradien bersifat konvergen. MLP sering digunakan dalam dunia machine learning sebagai *classifier* atau 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[12] 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.

[12] 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

2.5 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.[13] 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,

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

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.

2.6 Hungarian Match

Terdapat kesulitan dari penghitungan loss saat training model untuk Sequence-to-Set, yaitu cara memberi nilai terhadap prediksi entitas dan kata yang diprediksikan dengan *golden entities* (target prediksi sebenarnya). Hungarian match[14] adalah algoritma yang akan membantu permasalahan ini dengan kemampuannya untuk mencari *optimal assignment* (pencocokan/penyesuaian optimal) antara prediksi dengan golden entity. Kata *hungarian* dipilih untuk nama algoritma ini oleh penulis Khun karena inspirasinya terhadap dua algoritma lain yang ditemukan oleh penulis yang kedua-duanya berasal dari Hungarian. Dan tujuan dari algoritma Hungarian adalah untuk memberikan solusi terhadap *assignment* problem. Salah satu contoh umum untuk assignment problem adalah pembagian beberapa tugas dengan sejumlah pekerja dengan suatu nilai perbandingan. Nilai perbandingan itu bisa harga jasanya, atau mungkin akurasi dari pekerjaan itu dapat dilakukan dengan lancar.

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

[14] H. W. Kuhn, The Hungarian Method for The Assignment Problem

2.6.1 Simple Assignment Problem

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

$$Q = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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 (*).

$$Q = \begin{bmatrix} 1 & 1 & 1^* & 0 \\ 0 & 0 & 1 & 1^* \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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.

$$Q = \begin{bmatrix} 1^* & 1 & 1 & 0 \\ 0 & 0 & 1^* & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hasil Q matriks terbaru ini menyatakan kolom job 4 dapat ditugaskan kepada individu 3 atau 4. Dengan ini, meskipun untuk individu 1 dapat melakukan transfer juga kepada job 2, assignment ini dianggap complete (gambaran matriks terbaru berada dibawah paragraph ini). Sampai tahap ini, ditemukan beberapa pernyataan yang dapat mendukung teorema pertama nanti. Sebagai lemma pertama, dinyatakan bahwa jika ada person yang ditugaskan kepada suatu job artinya antara

person tersebut atau job tersebut penting (atau kata yang akan digunakan adalah *essential*), namun yang essential adalah salah satu antara individu atau job, tidak bisa kedua-duanya.

$$Q = \begin{bmatrix} 1^* & 1 & 1 & 0 \\ 0 & 0 & 1^* & 1 \\ 0 & 0 & 0 & 1^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

Tetapi ada masalah yang masih belum dipertimbangkan dalam penugasan ini yaitu adanya suatu anggaran yang dibutuhkan untuk menugaskan seorang individu kepada sebuah pekerjaan. Sebuah anggaran dianggap memadai jika, untuk

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

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

2.6.2 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 r_{ij}). 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 $u_i \times v_j \geq r_{ij}$ (2.11.

$$u_i \times v_j \geq r_{ij} \quad \text{..... (2.11)}$$

Sehingga muncul pertanyaan baru khusus untuk dual problem ini yaitu berapa nilai terkecil dari total alokasi budget (contoh total alokasi budget adalah $u_1 + \cdots + u_i + v_1 + \cdots + v_j$) untuk budget adequate? Dari Simple Assignment Problem, teorema 3 memiliki hubungan dengan teorema berikutnya teorema 5 yaitu, niali dari jumlah alokasi adequate budget apa pun adalah lebih besar sama dengan jumlah rating dari assignment apa pun. Agar lebih jelas, diberikan pembuktian (*proof*) untuk teorema 5. Karena setiap individu dan pekerjaan muncul tepat hanya sekali tiap penugasan/assignment, total alokasi anggaran (tiap individu atau pekerjaan) tepat sama dengan nilai jumlah seluruh alokasi anggaran. Anggaran assignment tersebut menjadi memadai/adequate dan bernilai tidak kurang dari total jumlah rating individu yang ditugaskan kepada suatu pekerjaan. Pelambangan dalam rumus matematika, rumus $u_1 + v_1 \geq r_{1j_1}, \ldots, u_n + v_n \geq r_{nj_n}$
(2.12 ditemukan dengan syarat anggaran yang dihitung termasuk anggaran yang memadai (adequate budget) dan rumus tersebut lebih disederhanakan menjadi rumus $u_1 + \cdots + u_n + v_1 + \cdots + v_n \geq r_{1j_1}, + \cdots + r_{nj_n}$ (2.13.

$$u_1 + v_1 \geq r_{1j_1}, \ldots, u_n + v_n \geq r_{nj_n} \quad \text{..... (2.12)}$$

$$u_1 + \cdots + u_n + v_1 + \cdots + v_n \geq r_{1j_1}, + \cdots + r_{nj_n} \quad \text{..... (2.13)}$$

Dampak dari adanya pernyataan dari rumus $u_1 + \cdots + u_n + v_1 + \cdots + v_n \geq r_{1j_1}, + \cdots + r_{nj_n}$ (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 $u_i + v_j = r_{ij}$, jika tidak bisa memenuhi syarat tersebut, individu tersebut tidak memenuhi syarat (*not qualified*). Dapat dilihat secara langsung teorema berikutnya (teorema 6), jika semua n individu dapat ditugaskan (assigned) dan memenuhi syarat (qualified person), dan anggaran penugasan tersebut memadai (adequate budget), artinya penugasan dan anggaran tersebut adalah solusi untuk permasalahannya dan nilai jumlah alokasi anggaran sama dengan nilai jumlah rating penugasan. Hasil dari teorema 6 dapat ditulis ulang secara matematis dari rumus $u_1 + \cdots + u_n + v_1 + \cdots + v_n = r_{1j_1}, + \cdots + r_{nj_n}$ (2.14.

$$u_1 + \cdots + u_n + v_1 + \cdots + v_n = r_{1j_1}, + \cdots + r_{nj_n} \quad \text{..... (2.14)}$$

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 r + s = m (2.15 .

$$r + s = m \quad \text{..... (2.15)}$$

Pernyataan corollary 1 dapat mendefinisikan perbaikan budget dapat dinyatakan:

$$u'_1 = u_1, \dots, u'_r = u_r, u'_{r+1} = u_{r+1} - 1, \dots, u'_n = u_n - 1 \quad \text{..... (2.16)}$$

$$v'_1 = v_1 + 1, \dots, v'_s = v_s + 1, v'_{s+1} = v_{s+1}, \dots, v'_n = v_n \quad \text{..... (2.17)}$$

Inti dari rumus $u'_1 = u_1, \dots, u'_r = u_r, u'_{r+1} = u_{r+1} - 1, \dots, u'_n = u_n - 1$ (2.16 dan $v'_1 = v_1 + 1, \dots, v'_s = v_s + 1, v'_{s+1} = v_{s+1}, \dots, v'_n = v_n$ (2.17, perubahan *u* (budget individu) adalah untuk individu pertama sampai individu ke *r* (jumlah individu penting/essential) akan memiliki nilai yang sama, namun untuk seluruh individu lainnya akan dikurangi (dengan syarat nilai akan non-positif/lebih dari sama dengan nol). Dan untuk perubahan *v* (budget pekerjaan) adalah untuk pekerjaan pertama sampai pekerjaan ke *s* (jumlah pekerjaan penting/essential) akan ditambahkan nilai satu, pekerjaan lainnya akan tetap dengan nilai awalnya.

Setelah prosedur perubahan/perbaikan budget, perlu diperiksa pakah budget terbaru ini memadai/adequate dan total alokasi anggaran telah berkurang dari sebelumnya. Perubahan dianggap gagal apalah bila u_i telah berkurang dan v_j tidak berganti, dengan kegagal ini menyatakan bahwa individu/pekerjaan tersebut tidak penting/inessential. Dan sifat inessential ini ditegaskan oleh teorema 1 karena $u_i + v_j > r_{ij}$. Perubahan budget dianggap adequate karena dapat dinyatakan dalam rumus $u'_i + v'_j = (u_i - 1) + v_j = (u_i + v_j) - 1 \geq r_{ij}$ (2.18.

$$u'_i + v'_j = (u_i - 1) + v_j = (u_i + v_j) - 1 \geq r_{ij} \quad \text{..... (2.18)}$$

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

paling banyak m < n individu bisa ditugaskan, memenuhi syarat dan memiliki anggaran yang memadai, maka nilai jumlah alokasi anggaran bisa berkurang dengan nilai integral yang positif. Dan berakhir pada penemuan jawaban untuk pertanyaan General Assignment Problem, nilai terbesar dari jumlah rating yang memungkinkan adalah sama dengan nilai terkecil jumlah alokasi anggaran dari anggaran apapun yang memadai.

2.6.3 Hungarian Algorithm

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

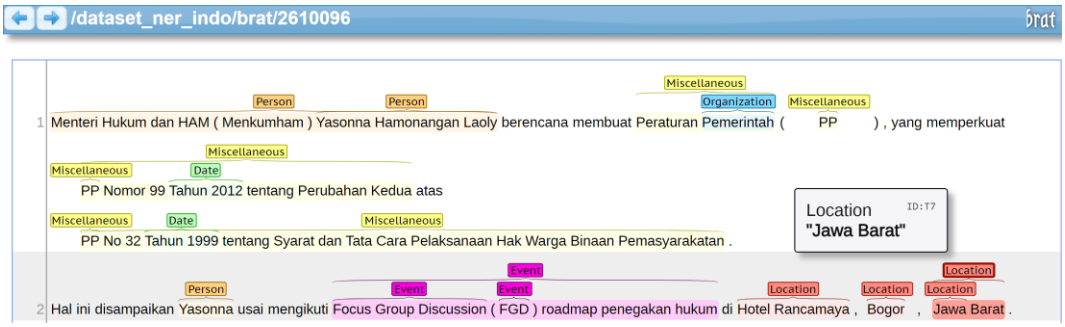
Jika diberikan nilai cover (budget) untuk matriks rating (dilambangkan R), maka ditemukan sebuah kumpulan independent mark terbesar. Jika set tersebut memiliki sejumlah n (jumlah individu/pekerjaan) marks, maka sesuai dengan teorema 6, mark tersebut adalah yang diinginkan untuk assignment tersebut. Jika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

1. Menghitung jumlah rating tiap baris dan kolom, dan mencari nilai terbesar dari semua baris (dilambangkan sebagai b), dan nilai terbesar dari semua kolom (dilambangkan sebagai a).
2. Untuk menentukan nilai dari anggaran ditentukan dari perbandingan nilai a dan b sebelumnya. Jika b ≥ a, maka u_i=a_i dan v_j=0. Untuk sebaliknya, a > b, maka v_j=a_i dan u_i=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, q_{ij} diisi angka 1 apabila $u_i + v_j = r_{ij}$, 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.

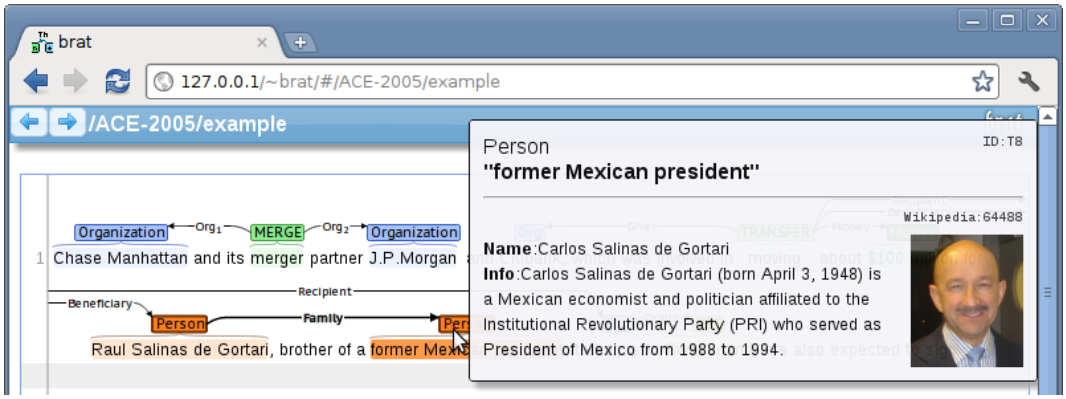
2.7 BRAT

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* (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[15]

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

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

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