

**RANCANG BANGUN SISTEM Pencarian Teks
DENGAN MENGGUNAKAN MODEL
CONTINUOUS-BAG-OF-WORDS DAN MODEL
CONTINUOUS SKIP-GRAM PADA KOLEKSI
DOKUMEN**

Proposal Skripsi

**Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer**



**Muhammad Zalghornain
3145153000**

**PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA**

2022

LEMBAR PERSETUJUAN

Dengan ini saya mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Negeri Jakarta

Nama : Muhammad Zalghornain
No. Registrasi : 3145153000
Program Studi : Ilmu Komputer
Judul : Rancang Bangun Sistem Pencarian Teks Dengan
Menggunakan Model *Continuous-Bag-Of-Words* Dan
Model *Continuous Skip-Gram* Pada Koleksi Dokumen

Menyatakan bahwa proposal skripsi ini telah siap diajukan untuk seminar pra skripsi.

Menyetujui:

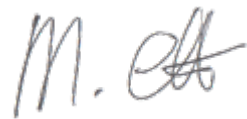
Dosen Pembimbing I



Ria Arafiah, M.Si

NIP. 19751121 200501 2 004

Dosen Pembimbing II



Muhammad Eka Suryana, M.Kom

NIP. 19851223 201212 1 002

Mengetahui,

Koordinator Program Studi Ilmu Komputer

Ir. Fariani Hermin M.T

NIP. 19600211 198703 2 001

DAFTAR ISI

LEMBAR PERSETUJUAN	i
DAFTAR ISI	ii
DAFTAR GAMBAR	iv
BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Perumusan Masalah	3
C. Pembatasan Masalah	4
D. Tujuan Penelitian	4
E. Manfaat Penelitian	4
BAB II KAJIAN PUSTAKA	5
A. <i>Natural Language Processing</i>	5
B. <i>Neural Network</i>	5
C. <i>Query Expansion</i>	6
D. <i>Word Embedding</i>	7
E. Representasi kata dalam bentuk vektor	7
F. <i>Word2Vec</i>	8
G. <i>Precision</i>	14
BAB III METODOLOGI PENELITIAN	16
A. Metodologi Pengembangan Sistem	16
B. Analisis Data	18
C. Proses <i>Training</i>	19
1. <i>Hidden Layer</i>	21
2. <i>Output Layer</i>	22

3. <i>Update Weight Matrix Output</i>	23
4. <i>Update Weight Matrix Input</i>	25
5. Penggunaan Sistem Untuk Melakukan Pencarian	26
6. Perankingan Dokumen	27
7. Data Keluaran dan Evaluasi Sistem	27
D. Rancangan Eksperimen	28
DAFTAR PUSTAKA	30

DAFTAR GAMBAR

2.1	Model CBOW	9
2.2	Model <i>Skip-Gram</i>	12
3.1	Proses Sistem	17
3.2	Proses <i>Training</i>	19
3.3	<i>Hidden Layer</i>	21
3.4	<i>Output Layer</i>	22
3.5	<i>Update Weight Matrix Output</i>	23
3.6	<i>Update Weight Matrix Input</i>	25

BAB I

PENDAHULUAN

A. Latar Belakang

Web merupakan tempat penyimpanan informasi dalam bentuk teks, gambar, audio dan video. Dengan banyaknya informasi yang ada di web, diperlukan alat untuk mencari dan mendapatkan sesuatu yang diperlukan pengguna dengan mudah.

Web Search Engine merupakan sistem perangkat lunak yang di desain untuk melakukan pencarian. Definisi dasar *search engine* bisa merujuk pada sistem *Information Retrieval* (IR) yang memungkinkan pencarian “kata kunci” pada teks digital terdistribusi (Halavais, 2017). *Search engine* melakukan pencarian pada sistem dengan sistematis untuk menemukan informasi spesifik yang di *input*. Hasil pencarian biasanya di tampilkan dalam bentuk *list*. Informasi yang ditampilkan bisa campuran dari *link* ke halaman web, gambar, video, artikel, paper, dan hal lainnya. Beberapa *search engine* juga melakukan pencarian pada database dan direktori terbuka. Tidak seperti web direktori yang perlu di tangani oleh manusia, *search engine* juga bisa menjaga informasi selalu *real-time* dengan menggunakan algoritma pada *web crawler*. *Web search engine* merupakan salah satu contoh pengaplikasian *Information Retrieval* atau Sistem Temu Balik Informasi.

Information Retrieval (IR) merupakan pencarian material (biasanya dokumen) tak berstruktur (biasanya teks) yang memenuhi kebutuhan informasi dari koleksi besar (biasanya disimpan di komputer) (Manning et al., 2010). IR dibutuhkan untuk mensortir data yang relevan secara otomatis terhadap *input* yang dimasukkan *user*, sehingga *user* tidak perlu mensortir data sendiri, terutama jika data tersebut ada dalam jumlah yang besar.

Perangkingan merupakan hal yang penting dalam *Information Retrieval*, mengembalikan dokumen yang diinginkan *user* merupakan bagian yang penting dalam *search engine*. Sistem pencarian tradisional biasanya menggunakan jumlah frekuensi kata pada dokumen untuk mencari dokumen yang relevan terhadap kueri *user*.

Walaupun metode ini tidak membutuhkan biaya komputasi yang tinggi, model ini memiliki kelemahan pada keterbatasan kata. Ketika pencari tidak mengetahui istilah tepat untuk suatu konsep, akan cukup sulit bagi pencari untuk mendapatkan informasi yang ia inginkan. Maka dari itu dibutuhkan suatu cara untuk mengerti kueri *user*, untuk mengerti apa yang *user* ingin cari terlepas keterbatasan kueri yang dimasukkan *user*.

Query Expansion merupakan teknik yang biasa di gunakan pada *Information Retrieval* untuk meningkatkan performa pengambilan data dengan memodifikasi kueri original, dengan menambahkan kata baru atau pembobotan ulang kata original. Menurut Vechtomova and Wang (2006), “*Query Expansion* idealnya harus memiliki beberapa karakteristik, salah satunya merupakan hubungan semantik dengan kueri original”. Penimbangan semantik kata bertujuan untuk mengerti maksud *user* untuk menghasilkan hasil pencarian yang lebih relevan. Dengan mempertimbangkan makna semantik diharapkan dapat meningkatkan kualitas pencarian dan mengurangi waktu *user* untuk memilah-milah dokumen yang relevan. Pada penelitiannya yang berjudul “*Enhanced word embedding similarity measures using fuzzy rules for query expansion*”, Liu et al. (2017) menemukan bahwa *Query Expansion* menggunakan *Word Embedding* dapat meningkatkan performa perankingan dokumen dibanding metode tradisional.

Word Embedding merupakan representasi kata-kata dalam bentuk vektor, dimana kata yang mirip akan berdekatan pada ruang vektor. *Word Embedding* dapat menangkap makna semantik dari kata. Salah satu cara untuk menghasilkan *Word Embedding* merupakan *Word2vec*. Menurut Al-Saqqa and Awajan (2019), *Word2vec* merupakan salah satu model paling umum yang digunakan dalam *Word Embedding*. *Word2vec* menggunakan *neural network* yang terdiri dari dua model yaitu *Continuous Bag-of-Words* atau CBOW dan *Continuous Skip-Gram* atau *Skip-Gram*. CBOW menggunakan kata konteks (kata di sekitar kata target) untuk memprediksi kata target, sedangkan *Skip-Gram* menggunakan kata target untuk memprediksi kata di sekitarnya atau kata konteks. Pada penelitiannya, Mikolov et al. (2013) menemukan

bahwa CBOW memiliki hasil akurasi sintaksis lebih tinggi daripada *Skip-Gram* terhadap kata yang sering muncul, sedangkan *Skip-Gram* memiliki akurasi semantik yang lebih tinggi dibanding CBOW. Pendekatan *Word2vec* telah banyak digunakan di berbagai eksperimen dan menjadi pijakan dalam meningkatkan ketertarikan pada *Word Embedding* sebagai teknologi.

Proses pencarian pada *search engine* terdiri dari *crawling*, *indexing*, dan *searching*. Pada penelitian kali ini peneliti akan fokus terhadap bagian *searching*, yaitu mengimplementasikan model *Continuous Bag-of-Words* (CBOW) dan *Continuous Skip-Gram* (*Skip-Gram*) pada sistem pencarian teks.

Peneiltian akan dimulai dengan pengumpulan data. Data yang dikumpulkan akan digunakan untuk melatih sistem dan juga sebagai target pencarian sistem. Sistem akan dilatih untuk menghasilkan data kata relevan dengan model CBOW dan *Skip-Gram*. Setelah sistem selesai di latih, *user* akan melakukan pencarian dokumen pada sistem. *User* akan memasukkan *keyword* pada mesin pencarian. Sistem akan mencari kata terdekat dari *input* kata *user* dengan menggunakan model CBOW dan *Skip-Gram*. Sistem menggunakan gabungan kata relevan dan kata input user untuk mencari dokumen. Dokumen dengan jumlah kemunculan kata input dan kata relevan yang tinggi akan dikeluarkan oleh sistem. Lalu di akhir user akan menilai relevansi dokumen.

Penelitian ini merupakan bagian dari penelitian besar *Search Engine*, hasil ranking dokumen yang di dapat dari penelitian ini merupakan hasil sementara yang pada akhirnya akan digabungkan dengan metode-metode lain untuk mendapatkan hasil ranking dokumen yang lebih baik lagi.

B. Perumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dirumuskan:

1. Bagaimana cara merancang sistem pencarian teks dengan metode *Continuous-Bag-of-Words* dan *Continuous Skip-Gram* ?

2. Apakah pencarian teks dengan menggunakan *Continuous-Bag-of-Words* dan *Continuous Skip-Gram* dapat menghasilkan hasil pencarian yang relevan ?

C. Pembatasan Masalah

Karena keterbatasan waktu, dana, tenaga, teori dan agar penelitian dapat dilakukan lebih mendalam, maka tidak semua masalah akan diteliti. Berikut merupakan batasan-batasan yang diterapkan oleh penulis:

1. Menggunakan kumpulan artikel yang di *crawl* dari situs <https://www.indosp-ort.com>.
2. *Tester* hanya menilai hasil lima dokumen teratas yang dikeluarkan sistem.

D. Tujuan Penelitian

Berdasarkan perumusan masalah di atas, adapun yang menjadi tujuan dari pembuatan skripsi ini adalah :

1. Merancang sistem pencarian teks dengan metode CBOW dan *Skip-Gram*.
2. Memvalidasi sistem pencarian.

E. Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat, antara lain:

1. Bagi penulis, dapat menambah pengetahuan teoritis dan praktikal penulis dalam pengembangan sistem pencarian teks.
2. Bagi Universitas Negeri Jakarta, penelitian ini diharapkan dapat menjadi acuan bagi pengembangan sistem pencarian teks yang serupa.
3. Bagi masyarakat, sebagai acuan pengetahuan metode terhadap sistem pencarian teks.

BAB II

KAJIAN PUSTAKA

Pada bab ini akan di jelaskan hal-hal yang berkaitan dengan penelitian yang akan dijalankan.

A. *Natural Language Processing*

Natural Language Processing (NLP) merupakan cabang dari komputer sains yang membahas tentang kemampuan komputer untuk mengerti teks dan kalimat sama seperti manusia menginterpretasikannya.

NLP menggabungkan *computational linguistics* dengan statistik, *machine learning*, dan *deep learning model*. Teknologi-teknologi tersebut memungkinkan komputer untuk memproses bahasa manusia dalam bentuk teks atau suara dan untuk mengerti maksudnya, lengkap dengan niat dan sentimen pembicara atau penulis.

NLP banyak digunakan di aplikasi *real-world* modern, contohnya *Spam detection*, *Machine Translation*, *Chatbot*, dan *Sentiment Analysis* Media Sosial.

B. *Neural Network*

Neural Network merupakan kumpulan algoritma yang menyerupai otak manusia. *Neural network* terdiri dari node layers, yang terdiri dari *input*, satu atau lebih *hidden layer*, dan *output layer*. Setiap *node* tersambung satu sama lain dan memiliki *weight* diantaranya. *Neural network* bergantung pada *training* data untuk memperbaiki akurasi *neural network* itu sendiri, tetapi setelah *training*, *Neural Network* dapat digunakan untuk mengklasifikasi dan mengkategorikan data dengan kecepatan tinggi.

Neural network terdiri dari beberapa kelas:

1. *Feedforward Neural Network*

Feedforward Neural Network merupakan *neural network* paling sederhana.

Feedforward Neural Network hanya bergerak dalam satu arah melalui *input*, *hidden layer*, dan *output*. Salah satu algoritma yang paling banyak digunakan untuk melatih data *Feedforward Neural Network* adalah *backpropagation*.

Backpropagation. *Backpropagation* merupakan algoritma yang banyak digunakan untuk melatih *Feedforward Neural Network*. Dalam pelatihan data, *backpropagation* menghitung *gradient* dari *loss function* berdasarkan *weight* dari *neural network*, sehingga *weight* bisa di ubah untuk meminimalisasi *loss*.

2. *Convolutional Neural Network*

Neural network ini mirip dengan *feedforward*, tapi biasanya digunakan untuk *image recognition* dan *pattern recognition*. Dalam *neural network* ini, *hidden layer* terdiri dari *layer* yang melakukan konvolusi.

3. *Recurrent Neural Network (RNN)*

Recurrent Neural Network merupakan tipe *neural network* yang menggunakan data sekuensial. *Neural network* ini biasa digunakan untuk terjemahan, *Natural Language Processing (NLP)*, dan *speech recognition*. Jika *neural network* sederhana menganggap *input* dan *output* independen, RNN memiliki *output* yang bergantung pada elemen sebelumnya dalam sekuens.

C. *Query Expansion*

Query Expansion merupakan proses menimbang ulang kueri untuk memperbaiki performa dalam *Information Retrieval*, terutama dalam konteks pemahaman kueri. Dalam *search engine*, *Query Expansion* melakukan evaluasi ulang terhadap *input user* dan memperbesar jangkauan kueri pencarian untuk menghasilkan dokumen yang lebih banyak. Dengan menampilkan dokumen yang lebih banyak diharapkan dokumen yang tidak akan muncul pada hasil, yang memiliki kemungkinan lebih relevan terhadap kueri user, akan masuk dalam hasil pencarian, walaupun relevan atau tidak.

D. Word Embedding

Word Embedding merupakan pendekatan dalam NLP (*Natural Language Processing*), yang digunakan untuk merepresentasikan kata dan dokumen dalam bidang vektor. *Word Embedding* didasarkan pada *Distributional Hypothesis* yang mengatakan bahwa kata-kata dicirikan oleh kata-kata di sekitarnya dan kata yang sering muncul dalam konteks yang sama biasanya memiliki maksud yang mirip. Pendekatan ini telah banyak di adopsi oleh berbagai peneliti setelah kemajuan pada tahun 2010 terjadi tentang teoritikal kerja kualitas vektor, kemajuan kecepatan *training* model, dan kemajuan *hardware* memperbolehkan parameter yang lebih besar di jelajah dengan menguntungkan. Tahun 2013, Mikolov et al. membuat *word2vec*, alat *Word Embedding* yang dapat melatih teks dalam bidang vektor dengan lebih cepat dari pendekatan sebelumnya.

E. Representasi kata dalam bentuk vektor

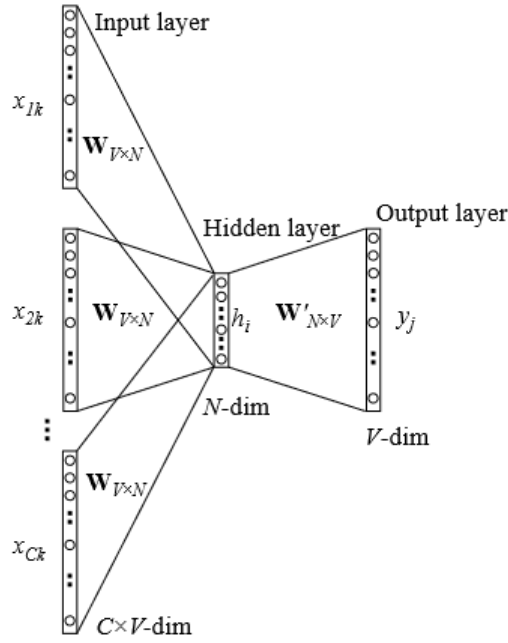
Salah satu model yang paling populer untuk merepresentasikan kata dalam bentuk vektor adalah *Feedforward Neural Net Language Model* (Operationnelle et al., 2001). Model ini terdiri dari *input*, *projection*, *hidden layer*, dan *output layer*. Model ini menjadi kompleks pada perhitungan antara *projection* dan *hidden layer*, karena nilai pada *projection layer*-nya banyak. Untuk mengatasi keterbatasan *Feedforward NNLM*, Mikolov et al. (2010), mengajukan *RNNLM (Recurrent Neural Net Language Model)*. Model ini tidak memiliki *projection layer*, hanya *input*, *hidden* dan *output*. Yang membuat model ini spesial merupakan matriks *recurrent* yang menyambungkan *hidden layer* dengan dirinya sendiri, yang dibatasi jangka waktu tertentu. Hal ini memungkinkan model ini membuat semacam *short term memory*, informasi dari waktu sebelumnya dapat direpresentasikan dengan *hidden layer* yang *ter-update* berdasarkan *input* sekarang dan *hidden layer* di waktu sebelumnya.

F. *Word2Vec*

Mikolov et al. (2013), mengajukan dua model baru untuk mempelajari representasi distribusi kata dengan fokus meminimalisasi kompleksitas komputasi. Dari model model sebelumnya, kompleksitas disebabkan oleh *hidden layer* yang tidak linear di model model tersebut. Walaupun ini yang membuat *neural network* menarik, Mikolov memilih untuk menjelajahi model yang lebih sederhana yang mungkin tidak dapat merepresentasikan data seakurat *neural network* tetapi dapat di latih dengan lebih efisien.

1. *Continuous Bag-of-Words*

Arsitektur pertama yang diajukan mirip dengan *Feedforward NNLM*, dimana non-linear *hidden layer*-nya dihilangkan dan *projection layer*-nya digunakan oleh semua kata (bukan hanya *projection* matriks-nya saja), jadi semua kata terproyeksi ke posisi yang sama (vektornya di rata-ratakan). Arsitektur ini disebut *bag-of-words* model karena urutan kata sebelumnya tidak mempengaruhi proyeksi. Arsitektur ini juga menggunakan kata dari setelahnya. Arsitektur ini memprediksi kata target dengan menggunakan kata konteks di sekitarnya. Berikut akan dijelaskan proses dari model CBOW yang dipaparkan oleh Rong (2014).



Gambar 2.1: Model CBOW

Dalam gambar 1, V menunjukkan ukuran kosakata, dan N menunjukkan ukuran *hidden layer*. *Input* merupakan vektor *one-hot encoded*, yang berarti untuk kata konteks masukan yang diberikan, hanya satu dari V unit, x_1, \dots, x_V , merupakan 1 dan unit lain merupakan 0. Contohnya dalam kalimat "budi sedang belajar matematika", *one-hot encoded* dari kata "sedang" merupakan $\{0 \ 1 \ 0 \ 0\}$.

Weight diantara *input layer* dan *output layer* direpresentasikan dengan $V \times N$ matriks W . Untuk mendapatkan *output* pada *hidden layer*, model CBOW mengambil rata-rata vektor dari kata *input* konteks dan mengalikannya dengan *weight matrix input->hidden*.

$$h = \frac{1}{C} W^T (X_1 + X_2 + \dots + X_C) \quad (1)$$

$$= \frac{1}{C} (V_{w_1} + V_{w_2} + \dots + V_{w_C})^T \quad (2)$$

dimana h merupakan *hidden layer*.

C merupakan jumlah kata konteks.

w_1, \dots, w_C merupakan kata di konteks.

Vw merupakan vektor *input* dari kata w .

Dari *hidden layer* ke *output layer*, ada *weight matrix* berbeda, W' . Dengan *weight* tersebut kita bisa mendapatkan u_j

$$u_j = V'_{w_j}{}^T h \quad (3)$$

V'_{w_j} merupakan kolom ke j dari matriks W' .

Lalu kita menggunakan *softmax* untuk mendapatkan distribusi posterior kata.

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (4)$$

dimana y_j merupakan *output* dari j unit di *output layer*.

Update matrix weight hidden->output

Dengan menggunakan *stochastic gradient descent*, kita mendapatkan persamaan *update weight hidden->output*

$$V'_{w_j}{}^{(\text{new})} = V'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \dots, V. \quad (5)$$

$$e_j = y_j - t_j \quad (6)$$

dimana η merupakan *learning rate*.

e_j merupakan *prediction error* kata ke- j dari *output layer*.

\mathbf{h} merupakan *hidden layer*.

V'_{w_j} merupakan *output* dari vektor w_j .

V_w *input* vektor dan V'_w *output* vektor, mereka merupakan dua representasi vektor berbeda dari kata w .

Update matrix weight input->hidden

$$V_{w_{I,c}}^{(\text{new})} = V_{w_{I,c}}^{(\text{old})} - \frac{1}{C} \cdot \eta \cdot \mathbf{EH}^T \quad \text{for } c = 1, 2, \dots, C. \quad (7)$$

dimana $V_{w_{I,c}}$ merupakan *input* vektor kata ke c di *input* konteks.

η merupakan *learning rate*.

\mathbf{EH} sama dengan :

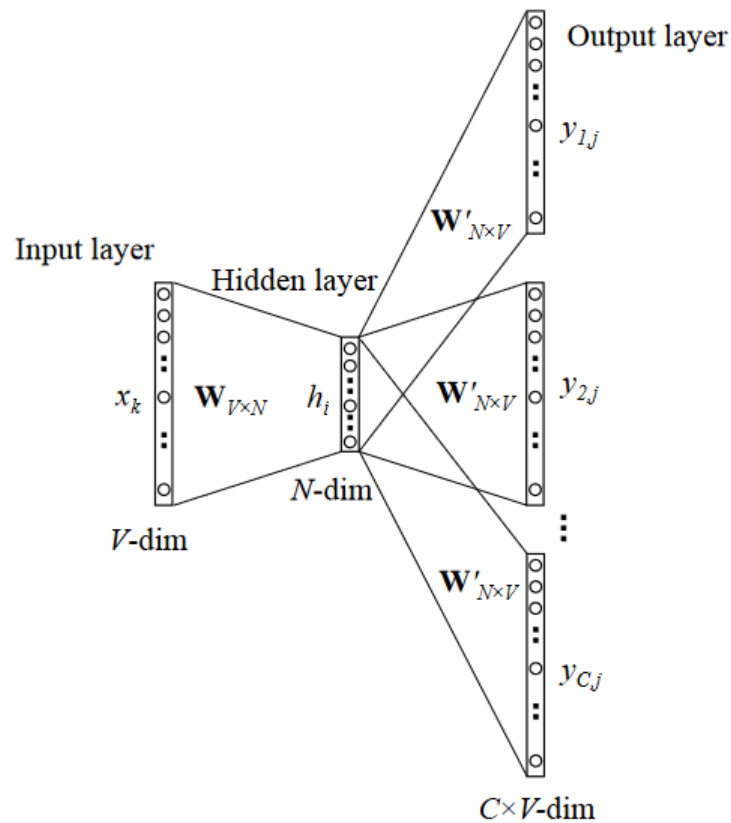
$$\mathbf{EH} = \sum_{j=1}^V e_j \cdot w'_{ij} \quad (8)$$

e_j , sama seperti persamaan (6), merupakan prediksi error kata ke- j di *output layer*.

\mathbf{EH} , N -dimensi vektor merupakan jumlah dari *output* vektor dari semua kata di kosakata yang dikalikan *weight error* prediksi.

2. Continuous Skip-Gram

Arsitektur kedua mirip dengan CBOW, tetapi dimana CBOW memprediksi kata berdasarkan konteks, model ini memprediksi kata konteks berdasarkan kata target yang di masukkan. Berikut akan dijelaskan proses dari model *Skip-Gram* yang dipaparkan oleh Rong (2014).



Gambar 2.2: Model *Skip-Gram*

Model ini, kebalikan dari CBOW, memiliki kata target di *input layer* dan kata konteks di *output layer*. *Hidden layer* memiliki *output* yang sama dengan CBOW, yang berarti h hanya menyalin (dan transpos) baris dari *weight input->hidden matrix*, W , dengan *input* kata w_i

$$\mathbf{h} = \mathbf{v}_{w_I}^T \quad (9)$$

untuk *output layer*, karena *Skip-Gram* memiliki satu *input* kata dan konteks kata yang berbeda, maka hasil *output layer*-nya semua sama

$$u_{c,j} = u_j = \mathbf{v}_{w_j}^T \cdot \mathbf{h} \quad \text{for } c = 1, 2, \dots, C \quad (10)$$

dimana $u_{c,j}$ merupakan hasil keluaran perkalian antara \mathbf{h} , *hidden layer* dan $\mathbf{v}_{w_j}^T$.

$\mathbf{v}_{w_j}^T$ merupakan hasil perkalian *input* vektor dan *input weight* matriks ke- j di panel *output layer* ke- c .

Untuk *output layer* :

$$y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^V \exp(u_{j'})} \quad (11)$$

sama seperti CBOW, *Skip-Gram* menggunakan *softmax* untuk *output layer*.

Dimana $y_{c,j}$ merupakan *output* dari *input* kata ke- j dengan konteks ke- c .

$y_{c,j}$ memiliki hasil *output* yang sama semua karena kata *input*-nya hanya satu.

Masing-masing *output* akan dibandingkan dengan vektor konteks kata C yang sebenarnya, yang akan menghasilkan vektor-vektor yang berbeda, yang pada akhirnya akan dijumlahkan untuk meng-*update weight matrix* model ini.

Update matrix weight hidden->output

$$\mathbf{V}_{w_j}'^{(\text{new})} = \mathbf{V}_{w_j}'^{(\text{old})} - \eta \cdot \text{EI}_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \dots, V. \quad (12)$$

Dimana EI merupakan V -dimensional vektor $EI = EI_1, \dots, EI_V$, yang merupakan total dari $e_{c,j}$.

$e_{c,j}$ merupakan *prediction error* semua kata konteks

$$EI_j = \sum_{c=1}^C e_{c,j} \quad (13)$$

$$e_{c,j} = y_{c,j} - t_{c,j} \quad (14)$$

$e_{c,j}$ merupakan selisih antara probabilitas prediksi dan *true vector* (vektor yang sebenarnya).

True vector merupakan *one-hot encode* kata konteks ke- c .

Update matrix weight input->hidden

$$V_{w_I}^{(\text{new})} = V_{w_I}^{(\text{old})} - \eta \cdot EH^T \quad (15)$$

Dimana EH merupakan N -dimension vektor yang setiap isinya didefinisikan dengan

$$EH_i = \sum_{j=1}^V EI_j \cdot w'_{ij} \quad (16)$$

Persamaan EI sama dengan persamaan (13).

G. Precision

Dalam IR, *precision* merupakan persentase hasil dokumen yang relevan terhadap kueri. *Precision* didapatkan dari jumlah hasil yang relevan dibagi dari jumlah hasil yang didapatkan. *Precision at k* (P@k) merupakan *precision* yang hanya menghitung relevansi terhadap k dokumen teratas.

$$p = \frac{|\{\text{dokumen relevan}\} \cap \{\text{dokumen keluaran}\}|}{|\{\text{dokumen keluaran}\}|} \quad (17)$$

BAB III

METODOLOGI PENELITIAN

A. Metodologi Pengembangan Sistem

Pada subbab ini akan dijelaskan proses yang digunakan untuk melakukan pencarian dengan menggunakan Model CBOW dan *Skip-Gram*. Sistem akan dibuat dalam bentuk terminal dengan bahasa pemrograman Python versi 3.9.1 dan local web server Apache dengan penyimpanan data dalam bentuk database MySQL, dengan menggunakan aplikasi XAMPP.



Gambar 3.1: Proses Sistem

Perancangan sistem akan dimulai dengan pengumpulan data. Data akan dikumpulkan dengan *web crawl* lalu akan dirapihkan dengan menghapus hal-hal yang tidak berkaitan dengan artikel, contohnya teks iklan, rekomendasi judul artikel lain, dan yang lainnya. Data yang sudah dikumpulkan dan diraphikan akan disimpan ke dalam

database. Data yang dikumpulkan merupakan data yang akan dijadikan sebagai target pencarian pada sistem *search engine*.

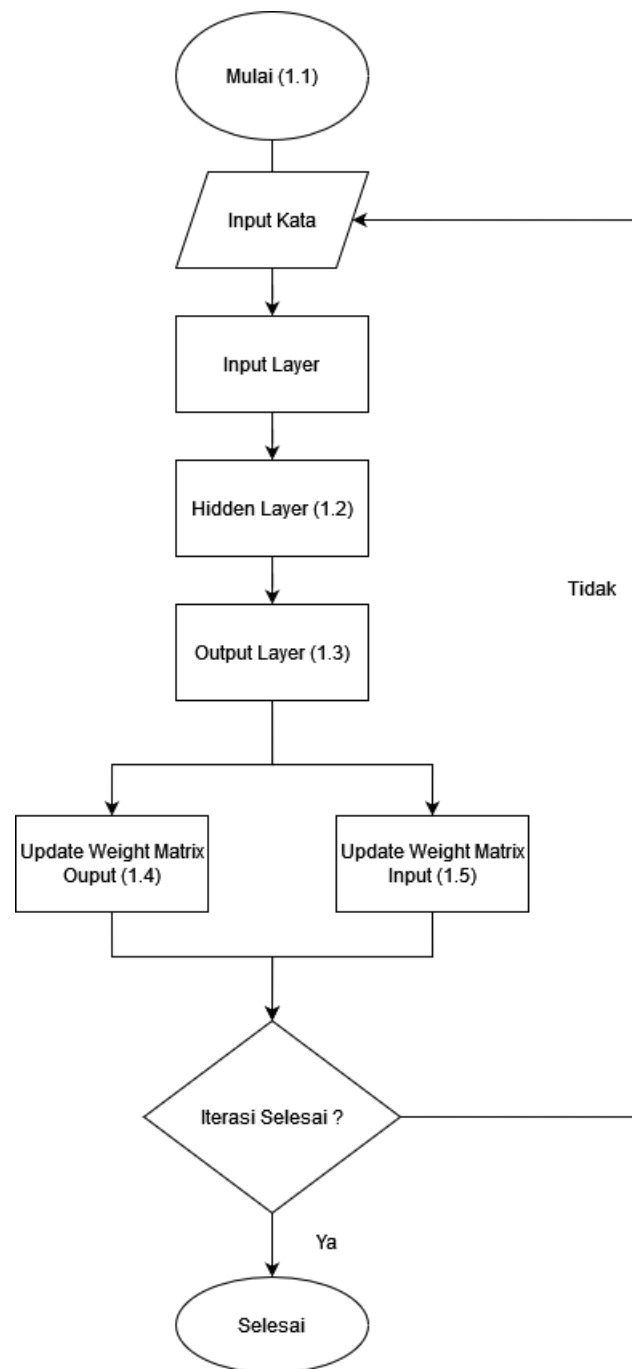
Selanjutnya *Training Sistem*. Sistem akan dilatih dengan menggunakan kumpulan dokumen yang telah dikumpulkan, dengan metode CBOW dan *Skip-Gram*. Masing-masing pelatihan data dilakukan untuk menemukan kata relevan terhadap kata *input* berdasarkan arsitektur model, CBOW untuk menemukan kata target dari kata *input* sekitar, dan *Skip-Gram* untuk menemukan kata konteks sekitar dari kata *input*. Setelah sistem di latih sistem dapat digunakan untuk pencarian.

Proses selanjutnya *Input Kata*, user memasukkan kata *input* pada *search engine* untuk mencari dokumen yang relevan. Kata *input* dibatasi maksimal tiga kata untuk metode *Skip-Gram* dan dua atau empat kata untuk metode CBOW dan metode gabungan *Skip-Gram* dan CBOW. Setelah kata di *input*, sistem akan mencari kata relevan terhadap kata *input* berdasarkan metode yang digunakan. Kata relevan yang ditemukan sistem akan digunakan untuk pencarian dokumen oleh sistem. Dokumen yang memiliki kata *input* dan kata relevan dengan frekuensi kemunculan tinggi akan diberikan ranking tinggi pada *output* pencarian sistem. Lima hasil dokumen dengan ranking tertinggi yang dikeluarkan sistem akan dinilai relevansinya oleh user.

B. Analisis Data

Tahap pertama dalam rancangan eksperimen adalah pengumpulan dataset. Dataset yang penulis pakai merupakan kumpulan artikel yang didapat dari situs : <https://www.indosport.com>, yang di *crawl* dengan menggunakan *tools* yang dirancang oleh Muhammad Fathan Qoriiba pada skripsinya PERANCANGAN CRAWLER SEBAGAI PENDUKUNG PADA SEARCH ENGINE.

C. Proses *Training*

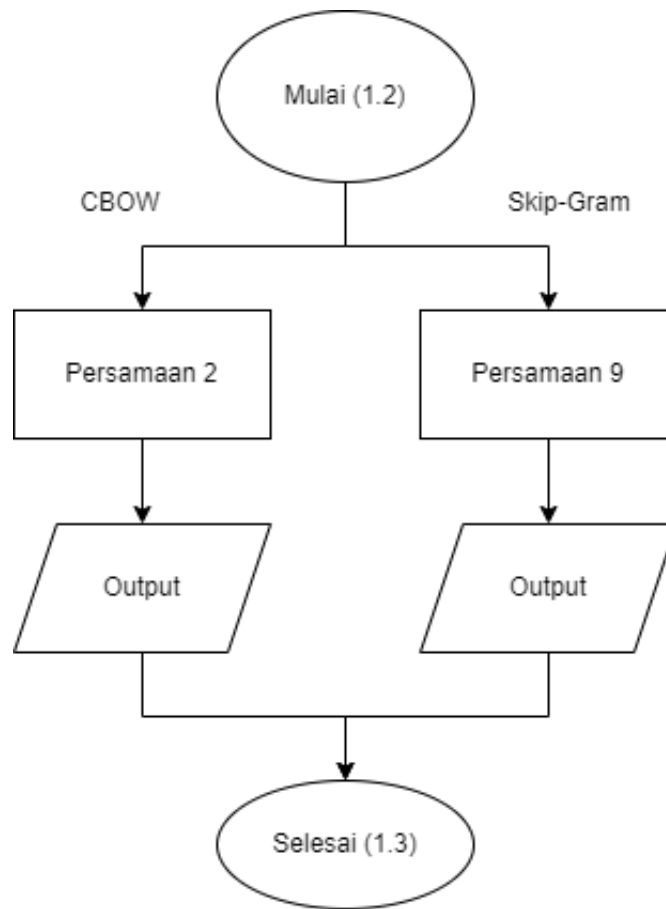


Gambar 3.2: Proses *Training*

Proses *Training* akan dilakukan untuk semua data yang telah dikumpulkan. Data yang telah dikumpulkan akan dibuat *dictionary*, yaitu kumpulan data semua kata unik yang muncul di kumpulan dokumen/artikel tersebut. *Dictionary* akan diubah ke dalam bentuk *one-hot encode* sesuai indeks kata untuk digunakan dalam proses *training* data (contohnya 1 0 0 0 untuk kata di indeks pertama dengan *dictionary* yang berisi empat kata). Proses *training* sepenuhnya menggunakan kata yang direpresentasikan dalam bentuk *one-hot encode*.

Data akan digabung menjadi satu dokumen, lalu *training* akan berjalan dari awal kata hingga akhir kata pada akhir dokumen. *Training* akan dijalankan sesuai model masing-masing, untuk model *Skip-Gram* kata tengah sebagai kata *input* dan kata sekitar sebagai kata konteks, dan untuk CBOW kata sekitar sebagai kata *input* dan kata tengah sebagai kata target. Dalam *input layer* kita hanya akan memasukkan *one-hot encode* kata ke dalam proses *training* yang selanjutnya akan digunakan ke dalam perhitungan masing-masing model, CBOW dan *Skip-Gram*.

1. *Hidden Layer*



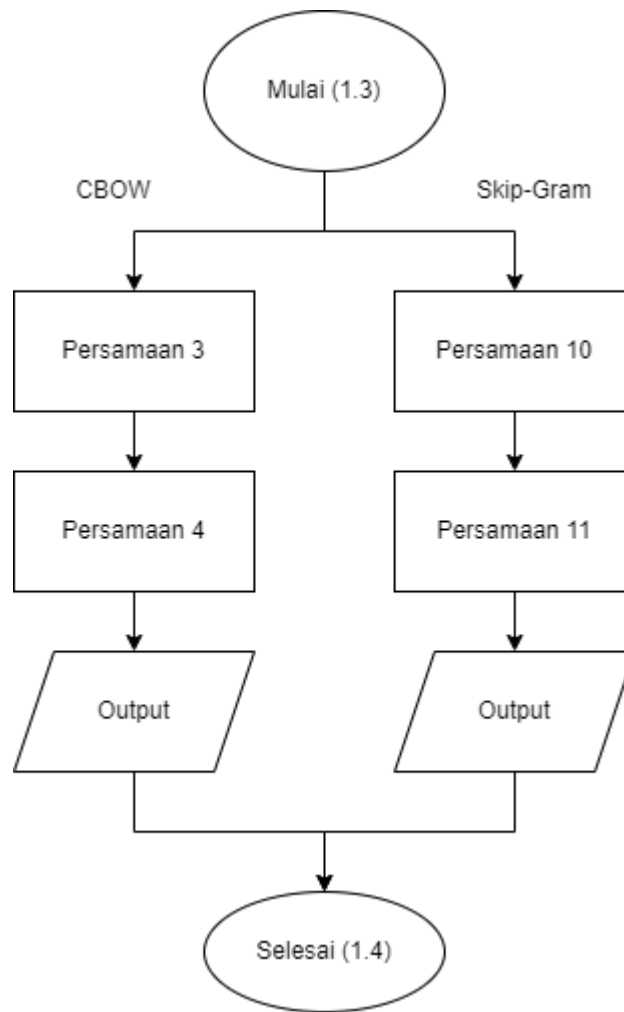
Gambar 3.3: *Hidden Layer*

Dalam *hidden layer*, masing-masing *one-hot encode* kata pada masing-masing model di proyeksi dengan *weight matrix input* yang sebelumnya terlebih dahulu di inisiasi dengan angka acak.

Dalam Model CBOW, kata sekitar (kata konteks) akan dimasukkan ke dalam persamaan (2), yang berarti merata-ratakan vektor kata konteks berdasarkan jumlah kata konteks yang dimasukkan yang lalu akan di transpos.

Dalam *Skip-Gram*, persamaan (9) berarti hanya men-transpos vektor kata *input* yang dimasukkan ke dalam model ini, karena model ini hanya bisa menerima satu *input* kata. Kedua model masing-masing akan menghasilkan *output* satu vektor kata yang akan dimasukkan ke dalam *Output Layer*.

2. Output Layer



Gambar 3.4: *Output Layer*

Hasil vektor kata dari *hidden layer* pada masing-masing model akan dikalikan dengan *weight matrix output* masing-masing yang sebelumnya terdahulu di inisiasi dengan angka acak.

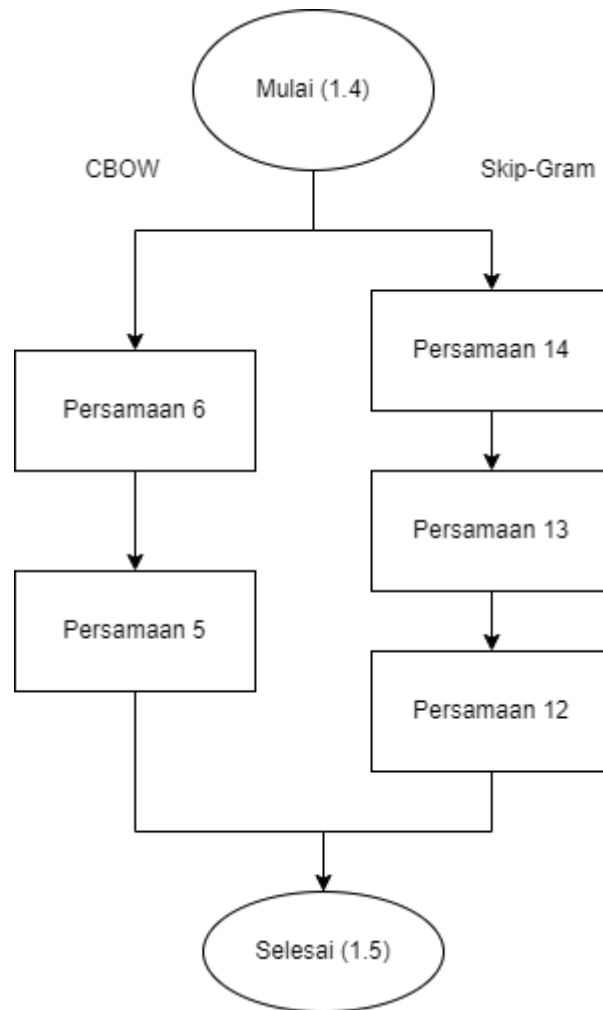
Dalam CBOW, seperti persamaan (3), vektor kata *hidden layer* dikalikan dengan *weight matrix output* yang akan menghasilkan u_j , yang akan digunakan dengan *softmax function* (4), yang hasilnya, y_j merupakan probabilitas vektor kata.

Sama seperti CBOW, persamaan (10) pada *Skip-Gram* berarti perkalian hasil *hidden layer* dengan *weight matrix output*. Karena *Skip-Gram* memiliki banyak konteks kata, $u_{c,j}$ di lakukan untuk semua c , yaitu untuk semua kata konteks, tapi

karena *input* katanya sama untuk banyak konteks kata, hasilnya semua sama, yang berarti $u_{c,j} = u_j$. u_j akan dimasukkan ke dalam *softmax function* yaitu persamaan (11) untuk mendapatkan hasil $y_{c,j}$, yaitu probabilitas vektor kata berdasarkan c konteks kata.

Setelah *training* kata, sistem akan melakukan *backpropagation* untuk memperbaiki *weight matrix* per iterasi untuk mengubah hasil *output layer* agar semakin dekat dengan *true vector*, yaitu *one-hot encode* kata target untuk CBOW dan *one-hot encode* kata konteks untuk *Skip-Gram*.

3. Update Weight Matrix Output

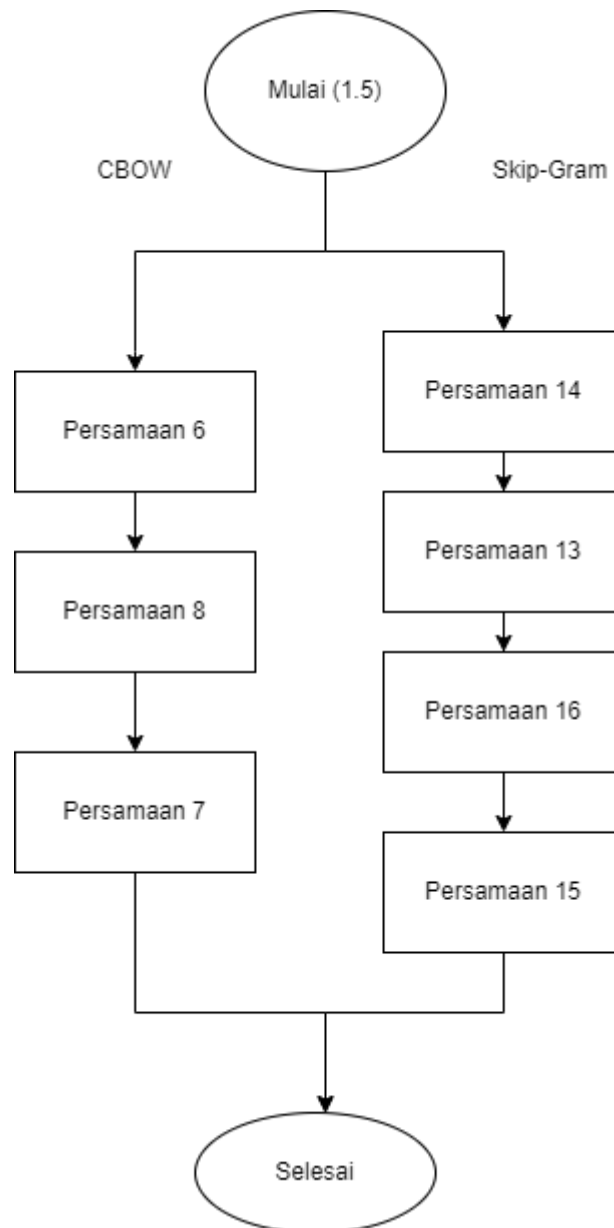


Gambar 3.5: Update Weight Matrix Output

Setelah sistem di latih, hasil dari *output layer* digunakan untuk meng-*update weight matrix output*. Dalam model CBOW kita menggunakan persamaan (6) untuk mendapatkan *error prediction*, dengan mengurangi hasil *output layer* dengan *true vector* kata target. Setelah mendapatkan e_j , kita dapat menggunakan persamaan (5) untuk mendapatkan *weight matrix output* baru.

Sedangkan untuk *Skip-Gram*, karena model ini memiliki konteks kata yang banyak, *error prediction* $e_{c,j}$ didapatkan per- c , perkonteks kata seperti persamaan (14), lalu semua *error prediction* dijumlahkan seperti pada persamaan (13), yang hasilnya akan dimasukkan ke persamaan (12) untuk mendapatkan *weight matrix output* baru model *Skip-Gram*.

4. *Update Weight Matrix Input*



Gambar 3.6: *Update Weight Matrix Input*

Di proses ini kita akan meng-*update weight matrix input*. Model CBOW dimulai dengan persamaan (6), sama seperti *Update Weight Matrix Output*, kita mencari *error prediction* yang akan kita masukkan ke persamaan (8). Persamaan (8) berarti perkalian *error prediction* dengan *matrix weight output* yang hasilnya akan kita masukkan ke dalam persamaan (7). Di persamaan (7), $V_{w_{I,c}}$ berarti vektor konteks

input kata ke-*c*, yang berarti kita akan meng-*update* semua vektor *weight matrix input* yang merepresentasikan konteks kata yang kita masukkan ke model ini.

Untuk model *Skip-Gram*, sama seperti *Update Weight Matrix Output*, kita menggunakan persamaan (14) yang hasilnya akan kita masukkan ke persamaan (13). Hasil dari persamaan ini akan kita lakukan perkalian dengan *matrix weight output* seperti persamaan (16) dan hasilnya akan kita masukkan ke dalam persamaan (15). Mirip seperti CBOW, di model *Skip-Gram* ini kita juga akan meng-*update* vektor *weight matrix input* yang merepresentasikan kata yang kita masukkan ke model ini. Karena dalam model *Skip-Gram* kita hanya memasukkan satu kata, maka vektor yang diubah juga hanya satu vektor, hanya vektor *weight matrix input* yang merepresentasikan kata tersebut.

Weight matrix output dan *weight matrix input* yang sudah kita *update* akan kita simpan ke dalam *database* yang akan digunakan untuk melakukan pencarian kata relevan terhadap kata *input user* pada pencarian *search engine*.

5. Penggunaan Sistem Untuk Melakukan Pencarian

Setelah proses *training* selesai sistem dapat digunakan untuk melakukan pencarian. *User* akan meng-*input* kata kedalam mesin pencarian dan sistem akan memproses data. Kata yang di-*input user* akan diproses mirip dengan cara *training* sistem tetapi dengan menggunakan *weight matrix input* dan *weight matrix output* yang telah disimpan di *database* setelah proses *training* selesai. Proses akan berjalan hingga ke *output layer* yang hasil di *output layer*-nya akan di cek, dimana bagian vektor yang paling mendekati satu merupakan indeks dari kata yang paling relevan terhadap kata *input*. Hal ini bisa didapatkan dari proses *backpropagation* yang berusaha memperbaiki hasil *output layer* sesuai dengan *true vektor* (*one-hot encode*) kata di sekitar kata *input*. Akan dicari beberapa kata relevan dari kata *input user* untuk memperluas hasil pencarian dari *search engine*. Di akhir, kata relevan yang ditemukan sistem tersebut akan digabungkan dengan kata *input user* untuk mencari dokumen yang relevan.

6. Perankingan Dokumen

Perangkingan hasil pencarian akan dilakukan dengan mencari jumlah frekuensi kemunculan kata pada dokumen dengan menggunakan kata *input user* dan kata relevan yang telah ditemukan sistem. Kata akan diberikan bobot, untuk kata *input* akan diberikan bobot paling tinggi, sedangkan kata relevan akan diberikan bobot sesuai dengan tingkat relevansi kata tersebut terhadap kata *input*. Semakin tinggi relevansi kata relevan terhadap kata *input* maka semakin tinggi bobot kata relevan. Dokumen yang memiliki kemunculan kata *input* dan kata relevan dengan bobot tinggi dan dengan frekuensi yang tinggi akan diberikan ranking tinggi. Berikut akan dijelaskan cara mendapatkan ranking dokumen :

$$D = \sum_{i=1}^n (W_i * Fk_i) \quad (18)$$

Dimana D merupakan ranking dokumen.

n merupakan jumlah kata relevan keluaran model CBOW atau *Skip-Gram*

W merupakan bobot kata ke-*i* hasil kata dari model CBOW atau *Skip-Gram* yang diberikan urutan prioritas sebagai berikut :

- Kata input
- Kata hasil pengembalian CBOW
- Kata hasil pengembalian *Skip-Gram*
- Dalam kasus pengembalian CBOW dan *Skip-Gram* multi kata, prioritas diberikan kepada kata pertama dan seterusnya

Fk_i merupakan jumlah frekuensi kata ke-*i* pada dokumen

7. Data Keluaran dan Evaluasi Sistem

Data yang dihasilkan dari mesin pencarian merupakan dokumen yang diharapkan relevan dengan pencarian *user*. Sistem pencarian akan mengeluarkan lima hasil

dokumen. *User* akan menilai apakah dokumen yang dikeluarkan sistem relevan atau tidak. Penilaian akan dilakukan dengan menggunakan *precision* pada lima dokumen teratas ($P@5$). Dokumen yang ditemukan relevan terhadap pencarian oleh *user* akan dihitung dan dibagi dengan jumlah dokumen yang dikeluarkan sistem pada k (dibagi lima). Semakin tinggi nilai *precision* berarti semakin besar kemungkinan sistem berjalan dengan baik.

D. Rancangan Eksperimen

1. Skenario pertama (Menggunakan metode *Skip-Gram*)

- *Tester* memasukkan kata *input* yang sama ke dalam *search engine*
- Sistem mencari kata dengan kemiripan tinggi (kata konteks) dari kata *input user*
- Sistem melakukan pencarian dokumen terhadap kata *input* dan kata konteks yang terkait
- Perankingan dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input*
- Perankingan juga ditemukan dengan menggunakan frekuensi kata konteks yang berbobot (semakin dekat kata konteks dengan kata *input*, maka bobot kata akan semakin tinggi) yang muncul
- Sistem mengembalikan ranking dokumen berdasarkan hasil tersebut
- *Tester* menilai relevansi yang dihasilkan oleh sistem

2. Skenario kedua (Menggunakan metode *Continuous Bag-of-Words*)

- *Tester* memasukkan kata *input* yang sama ke dalam *search engine*
- Sistem mencari kata dengan kemiripan tinggi (kata target) dari kata *input user*
- Sistem melakukan pencarian dokumen terhadap kata *input* dan kata target yang terkait

- Perankingan dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input*
- Perankingan juga ditemukan dengan menggunakan frekuensi kata target yang berbobot (kata target yang memiliki kemiripan tinggi dengan kata *input* akan diberikan bobot yang tinggi) yang muncul
- Sistem mengembalikan ranking dokumen berdasarkan hasil tersebut
- *Tester* menilai relevansi yang dihasilkan oleh sistem

3. Skenario ketiga (Menggunakan gabungan metode *Skip-Gram* dan *Continuous Bag-of-Words*)

- *Tester* memasukkan kata *input* yang sama ke dalam *search engine*
- Sistem mencari kata konteks dan kata target dengan kemiripan tinggi dari kata *input user*
- Sistem melakukan pencarian dokumen terhadap kata *input*, kata target, dan kata konteks terkait
- Perankingan dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input*
- Perankingan juga ditemukan dengan menggunakan frekuensi kata konteks berbobot yang muncul
- Perankingan juga ditemukan dengan menggunakan frekuensi kata target berbobot yang muncul
- Sistem mengembalikan ranking dokumen berdasarkan hasil tersebut
- *Tester* menilai relevansi yang dihasilkan oleh sistem

DAFTAR PUSTAKA

- Al-Saqq, S., & Awajan, A. (2019). The use of word2vec model in sentiment analysis: A survey. In *Proceedings of the 2019 international conference on artificial intelligence, robotics and control* (p. 39–43). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/3388218.3388229> doi: 10.1145/3388218.3388229
- Halavais, A. (2017). *Search engine society*. John Wiley & Sons.
- Liu, Q., Huang, H., Lut, J., Gao, Y., & Zhang, G. (2017). Enhanced word embedding similarity measures using fuzzy rules for query expansion. In *2017 IEEE international conference on fuzzy systems (fuzz-IEEE)* (p. 1-6). doi: 10.1109/FUZZ-IEEE.2017.8015482
- Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1), 100–103.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, 01). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR, 2013*.
- Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, 01). Recurrent neural network based language model. In (Vol. 2, p. 1045-1048).
- Operationnelle, D., Bengio, Y., Ducharme, R., Vincent, P., & Mathématiques, C. (2001, 10). A neural probabilistic language model.
- Rong, X. (2014). *word2vec parameter learning explained*. arXiv. Retrieved from <https://arxiv.org/abs/1411.2738> doi: 10.48550/ARXIV.1411.2738
- Vechtomova, O., & Wang, Y. (2006). A study of the effect of term proximity on query expansion. *Journal of Information Science*, 32(4), 324-333. Retrieved from <https://doi.org/10.1177/0165551506065787> doi: 10.1177/0165551506065787