**SISTEM TEMU KEMBALI INFORMASI**

**RANGKUMAN MINGGU 5, 6, DAN 7**

****

**Muhammad Fauzan Yamani**

**A11.2020.12927**

**A11.4703**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS DIAN NUSWANTORO OKTOBER 2023**

# 1. Word Embedding

## A. Penjelasaan

Word Embedding atau Penyematan Kata adalag teks yang diubah menjadi angka dan mungkin ada representasi numerik yang berbeda dari teks yang sama. Sebagai contoh Sekarang mari kita definisikan Word Embeddings secara formal. Format Penyematan Kata umumnya mencoba memetakan kata menggunakan kamus ke vektor. Mari kita pecahkan kalimat ini menjadi detail yang lebih baik untuk mendapatkan gambaran yang jelas.

sentence= `Word Embeddings are Word converted into numbers`,   
maka dictionary = `[‘Word’,’Embeddings’,’are’,’Converted’,’into’,’numbers’]`

Representasi vektor dari sebuah kata mungkin merupakan vektor one-hot encoding di mana 1 mewakili posisi di mana kata itu ada dan 0 di tempat lain. Representasi vektor '**numbers**' dalam format ini menurut kamus di atas adalah

[‘Word’,’Embeddings’,’are’,’Converted’,’into’,’**numbers**’]`  
[0,0,0,0,0,1]

Representasi Vector '**Converted**' adalah

[‘Word’,’Embeddings’,’are’,’**Converted**’,’into’,’numbers’]`  
[0,0,0,1,0,0].

## B. Type Word Embedding

### a. Penyematan berbasis frekuensi (Frequency based Embedding)

#### a) Count Vector

Count Vector adalah representasi vektor dari dokumen atau teks yang mengukur jumlah kemunculan kata-kata dalam dokumen. Teknik ini digunakan dalam pemrosesan bahasa alami (NLP) untuk mengubah teks menjadi representasi vektor yang dapat diolah oleh algoritma pembelajaran mesin.

Setiap dokumen digambarkan sebagai vektor dengan setiap dimensi vektor menggambarkan sebuah kata, dan nilai dari vektor tersebut adalah jumlah kata yang muncul dalam dokumen tersebut. Prosesnya melibatkan menghitung frekuensi kemunculan setiap kata dalam dokumen, dan vektor yang dihasilkan mencerminkan distribusi kata-kata tersebut dalam dokumen.

Untuk contoh, lihat dokumen Corpus *C* dari *D*

*d1,d2…..dD*

dan *N* token unik yang diekstraksi dari corpus *C*. Token *N* akan membentuk kamus dan ukuran hitungan vektor *M* akan diberikan oleh *D X N*. Setiap baris dalam matriks *M* berisi frekuensi token dalam dokumen *D(i)*.

contoh:

D1: He is a lazy boy. She is also lazy.  
D2: Neeraj is a lazy person.

dictionary yang terbuat:

[‘He’,’She’,’lazy’,’boy’,’Neeraj’,’person’],  
maka *D = 2* (ada 2 dokumen) dan *N = 6* (ada 6 isian di list)

matrik *M* dengan ukuran *2 X 6* dapat ditulis sebagai berikut:

He She lazy boy Neeraj person

|D1| 1 |1 |2 |1 |0 |0|  
|D2| 0 |0 |1 |0 |1 |1|

Sekarang, kolom juga dapat dipahami sebagai word vector / vektor kata untuk kata yang sesuai dalam matriks M. Misalnya, vektor kata untuk 'lazy' dalam matriks di atas adalah [2,1] dan seterusnya. Di sini, baris sesuai dengan dokumen dalam korpus dan kolom sesuai dengan token dalam kamus. Baris kedua pada matriks di atas dapat dibaca sebagai – D2 berisi 'lazy': sekali, 'Neeraj': sekali dan 'person' sekali.

Untuk implementasi pada python akan seperti ini,

from sklearn.feature\_extraction.text import CountVectorizer

import pandas as pd

doc1 = "He is a lazy boy. She is also lazy."

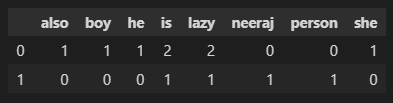
doc2 = "Neeraj is a lazy person."

bow\_vectorizer = CountVectorizer()

X = bow\_vectorizer.fit\_transform([doc1,doc2])

bow\_df = pd.DataFrame(X.toarray(),columns=bow\_vectorizer.get\_feature\_names\_out())

bow\_df.head()



#### b) TF-IDF Vector

Dalam pemrosesan teks, vektor TF-IDF (Term Frequency-Inverse Document Frequency) digunakan untuk menggambarkan bobot kata-kata dalam dokumen berdasarkan nilai TF-IDF dari setiap kata.

TF-IDF Vector menghitung skor untuk setiap kata dalam dokumen berdasarkan dua faktor.

1. Term Frequency (TF) adalah ukuran seberapa sering sebuah kata muncul dalam sebuah dokumen. Untuk menghindari bias dalam dokumen yang lebih panjang, ini biasanya dinormalisasi dengan jumlah total kata dalam dokumen.
2. Inverse Document Frequency (IDF) adalah ukuran seberapa pentingnya sebuah kata dalam korpus atau koleksi dokumen. Kata-kata yang jarang muncul di banyak dokumen memiliki skor IDF yang lebih tinggi, sementara kata-kata yang umum mungkin memiliki skor IDF yang lebih rendah.

Setiap kata yang ada dalam dokumen diberi skor TF-IDF oleh gabungan TF dan IDF; vektor TF-IDF mewakili dokumen sebagai vektor di mana setiap dimensi mewakili kata, dan nilainya adalah skor TF-IDF dari kata tersebut.

#### c) Co-Occurence Vector

Co-Occurence Vector, juga dikenal sebagai Vektor Kemunculan Bersama, adalah representasi vektor yang menunjukkan hubungan antara kata-kata berdasarkan frekuensi kemunculan bersama (co-occurrence) dalam teks. Metode ini menghasilkan vektor yang menunjukkan kemunculan bersama antara kata-kata dalam konteks tertentu.

Matriks kemunculan bersama merekam seberapa sering pasangan kata muncul bersama dalam suatu jendela konteks dalam teks atau korpus. Misalnya, jika kata A muncul bersama kata B beberapa kali dalam jendela teks tertentu, matriks kemunculan bersama akan merekam frekuensi kemunculan bersama ini.

Proses ini menghasilkan vektor dimensi yang besar, dengan nilai dalam setiap dimensi menunjukkan hubungan kemunculan bersama antara kata-kata dalam korpus, dan setiap dimensi mewakili kata-kata sendiri.

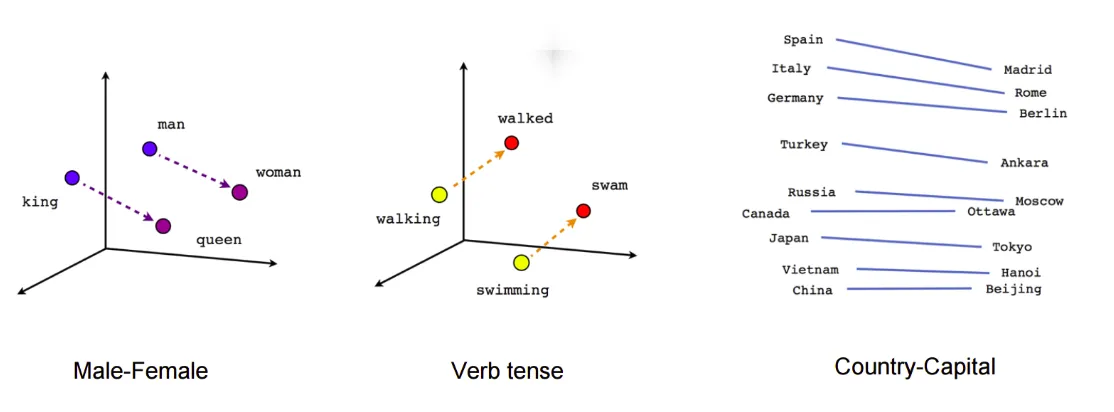
### b. Penyematan berdasarkan prediksi (Prediction based Embedding)

#### a) Word2Vec

Word2vec adalah suatu metode untuk merepresentasikan setiap kata di dalam konteks sebagai vektor dengan N demensi. Dalam mempresentasikan suatu kata, Word2Vec mengimplementasi neural network untuk menghitung contextual and semantic similarity (kesamaan kontekstual dan semantik) dari setiap kata (inputan) yang berbentuk one-hot encoded vectors.

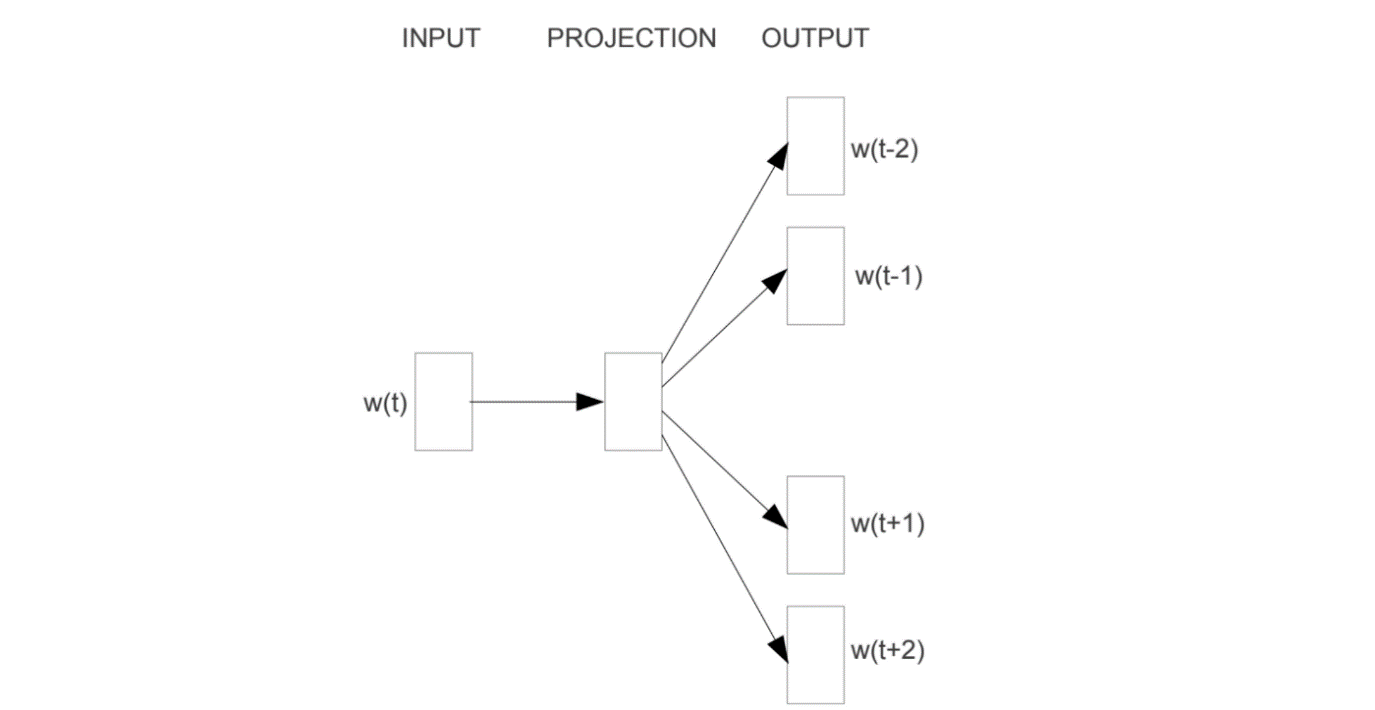
Misalnya sebuah kata “Indonesia” di representasikan menjadi sebuah vektor dengan panjang 5 yaitu : [0.2, 0.4, -0.8, 0.9, -0.5]. Vektor tersebut tidak hanya merepresentasikan kata secara sintaktik tapi juga secara semantik atau secara makna.

Sebagai contoh, apabila word2vec di train menggunakan korpus yang cukup lengkap, maka vektor representasi dari kata “Indonesia” akan berdekatan dengan vektor “Jakarta” sebagaimana vektor “Perancis” akan berdekatan dengan vektor “Paris”. Dengan kata lain, model word2vec akan memahami bahwa “Indonesia” dan “Jakarta” memiliki hubungan yang sama dengan “Japan” dan “Tokyo” yaitu negara dan ibukotanya.



Terdapat 2 model untuk Word2Vec yaitu Skip-gram dan Continous Bag-of-Word (CBOW).

#### 1) Skip-gram



Tujuan dari arsitektur skip-gram adalah untuk memprediksi konteks (output) di sekitar current word (input). Untuk memahaminya mari kita lihat ilustrasi berikut. Misal data trainingnya adalah sebuah kalimat “Ibu kota Negara Indonesia adalah Jakarta” dengan window size = 2

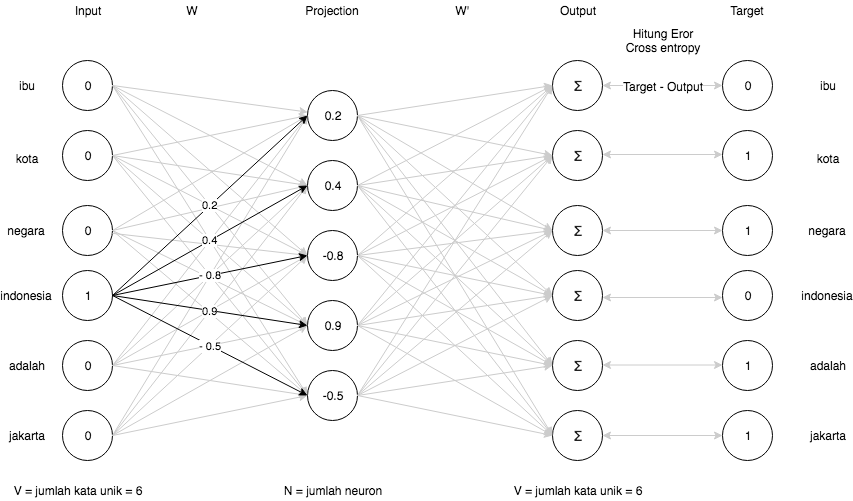
Untuk contoh ilustrasinya :

* Jika inputannya “Ibu”, maka target outputnya adalah “kota” dan “Negara”
* Jika Inputannya “Negara”, maka target outputnya adalah “Ibu”, “kota”, “Indonesia”, dan “adalah”
* Jika inputnya “adalah”, maka target outputnya adalah “Negara”, “Indonesia”, dan “Jakarta”

Data input berbentuk one-hot encoded vector sehingga bentuk datanya adalah seperti berikut (anggap seluruh data di lowercase terlebih dahulu):

* ibu :[1,0,0,0,0,0]
* kota :[0,1,0,0,0,0]
* negara :[0,0,1,0,0,0]
* indonesia :[0,0,0,1,0,0]
* adalah :[0,0,0,0,1,0]
* jakarta :[0,0,0,0,0,1]

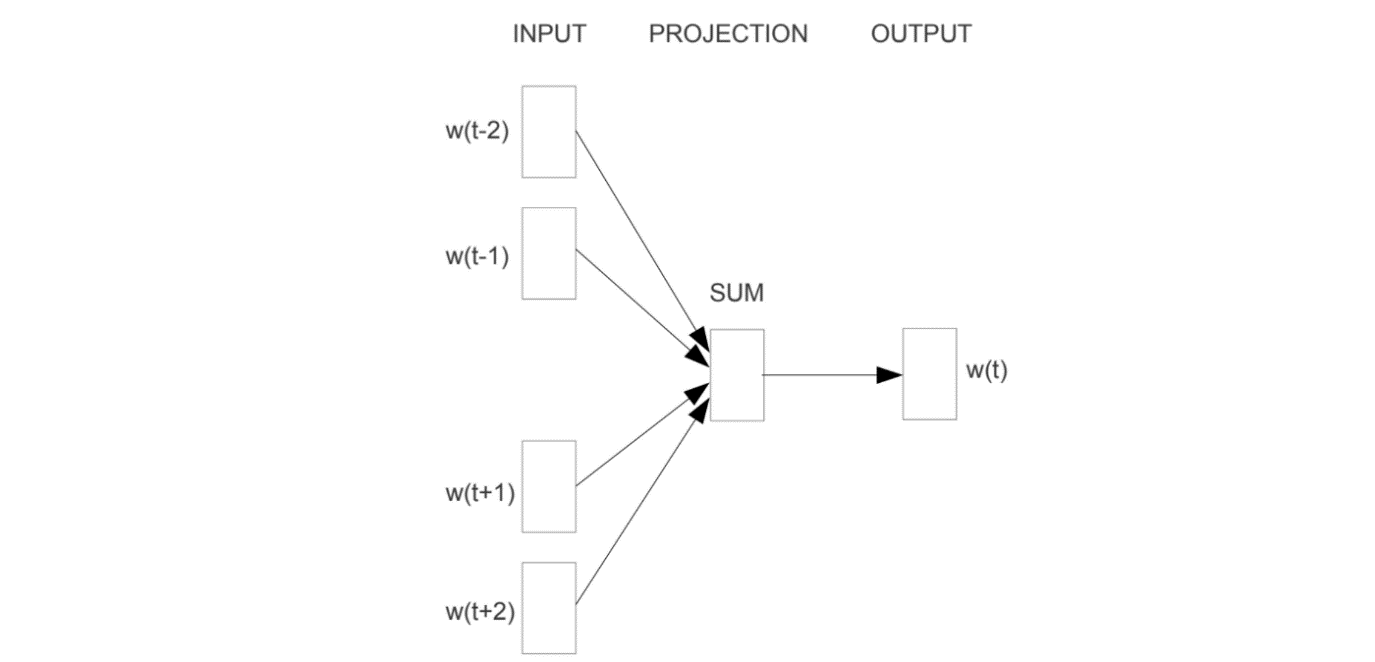
Misal current word = Indonesia, maka ilustrasinya adalah sebagai berikut (Feed forward Neural Network Skip-gram) :



Inisialisasi bobot pada W dan W’ adalah random. Bobot W dan W’ merupakan matrik dengan ukuran W = V x N dan W’ = N x V. Pada proses feedforward, vektor input akan di dot product dengan bobot W dan menghasilkan nilai pada layer projection. Kemudian layer projection di dot product dengan bobot W’ dan menghasilkan vektor output. Setelah mendapatkan nilai output pada tahap feedforward, maka akan dihitung nilai eror nya dengan menggunakan metode cross entropy yaitu Target - Output. Selanjutnya adalah tahap backpropagation dengan memanfaatkan teknik gradient descent yaitu dengan melakukan update bobot W dan W’. Proses ini akan diulang kembali ke tahap feedforward hingga tercapai nilai eror minimum.

Setelah didapatkan nilai eror minimum pada cross entropy, maka vektor yang merepresentasikan kata tersebut diambil dari bobot W dengan cara mengalikan dot product antara one-hot encoded vector masing-masing kata dengan bobot W, sedangkan bobot pada W’ akan diabaikan.

#### 2) Continous Bag of Word (CBOW)



Arsitektur Word2Vec CBOW adalah kebalikan dari Word2vec skip-gram. Tujuannya adalah untuk memprediksi kata (output) ketika diberikan konteks disekitar kata tersebut (input). Berikut ini adalah ilustrasinya.

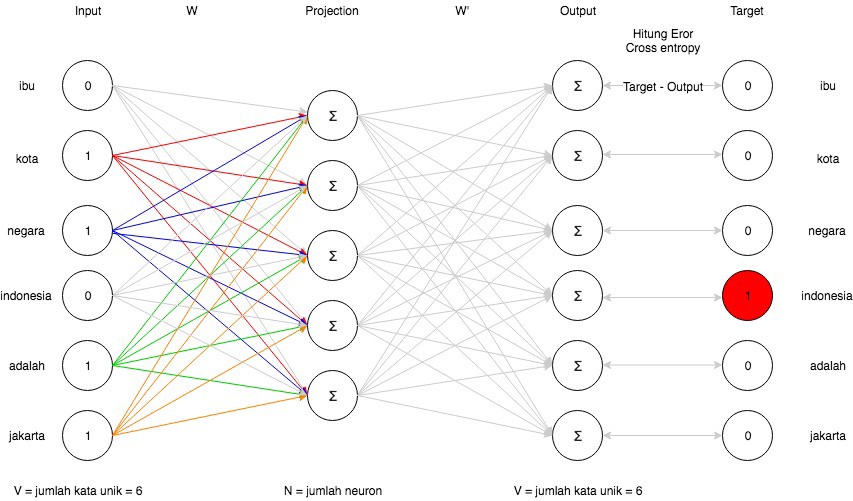
Saya menggunakan kalimat yang sama dengan yang dicontohkan sebelumnya yaitu kalimat “Ibu kota Negara Indonesia adalah Jakarta”

Untuk contoh ilustrasi :

* Jika inputnya “kota Negara” dan “adalah Jakarta”, maka target outputnya adalah “Indonesia”
* Jika inputnya “Ibu” dan “Negara Indonesia”, maka target outputnya adalah “kota”
* Jika inputnya “Indonesia adalah”, maka target outputnya adalah “Jakarta”

Data input Word2Vec Skip-gram berbentuk n-hot encoded vector. Ketika proses training berjalan, kata yang sedang menjadi inputan akan bernilai 1 sedangkan kata yang lainnya akan bernilai 0. Karena targetnya hanya 1 kata, maka target output akan berbentuk one-hot encoded vector.

Berikut ini ilustrasi Feedforward ketika target outputnya adalah kata “Indonesia”



Prosesnya tidak jauh berbeda dengan arsitektur Skip-gram. Perbedaannya hanya terletak pada data inputnya yang berupa n-hot encoded vector dan target outputnya berupa one-hot encoded vector.

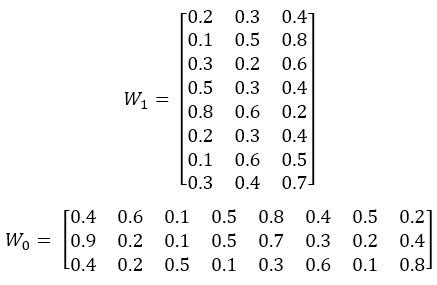
Setelah dilakukan training hingga mencapai eror minimum, maka kita dapat mengambil vektor representasi kata dengan cara mengalikan one-hot encoded vector masing masing kata dengan bobot W.

Misal kita memiliki sebuah kalimat “Kucing melihat seekor tikus, kucing mengejar tikus, tikus berlari keluar rumah” korpus kosakata dalam kalimat tersebut memiliki delapan kata, selanjutnya disusun berdasarkan urutan abjad, masing-masing kata diberi indeks

Kosakata (Indeks)

- Berlari (1)  
- Keluar (2)  
- Kucing (3)  
- Melihat (4)  
- Mengejar (5)  
- Rumah (6)  
- Seekor (7)  
- Tikus (8)

Dalam Tabel 2 jaringan syaraf memiliki delapan neuron input dan delapan neuron output. Asumsikan bahwa akan digunakan tiga neuron pada hidden layer. Hal ini akan membentu matriks W1 = 8×3 dan W0 = 3×8. Sebelum memulai pelatihan, matriks ini akan diinisialisasi sebagai nilai acak yang kecil seperti biasa dalam pelatihan jaringan syaraf tiruan. Nilai W1 dan W0 dapat diinisialisasikan sebagai berikut:



Misalkan jaringan akan mempelajari hubungan antara kata “keluar” dengan kata “melihat”. Maka jaringan harus menunjukkan probabilitas tinggi terhadap kata “melihat” saat kata “keluar” dimasukkan dalam jaringan. Secara terminologi word embedding, kata “keluar” disebut konteks dan kata “melihat” disebut sebagai sasaran kata. Diasumsikan vektor input pada kata “keluar” akan dikodekan menjadi [01000000], indeks kedua bernilai 1, hal ini karena kata input “keluar” berada pada posisi kedua dalam daftar korpus kata. Sementara kata “melihat” yang menjadi target dikodekan menjadi [00010000]. Dengan input vektor dari kata “keluar”, output pada hidden layer neuron dapat dihitung dengan persamaan 1



Sebagaimana H adalah hidden layer, X adalah neuron input sebelumnya dan W1 adalah bobot, output dari hidden neuron yaitu H yang direpresentasikan dari baris kedua matriks W1. Hal ini karena input yang diberikan ke jaringan dikodekan menggunakan representasi “1-out of-V”. Sehingga hanya satu baris masukkan yang mempunyai nilai satu dan sisanya untuk jalur input diatur ke nol. Dengan demikian fungsi input ke koneksi hidden layer yaitu menyalin vektor input kata ke hidden layer, kemudian untuk outputnya hidden layer diperoleh pada persamaan 2.



Setelah itu untuk menghasilkan probabilitas kata, Word2vec menggunakan fungsi Softmax dalam lapisan output, maka probabilitas dari delapan kata dalam korpus dapat dihitung menggunakan persamaan 3.



Dengan demikian probabilitas dari delapan kata dalam korpus menghasilkan



Nilai yang dicetak tebal merupakan hasil probabilitas kata “melihat” mengingat vektor sasaran berada pada indeks keempat [00010000] dalam korpus kata. Kesalahan vektor untuk output layer dapat dihitung dalam mengurangkan vektor probabilitas dengan vektor sasaran. Setelah diketahui tingkat kesalahan, bobot pada matriks W1 dan W0 dapat diperbarui menggunakan backpropagation. Setelah ini, pelatihan dapat diteruskan terhadap berbagai pasangangan konteks-target kata dari korpus.

# 2. Search Engine Concept

## A. Penjelasan

Search Engine atau Mesin Pencarian adalah sebuah kode software yang digunakan untuk mencari informasi di web. Informasi yang ditemukan, disajikan pada halaman hasil mesin pencari.

Contoh mesin pencari yang sering digunakan :

* Google
* Yahoo
* Bing
* Ask
* Yandex
* dll

## B. Cara Kerja Search Engine

### a. Webcrawling

Mesin pencari menjelajah web untuk mengumpulkan berbagai informasi terkait website yang terhubung dengan web.

### b. Indexing

Mesin pencari menyusun informasi yang ditemukan tadi kedalam database untuk digunakan lagi ketika user melakukan pencarian.

### c. Searching

Keyword yang dimasukkan oleh pengguna akan dicocokan dengan database yang dimiliki. Bila ada yang sesuai maka data akan disajikan, bila tidak maka kata kunci tersebut akan digunakan sebagai “misi baru” untuk menjelajah web. Harapannya, ketika pengguna menggunakan kata kunci tersebut, mesin pencari akan dapat menyajikan informasi yang diminta pengguna.

## C. Mekanisme Google

Beberapa syntax yang sering digunakan dalam pencarian melalui Google :

* intitle and allintitile
* inurl and allinurl
* site
* filetype

### a. intitle

Berfungsi untuk mempermudah Google membatasi hasil pencarian pada halaman yang terdapat pada judul atau title

**do** -> **intitle:sistem informasi manajemen**

hasil -> akan menampilkan halaman yang menggunakan judul sistem dan pada isi halaman terdapat kata informasi

### b. allintitle

Untuk pencarian lebih dalam atau bila ingin mendapatkan hasil dari dua kata tersebut, digunakan perindah allintitle

**do** -> **allintitle:sistem informasi manajemen**

hasil -> akan menampilkan hasil pencarian hanya pada dua kata tersebut, yaitu sistem dan informasi.

### c. inurl

Berfungsi untuk membatasi hasil pencarian pada URL yang mengandung kata kunci yang dikehendaki

**do** -> **inurl:sistem informasi manajemen**

hasil -> akan menampilkan URL yang mengandung info tentang sistem dan informasi.

### d. allinurl

Sama seperti allintitle, tidak dapat digabung dengan syntax lainnya

**do** -> **allinurl:sistem informasi manajemen**

hasil -> akan mengandung URL yang hanyya mengandung info tentang sistem dan informasi

Catatan : syntax allinurl tidak dapat digabung dengan syntax lainnya

### e. site

Menampilkan situs dengan ekstensi tertentu, misal .id, .com, .net, dst

**do** -> **sistem informasi manajemen site:ac.id**

**sistem informasi manajemen site:id**

### f. filetype

Menampilkan sebuah situs yang memiliki file tertentu. Format yang dapat dicari oleh Google antara lain: \*.pdf, \*.xls, \*.doc, \*.ppt, \*.ps, \*.swf, \*.txt, \*.rtf, dan lainnya.

**do ->**  **sistem informasi manajemen filetype:pdf**

## E. Pengalaman User

Ada beberapa kejadian user ketika menggunakan search engine, seperti:

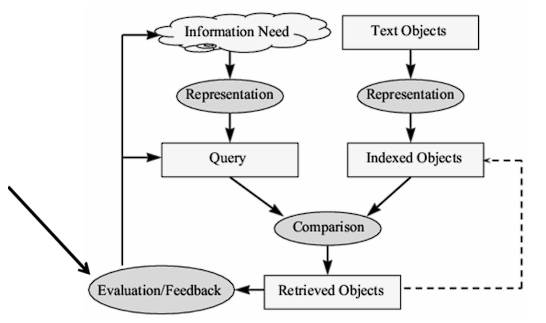
* Hasil pencarian setiap devices berbeda
* Hasil pencarian setiap search engine berbeda

Untuk alasan pertama, setiap user memiliki devices-nya masing-masing. Maka dari itu setiap user memiliki riwayat dan preferensi masing-masing juga. Semisal user A suka dengan hal memancing, maka search engine akan menampilkan pencarian yang berhubungan dengan memancing. Bisa juga pengaruh geografis, semisal user A tinggal di Indonesia, maka akan ditampilkan informasi seputar memancing di daerah Indonesia.

Untuk alasan kedua, karena setiap search engine memiliki database dan algoritmanya masing-masing. Semisal user A mencari tentang memancing, di search engine A menampilkan laman A dan di search engine B menampilkan laman B.

# 3. Evaluasi

## A. Penejelasan



Cara mengetahui bahwa hasil yang diperoleh yang relevan adalah :

* Mengevaluasi suatu “Search Engine”
* Benchmark (Patokan)
* Presisi dan Recall
* Akurasi
* Ketidaksepekatan antar hakim
* Normalisasi potongan untung kumulatif
* Pengujian A/B

Masukan yang diberikan user, akan di-feedback dan akan memberikan manfaat kepada user.

## B. Macam-Macam Ukuran

### 1. Ukuran bagi Search Engine

* Seberapa cepat **membangun indeks**
* Jumlah dokumen/jam
* Rata-rata ukuran dokumen
* Seberapa cepat **melakukan pencarian**
* Latency sebagai fungsi dari ukuran indeks
* Ekspresi dari bahasa query
* Kemampuan mengekspresikan kebutuhan informasi yang kompleks
* Kecepatan pemrosesan query kompleks
* Seberapa murah dan rapi atau tertara UI (User Interface) yang digunakan
* Apakah gratis atau berbayar
* Semua kriteria tersebut measurable: dapat dihitung kecepatan/ukurannya
* Dapat diekspresikan dengan tepat
* Ukuran kunci: happines
* Yaitu kecepatan respon/ukuran dari indeks adalah faktor penting
* Tetapi tidak asal cepat, jawabn yang tak berguna membuat pengguna kecewa
* Perlu cara menghitung kepuasan pengguna

### 2. Ukuran Kebahagiaan Pengguna

* Masalah: Siapa pengguna yang akan dibuat bahagia?
* Web Engine:
* Pengguna mencari apa yang diinginkan dan kembali ke engine
* Dapat diukur angka pengguna yang kembali
* Pengguna melengkapi tugasnya: pencarian sebagai alat (sarana), bukan akhir
* Situs eCommerce: pengguna mencari apa yang diinginkan dan dibeli
* Kepuasan bagi end-user atau situs eCommerce?
* Waktu belanja atau % pencarian yang menjadi pembelian?
* Recommender System: pengguna mencari rekomendasi yang berguna atau sistem mampu memprediksi rating pengguna

Untuk mengukur kebahagiaan pengguna, menggunakan ***Enterprise ­***(perusahaan/pemerintah/kampus) harus ***peduli*** dengan “***produktifitas pengguna***”

* Berapa waktu yang dihemat oleh pengguna ketika mencari informasi?
* Banyak kriteria lain yang harus diperhatikan, terutama yang berkaitan dengan keleluasaan, kemudahan dan keamanan akses.

Kebahagiaan yang sukar diukur :

* Proxy paling umum: Relevansi hasil pencarian.
* Tetapi bagaimana mengukur relevansi?
* Ada metodologi => ada persoalan yang muncul.
* Ukuran relevansi memerlukan 3 elemen:

1. Koleksi dokumen benchmark.
2. Paket query benchmark.
3. Biasanya taksiran biner: Relevan atau Tak-Relevan untuk setiap query dan setiap dokumen.

* Ada yang tak biner, tapi tak standard.

Kebutuhan => Query



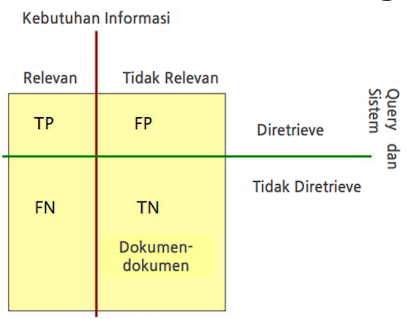
## C. Mengevaluasi Sistem STKI

* Kebutuhan informasi diterjemahkan ke dalam Query.
* Relevansi ditaksir relatif terhadap kebutuhan informasi, bukan Query.
* Misal, kebutuhan informasi: “I’m looking for information on whether using olive oil is effective at reducing your risk of heart attacks.”
* Query: olive oil heart attack effective.
* Evaluasi: apakah dokumen menjawab kebutuhan informasi, atau hanya menyesuaikan kata-kata yang terkandung dalam query.

### a. Benchmark Relevansi Standard

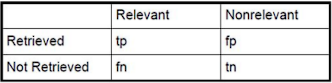
* TREC – National Institute of Standards and Technology (NIST) telah menjalankan test bed TKI besar selama bertahun-tahun.
* Dokumen benchmark: Reuters dan lainnya.
* “Tugas-tugas Retrieval” ditetapkan:
* Kadang kala sebagai query
* Pakar manusia menilai kedekatan setiap query dengan untuk setiap dokumen: Relevan atau Tak-relevan.
* Atau setidaknya untuk subset dari dokumen yang dikembalikan oleh sistem untuk query tersebut

### b. Relevansi dan Dokumen Yang Ditemukan Kembali



### c. Evaluasi Retrieval Tak-Terangking:

* Presisi: % dokumen yang diretrieve dan relevan = P(relevan | diretrieve)
* Recall: % dokumen relevan yang berhasil diretrieve = P(diretrieve | relevan)



* Presisi P = tp/(tp + fp) = tp/diretrieve
* Recall R = tp/(tp + fn) = tp/relevan

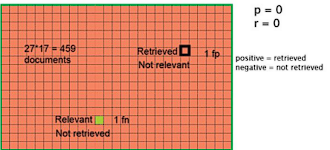
### d. Akurasi

* Diberikan suatu query, suatu engine (classifier) mengelompokkan setiap dokumen sebagai “Relevan” atau “Tak-relevan”.
* Apakah yang diretrieve terklasifikasi oleh engine sebagai “relevan” dan apakah yang tidak diretrieve diklasifikasikan sebagai “tak- relevan”.
* Akurasi dari engine: % ketepatan dari klasifikasi
* (tp + tn) / (tp + fp + fn + tn)
* Akurasi adalah ukuran evaluasi yang umum digunakan dalam kerja klasifikasi machine learning.
* Mengapa ini bukan ukuran evaluasi yang sangat penting

Mengapa tidak hanya menggunakan Akurasi saja? Sistem Temu Kembali Informasi digunakan untuk mendapatkan sesuatu dan mempunyai toleransi tertentu terhadap sampah (junk).

### e. Presisi, Recall, dan Akurasi

* Presisi sangat rendah, Recall sangat rendah, Akurasi tinggi.



* A = (tp + tn) / (tp + fp + fn + tn) = (0 + (27\*17 - 2)) / (0+1+1+(27\*17 - 2)) = 0.996