

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

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**

2023

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

RANCANG BANGUN SISTEM PENCARIAN TEKS

DENGAN MENGGUNAKAN MODEL

CONTINUOUS-BAG-OF-WORDS DAN MODEL

CONTINUOUS SKIP-GRAM PADA KOLEKSI

DOKUMEN

Nama : Muhammad Zalghornain

No. Registrasi : 3145153000

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: <u>Prof. Dr. Muktiningsih N, M.Si..</u>
	NIP. 196405111989032001		
Wakil Penanggung Jawab			
Wakil Dekan I	: <u>Dr. Esmar Budi, S.Si., MT.</u>
	NIP. 197207281999031002		
Ketua	: <u>Drs. Mulyono, M.Kom.</u>	17-2-23
	NIP. 196605171994031003		
Sekretaris	: <u>Ibnu Hadi, M.Si</u>	17-2-23
	NIP. 198107182008011017		
Penguji	: <u>Med Irzal, M.Kom.</u>	17-2-23
	NIP. 197706152003121001		
Pembimbing I	: <u>Ria Arafiyah, M.Si</u>	20-2-23
	NIP. 197511212005012004		
Pembimbing II	: <u>Muhammad Eka Suryana, M.Kom</u>	20-2-23
	NIP. 19851223201211002		

Dinyatakan lulus ujian skripsi tanggal: 14 Februari 2023

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul "**Rancang Bangun Sistem Pencarian Teks dengan Menggunakan Model *Continuous-Bag-of-Words* dan Model *Continuous Skip-Gram* pada Koleksi Dokumen**" yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Sumber informasi yang diperoleh dari penulis lain yang telah dipublikasikan yang disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika di kemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian-bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Jakarta, 28 Februari 2023

Muhammad Zalghornain

KATA PENGANTAR

Segala puji dan syukur kepada Allah Subhanahu wa ta'ala, karena telah memberikan berkah dan rahmat-Nya sehingga penulis dapat menyelesaikan penyusunan tugas akhir ini dengan judul “Rancang Bangun Sistem Pencarian Tekst dengan Menggunakan Model *Continuous-Bag-of-Words* dan Model *Continuous Skip-Gram* pada Koleksi Dokumen”. Keberhasilan dalam menyusun dan menyelesaikan tugas akhir ini tidak lepas dari bantuan berbagai pihak yang memberikan dukungan moril dan materil. Oleh karena itu, pada kesempatan ini dengan kerendahan hati penulis ingin mengucapkan terima kasih kepada:

1. Ibu Ir. Fariani Hermin Indiyah, M.T selaku Koordinator Program Studi Ilmu Komputer yang selalu mendorong mahasiswa untuk menyelesaikan Tugas Akhir.
2. Ibu Ria Arafiah, M.Si selaku dosen pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran yang baik untuk penulis dalam menyelesaikan Tugas Akhir ini.
3. Bapak Muhammad Eka Suryana, M.Kom. selaku dosen pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran yang baik untuk penulis dalam menyelesaikan Tugas Akhir ini.
4. Bapak Med Irzal, M.Kom., selaku dosen pembimbing akademis penulis yang senantiasa membantu mahasiswa.
5. Para dosen dan teman-teman Program Studi Ilmu Komputer angkatan 2015 yang selalu membantu dan mendukung sehingga Tugas Akhir ini dapat diselesaikan dengan baik.
6. Kedua orang tua dan kakak penulis yang senantiasa memberikan dukungan untuk menyelesaikan Tugas Akhir.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna, oleh karena itu, Penulis dengan senang hati menerima kritik dan saran yang bersifat membangun dari semua pihak sehingga tugas akhir ini dapat disusun menjadi lebih baik lagi. Akhir kata harapan Penulis agar tugas akhir ini dapat berguna sebagai sumber bacaan bagi seluruh kalangan sehingga dapat bermanfaat bagi lingkungan akademik maupun pihak lain yang membutuhkannya. Atas perhatiannya, Penulis mengucapkan terima kasih.

Jakarta, 6 Februari 2023

Penulis

ABSTRAK

MUHAMMAD ZALGHORNAIN. Rancang Bangun Sistem Pencarian Teks dengan Menggunakan Model *Continuous-Bag-of-Words* dan Model *Continuous Skip-Gram* pada Koleksi Dokumen. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2023. Di bawah bimbingan Ria Arafiyah, M.Si dan Muhammad Eka Suryana, M.Kom.

Dengan banyaknya data yang dibutuhkan, diperlukan suatu alat untuk mensortir data yang diinginkan secara cepat dan mudah. *Search engine* merupakan salah satu cara untuk mensortir data sesuai keinginan yang dibutuhkan *user* berdasarkan kata yang di-*input user* secara cepat dan mudah. Sistem *search engine* tradisional hanya menggunakan jumlah frekuensi kata pada dokumen untuk mencari kata yang relevan. Diperlukan cara untuk mengerti kueri *user* agar bisa didapatkan hasil pencarian yang sesuai keinginan *user* terlepas keterbatasan kueri *user*. Akan digunakan metode *Continuous-Bag-of-Words* dan *Continuous Skip-Gram* untuk mencari kata di sekitar kata yang di-*input user* yang lalu akan digunakan untuk melakukan pencarian dokumen. Hasil menunjukkan bahwa *neutral network* tidak cocok digunakan untuk mesin pencarian karena waktu *training*-nya yang lama untuk mendapatkan hasil yang diinginkan. Sedangkan untuk hasil relevansinya, metode *Continuous-Bag-of-Words* dapat menghasilkan hasil relevan dengan hasil kesesuaian 86.67% terhadap hasil *ranking user*.

Kata kunci : *Continuous-Bag-of-Words*, *Continuous-Skip-Gram*, pencarian, pencarian teks, *ranking* dokumen, relevansi dokumen.

ABSTRACT

MUHAMMAD ZALGHORNAIN. Building a Text Search System using Continuous-Bag-of-Words Model and Continuous Skip-Gram Model on a Collection of Documents. Thesis. Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2023. Supervised by Ria Arafiyah, M.Si and Muhammad Eka Suryana, M.Kom.

With so much data that is needed, there is a need for some tools to sort all the data that we want quickly and easy. Search engine is one of the tools that is used to sort data quickly and easy that the user want according to the words that he input in. Traditional search engine system usually only used words frequency on the document to find the relevant document. There is a need to understand users query so users could get result they want regardless of users query limitations. Continuous-Bag-of-Words models and Continuous Skip-Gram models will be used to search for the words around the words that user input to search for the document. The results shows that neural network isn't exactly good to be used for search engine because of it's training time that could take a while to get a good results. As for the relevancy, Continuous-Bag-of-Words models can get a relevant results with accuracy of 86.67% compared to user ranking results.

Keywords : *Continuous-Bag-of-Words, Continuous-Skip-Gram, document ranking, document relevancy, searching, text searching.*

DAFTAR ISI

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI	i
LEMBAR PERNYATAAN	ii
KATA PENGANTAR	iii
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR GAMBAR	ix
DAFTAR TABEL	x
BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Perumusan Masalah	4
C. Pembatasan Masalah	4
D. Tujuan Penelitian	4
E. Manfaat Penelitian	4
BAB II KAJIAN PUSTAKA	6
A. <i>Natural Language Processing</i>	6
B. <i>Neural Network</i>	6
C. <i>Query Expansion</i>	7
D. <i>Word Embedding</i>	8
E. Representasi kata dalam bentuk vektor	8
F. <i>Word2Vec</i>	9
G. <i>Euclidean Distance</i>	15
H. <i>Hit Rate</i>	16
BAB III METODOLOGI PENELITIAN	17
A. Metodologi Pengembangan Sistem	17
B. Analisis Data	19
C. Proses <i>Training</i>	20
1. <i>Hidden Layer</i>	22
2. <i>Output Layer</i>	23
3. <i>Update Weight Matrix Output</i>	25
4. <i>Update Weight Matrix Input</i>	26
5. Penggunaan Sistem Untuk Melakukan Pencarian	27
6. Pe-ranking-an Dokumen	28
7. Data Keluaran dan Evaluasi Sistem	28
D. Rancangan Eksperimen	29
BAB IV HASIL DAN PEMBAHASAN	31
A. Data dan <i>Training Parameter</i>	31

B.	Pembersihan Data	33
C.	<i>Training Data</i>	35
D.	Pencarian Kata Pada Dokumen	39
E.	Pengetesan <i>User</i>	41
F.	Analisis Hasil	42
BAB V	KESIMPULAN DAN SARAN	64
A.	Kesimpulan	64
B.	Saran	65
	DAFTAR PUSTAKA	66
	LAMPIRAN	67
	RIWAYAT HIDUP	108

DAFTAR GAMBAR

Gambar 2.1 : Model CBOW	10
Gambar 2.2 : Model <i>Skip-Gram</i>	13
Gambar 3.1 : Proses Sistem	18
Gambar 3.2 : Proses <i>Training</i>	20
Gambar 3.3 : <i>Hidden Layer</i>	22
Gambar 3.4 : <i>Output Layer</i>	23
Gambar 3.5 : <i>Update Weight Matrix Output</i>	25
Gambar 3.6 : <i>Update Weight Matrix Input</i>	26
Gambar 4.1 : Struktur <i>Database</i>	31
Gambar 4.2 : Pembersihan Data	34
Gambar 4.3 : Contoh <i>List Kata</i> dan <i>One-Hot-Encode</i>	35
Gambar 4.4 : Inisiasi <i>Weight</i> Secara <i>Random</i>	36
Gambar 4.5 : Mencari <i>Softmax</i> Lalu Mengurangi Hasilnya dengan Angka Sebenarnya	37
Gambar 4.6 : <i>Weight Matrix</i> Setelah <i>Training</i>	38
Gambar 4.7 : Program Pencarian Berbasis Terminal	39
Gambar 4.8 : Pencarian Kata Tengah pada CBOW dengan Mencari Indeks Angka Paling Besar pada Hasil <i>Softmax</i>	40
Gambar 4.9 : Pencarian Kata pada Semua Dokumen Menggunakan <i>Regular Expression</i>	40
Gambar 4.10 : Contoh Hasil Pencarian Dokumen	41
Gambar 4.11 : Contoh Hasil Pencarian Dokumen oleh <i>User</i>	42
Gambar 4.12 : Jarak Antara Prediksi dengan Hasil Sebenarnya <i>Skip-Gram</i>	43
Gambar 4.13 : Jarak Antara Prediksi dengan Hasil Sebenarnya CBOW Dua Kata	43
Gambar 4.14 : Jarak Antara Prediksi dengan Hasil Sebenarnya CBOW Empat Kata	44

DAFTAR TABEL

Tabel 4.1 :	Pencarian Pertama CBOW dengan Menggunakan Kata “hasil” dan “italia”	45
Tabel 4.2 :	Pencarian Kedua CBOW dengan Menggunakan Kata “pemain” dan “milan”	47
Tabel 4.3 :	Pencarian Ketiga CBOW dengan Menggunakan Kata “juara” dan “dunia”	49
Tabel 4.4 :	Pencarian Keempat CBOW dengan Menggunakan Kata “olahraga” dan “tinju”	51
Tabel 4.5 :	Pencarian Kelima CBOW dengan Menggunakan Kata “striker” dan “italia”	53
Tabel 4.6 :	Pencarian Pertama CBOW dengan Menggunakan Kata “pemain”, “kunci”, “juara”, dan “piala”	55
Tabel 4.7 :	Pencarian Kedua CBOW dengan Menggunakan Kata “chelsea”, “lepas”, “abraham”, dan “arsenal”	57
Tabel 4.8 :	Pencarian Ketiga CBOW dengan Menggunakan Kata “cristiano”, “ronaldo”, “manchester”, dan “united”	59
Tabel 4.9 :	Pencarian Keempat CBOW dengan Menggunakan Kata “asam”, “lambung”, “olahraga”, dan “cocok”	60
Tabel 4.10 :	Pencarian Kelima CBOW dengan Menggunakan Kata “inter”, “milan”, “posisi”, dan “klasemen”	62

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 sistematik untuk menemukan informasi spesifik yang di *input*. Hasil pencarian biasanya di tampilkan dalam bentuk *list*. Infomasi 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.

Pe-ranking-an 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 pe-ranking-an 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.indosport.com>.
2. Teks yang digunakan merupakan teks bertema olahraga.
3. *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 dijelaskan 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, cotohnya *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 menambah 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 jangkau 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 adoptasi 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

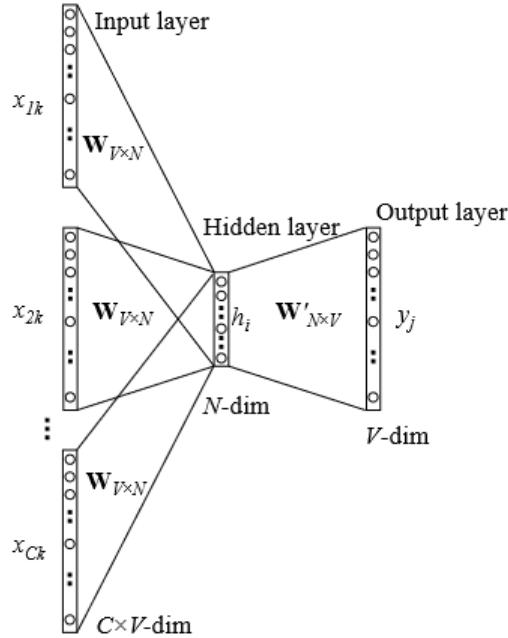
recurrent yang menyambungkan *hidden layer* dengan dirinya sendiri, yang dibatasi jangka waktu tertentu. Hal ini memungkinkan model ini membuat semacam *short term memory*, informasi dari waktu sebelumnya dapat direpresentasikan dengan *hidden layer* yang ter-update berdasarkan *input* sekarang dan *hidden 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 terprojeksi ke posisi yang sama (vektornya di rata-ratakan). Arsitektur ini disebut *bag-of-words* model karena urutan kata sebelumnya tidak mempengaruhi projeksi. 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

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

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

η merupakan *learning rate*.

$\mathbf{E}\mathbf{H}$ sama dengan :

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

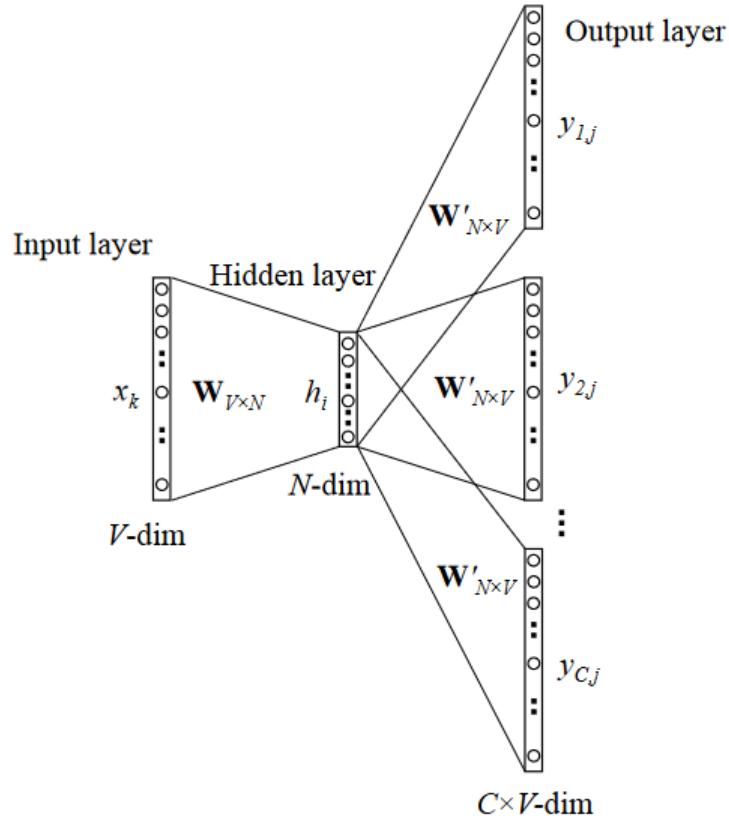
e_j , sama seperti persamaan (6), merupakan prediksi error kata ke- j di *output layer*.

$\mathbf{E}\mathbf{H}$, 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

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

Dimana \mathbf{EI} merupakan V -dimensional vektor $\mathbf{EI} = \mathbf{EI}_1, \dots, \mathbf{EI}_V$, yang merupakan total dari $e_{c,j}$.

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

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

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

$e_{c,j}$ 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

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

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

$$\text{EH}_i = \sum_{j=1}^V \text{EI}_j \cdot w'_{ij} \quad (16)$$

Persamaan EI sama dengan persamaan (13).

G. Euclidean Distance

Euclidean distance merupakan rumus untuk mengukur jarak antara dua vektor. Nilai yang semakin kecil berarti vektor semakin dekat. Euclidean distance akan digunakan untuk mengukur jarak antara hasil prediksi dengan vektor yang sebenarnya.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} \quad (17)$$

dimana p merupakan vektor pertama dan q merupakan vektor kedua

H. *Hit Rate*

Hit Rate merupakan pembagian dari *user* yang dimana jawaban benarnya termasuk dalam daftar rekomendasi terhadap total *user*. *Hit Rate* akan digunakan untuk mengukur kesesuaian sistem pencarian terhadap *user*.

$$HR = \frac{|U_{hit}^L|}{|U^{all}|} \quad (18)$$

dimana U_{hit}^L merupakan jumlah *user* yang memberikan jawaban yang benar dalam top L rekomendasi.

U^{all} merupakan total *user* dalam tes.

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 Phyton 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*.

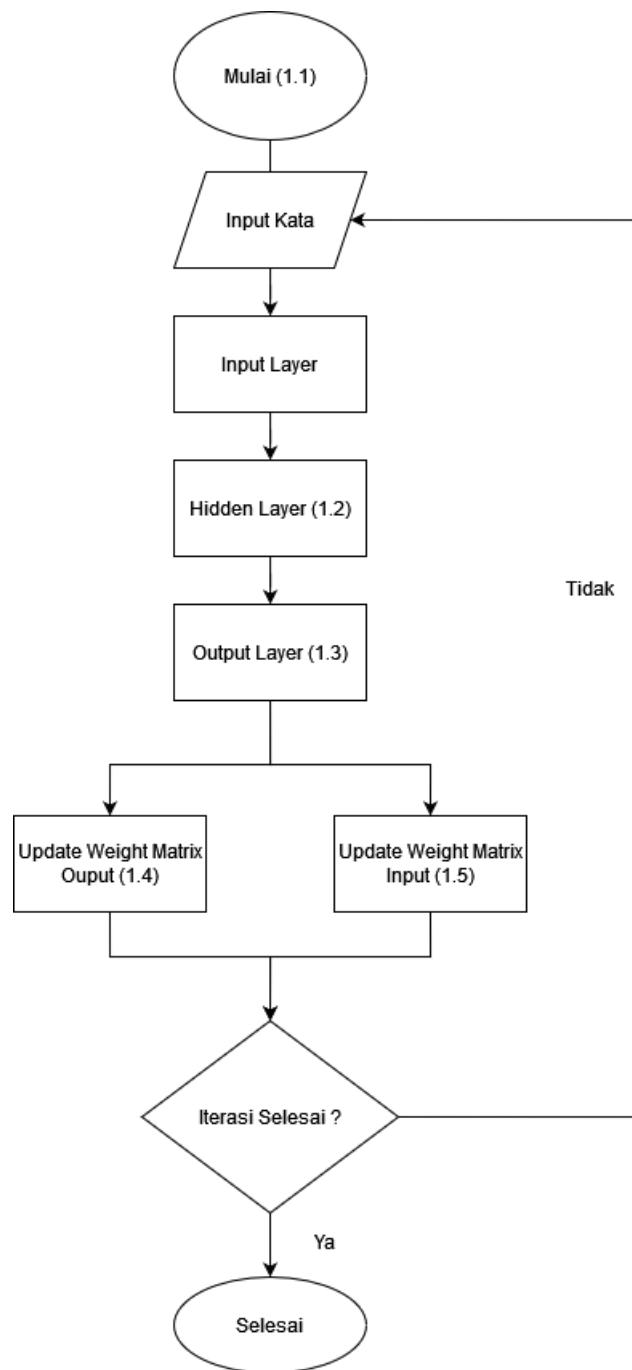
Selanjutnya *Training* Sistem. Sistem akan dilatih dengan menggunakan kumpulan dokumen yang telah dikumpulkan, dengan metode CBOW dan *Skip-Gram*. Masing-masing pelatihan data dilakukan untuk menemukan kata relevan terhadap kata *input* berdasarkan arsitektur model, CBOW untuk menemukan kata target dari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



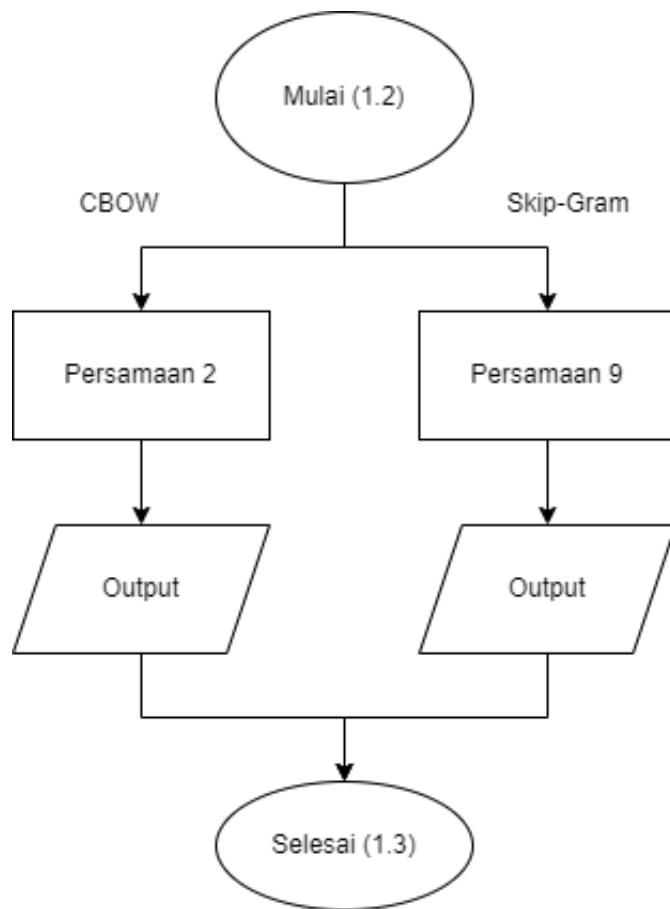
Gambar 3.2 : Proses *Training*

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*

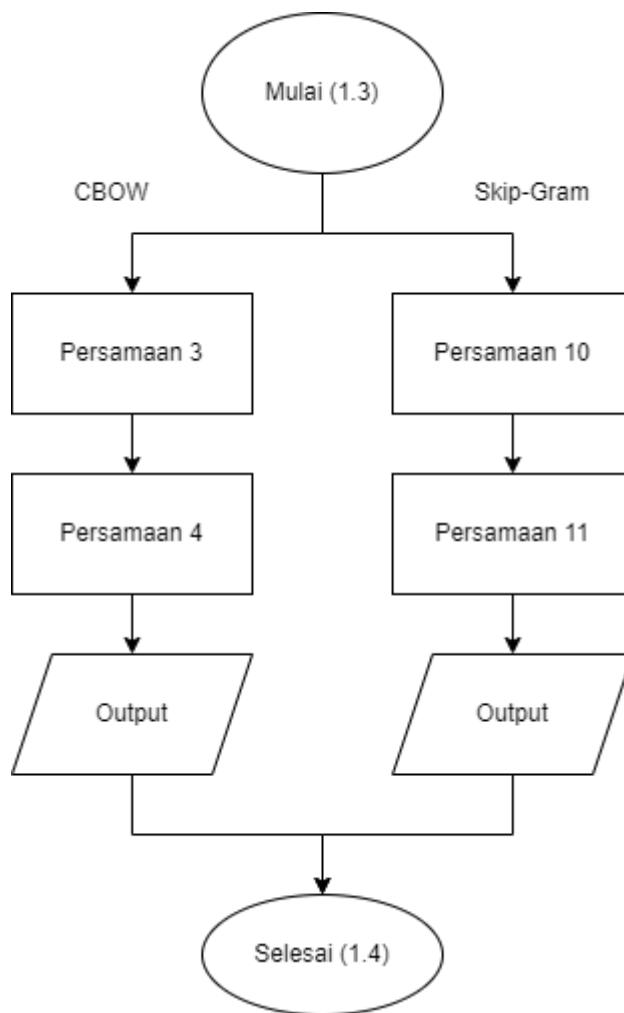
Dalam *hidden layer*, masing-masing *one-hot encode* kata pada masing-masing model di projeksi dengan *weight matrix input* yang sebelumnya terlebih dahulu diinisiasi dengan angka acak.

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*

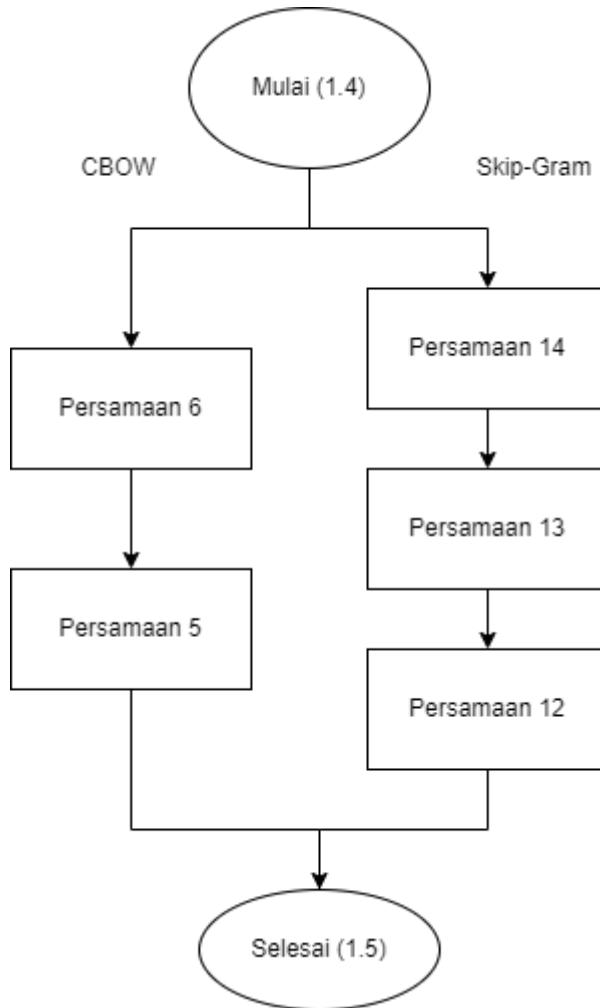
Hasil vektor kata dari *hidden layer* pada masing-masing model akan dikalikan dengan *weight matrix output* masing-masing yang sebelumnya terdahulu di inisiasi dengan angka acak.

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



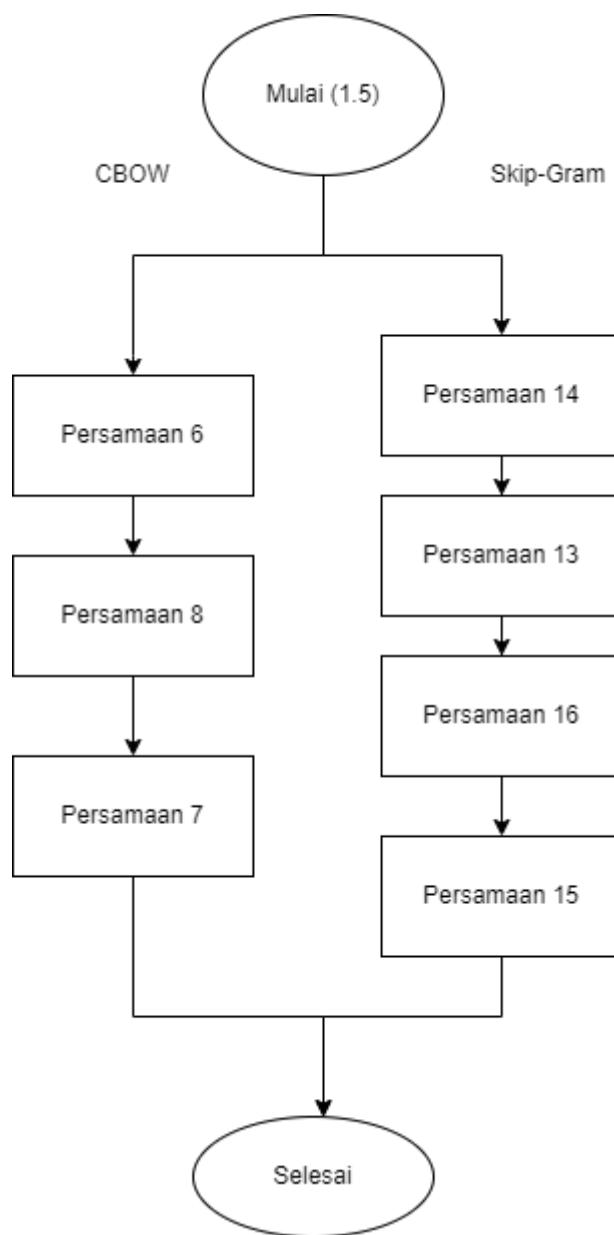
Gambar 3.5 : *Update Weight Matrix Output*

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), $Vw_{I,c}$ 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. Pe-ranking-an Dokumen

Pe-ranking-an hasil pencarian akan dilakukan dengan mencari jumlah frekuensi kemunculan kata pada dokumen dengan menggunakan kata *input user* dan kata relevan yang telah ditemukan sistem. Pada pencarian dengan menggunakan CBOW, dua kata *input* yang dimasukkan *user* akan dicari kata tengahnya, lalu tiga kesatuan kata tersebut akan digabungkan lalu dicari di semua dokumen. Begitu juga dengan Skip-Gram, kata yang di *input user* akan digabungkan dengan dua kata keluaran sistem lalu akan digabungkan menjadi satu kesatuan lalu dicari di seluruh dokumen. Karena Skip-Gram dan CBOW hanya dapat memprediksi kemungkinan kata yang keluar berdasarkan modelnya tanpa mengetahui tetangga yang mana yang dibagian kiri atau kanan, maka pencarian dimodifikasi dengan mencoba dua kombinasi. Untuk CBOW kata satu dan dua yang di *input user* juga akan dicari dengan cara dibalik. Contohnya kata *input* saya dan bermain dengan kata tengah suka akan dicari saya suka bermain dan bermain suka saya, sedangkan Skip-Gram dua kata keluarannya yang akan dibalik, contohnya kata *input* suka dengan keluaran saya dan bermain akan dicari saya suka bermain dan bermain suka saya.

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 diminta untuk memberi *ranking* pada lima hasil dokumen tersebut terhadap kesesuaianya dengan kata *input*. Lalu penulis di akhir akan menilai kesesuaian *ranking* dokumen keluaran sistem terhadap *ranking* dokumen

yang diberikan *user*.

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
- Pe-ranking-an dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input* dan kata konteks
- 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
- Pe-ranking-an dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input* dan kata target
- 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
- Pe-ranking-an dilakukan terhadap dokumen dengan menghitung frekuensi kemunculan kata *input* dan kata target
- Pe-ranking-an juga ditemukan dengan menggunakan frekuensi kata *input* dan kata konteks
- Sistem mengembalikan *ranking* dokumen berdasarkan hasil tersebut
- *Tester* menilai relevansi yang dihasilkan oleh sistem

BAB IV

HASIL DAN PEMBAHASAN

A. Data dan *Training Parameter*

Data yang digunakan merupakan kumpulan artikel yang di *crawl* dari situs <https://www.indosport.com>. Data yang digunakan dibatasi sejumlah 70 artikel (70 *link*) yang per-artikelnya berisi teks yang berada pada *link* artikel. Data lalu akan dirapihkan dengan menghapus simbol-simbol spesial, contohnya strip (-), koma (,), petik ('), dll. Data per-artikel yang sudah dirapihkan akan digabungkan menjadi satu data besar, yaitu data yang berisi semua teks pada semua artikel. Data ini akan disimpan ke dalam *database* dan akan digunakan untuk *training sistem*. Berikut merupakan struktur *database*.

skripsi weight
id : int(11)
metode : longtext
matrix_weight_input : longtext
matrix_weight_output : longtext
kata_unik : int(11)
hidden_layer : int(11)

skripsi text
id : int(255)
sumber_url : varchar(255)
content : longtext

skripsi dictionary
id : int(11)
kata : varchar(255)
one_hot_encode : text
vector_skip_gram : text
vector_cbow : longtext
vector_cbow_4kata : longtext

skripsi big_data
id : int(11)
compiled_string : longtext

Gambar 4.1 : Struktur *Database*

Tabel *dictionary* merupakan tabel tempat menyimpan semua kata unik, kata unik disimpan pada kolom kata, *one-hot-encode* kata disimpan pada kolom *one_hot_encode*, dan vektor-vektor dari kata disimpan ke dalam kolom yang sesuai, vektor *Skip-Gram* pada *vector_skip_gram*, vektor CBOW dua kata pada *vector_cbow*, dan vektor CBOW empat kata pada *vector_cbow_4kata*. Tabel *big_data* berisi kolom *compiled_string*, dimana kolom diisi oleh data gabungan yang sudah dirapihkan dari semua isi dokumen. Data ini akan digunakan untuk

training sistem. Tabel *text* berisi kolom *sumber_url* dan *content*, *sumber_url* berisi semua *link* dari data yang di *crawl* dan *content* merupakan isi konten dari *link* tersebut. Tabel *weight* berisi nama metode, *matrix_weight_input*, *matrix_weight_output*, *kata_unik*, dan *hidden_layer*. Tabel ini akan digunakan untuk *searching* dokumen, dimana *matrix weight* yang sudah selesai di *train* akan disimpan disini untuk diambil nanti pada saat pencarian teks.

Untuk training metode *Skip-Gram*, kata target *training* akan dijalankan dari kata awal hingga akhir satu per satu dengan satu pasang tetangganya sebagai kata prediksi *output* (satu tetangga kanan dan satu tetangga kiri). Untuk kata pada awal dan akhir data yang hanya memiliki satu tetangga (hanya sebelah kanan dan hanya sebelah kiri), kata hanya akan dilatih dengan satu tetangga tersebut.

Untuk pelatihan metode CBOW, kata yang digunakan hanya merupakan kata pertama dan ketiga dari data sebagai kata *input* untuk metode CBOW dua kata, dan kata pertama, kedua, keempat, dan kelima untuk metode CBOW empat kata. Metode CBOW akan menggunakan kata tengah sebagai kata prediksi *output*. *Training* juga akan berjalan dari kata awal hingga akhir data, kata ketiga dihitung dari akhir data untuk metode dua kata dan kata kelima dihitung dari akhir data untuk metode empat kata.

Parameter yang digunakan untuk *training* data merupakan *learning rate* sebesar 0.05 dan 400 *hidden layer* (400 dimensi vektor). Dikarenakan lamanya waktu *training* data, penulis memilih untuk menghapus *stopwords* yang ada dalam teks. Daftar kata *stopwords* yang dihapus didapat dari “*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*”, Tala (2003). Kata yang hanya berisi angka juga dihapus karena dianggap tidak memiliki makna (contohnya kata ”10:43”).

Sistem terbagi menjadi tiga bagian : pembersihan data, *training* data, dan program pencarian. Pembersihan data terdiri dari satu kode skrip Python : *onehotencode.py*, *training* data terdiri dari tiga kode skrip : *training.py*, *trainingcbow.py*, *trainingcbow4kata.py* dan program pencarian terdiri dari tiga

kode skrip : *search.py*, *search2cbow.py* dan *search4cbow.py*.

B. Pembersihan Data

Pada pembersihan data, *onehotencode.py* berfungsi untuk memindahkan dan merapihkan data yang sudah di *crawl*. Berikut akan dijelaskan fungsi *onehotencode.py*. *onehotencode.py* akan mengambil sampel *random* data sebanyak 70 data artikel/dokumen dari data yang di *crawl*. Data ini per dokumennya berisi paragraf kata. Data yang diambil akan dihapus simbol-simbol spesialnya dari kata dan akan dihapus *stopwords*-nya dari paragraf. Data juga akan dirapihkan dengan menghapus *whitespace* yang berlebihan (*double whitespace* atau lebih) pada paragraf. Kata yang hanya terdiri dari angka juga akan dihapus. Paragraf yang sudah rapih akan digabungkan dengan paragraf-paragraf dari artikel lain lalu disimpan ke dalam tabel *big_data*. Paragraf besar ini akan digunakan untuk pelatihan sistem. Data yang sudah dirapihkan per-paragraf juga akan dimasukkan ke *database* tabel *text* dengan format sumber, isi. Tabel ini akan digunakan untuk pencarian dokumen dengan kata pada tahap akhir.

```

#ubah data character unicode
stringvalue = unidecode.unidecode(tupledatabase[x][1])
stringvalue = stringvalue.lower()
stringvalue = stringvalue.replace(","," ")
stringvalue = stringvalue.replace("."," ")
stringvalue = stringvalue.replace("\n"," ")
stringvalue = stringvalue.replace("#"," ")
stringvalue = stringvalue.replace("&"," ")
stringvalue = stringvalue.replace("''","")
stringvalue = stringvalue.replace("(,"," ")
stringvalue = stringvalue.replace(")"," ")
stringvalue = stringvalue.replace("@"," ")
stringvalue = stringvalue.replace("/"," ")
stringvalue = stringvalue.replace(":"," ")
stringvalue = stringvalue.replace("*"," ")
stringvalue = stringvalue.replace("[," ")
stringvalue = stringvalue.replace("]","," ")
stringvalue = stringvalue.replace("?"," ")
stringvalue = stringvalue.replace("-"," ")
stringvalue = stringvalue.replace("%"," ")
stringvalue = stringvalue.replace("$"," ")
stringvalue = stringvalue.replace("!"," ")
stringvalue = stringvalue.replace("<"," ")
stringvalue = stringvalue.replace(">"," ")
stringvalue = stringvalue.replace("="," ")
stringvalue = stringvalue.replace("|"," ")
stringvalue = stringvalue.replace("_"," ")
stringvalue = stringvalue.replace(";","," )

#hapus stopwords
for nomor in range(len(stopwords)) :
    stringvalue = re.sub('\\b'+stopwords[nomor]+'\b', '', stringvalue)

```

Gambar 4.2 : Pembersihan Data

Setelah data paragraf disimpan, *onehotencode.py* akan mengambil kata unik pada semua data. Semua kata unik akan dikumpulkan lalu dibuat *one-hot-encode*-nya. Total kata unik yang di dapatkan setelah data di bersihkan di dapat 4199 total kata. Kata unik dan *one-hot-encode* kata tersebut lalu akan disimpan di *database* pada tabel *dictionary*. Tabel ini akan digunakan sebagai tabel *lookup one-hot-encode* dari kata pada saat proses *training* dan *searching*.

Gambar 4.3 : Contoh *List Kata* dan *One-Hot-Encode*

C. *Training Data*

Training data terdiri dari tiga kode skrip, yaitu *training.py*, *trainingsbow.py*, dan *trainingsbow4kata.py*. Berikut akan dijelaskan fungsi *training.py*. *Training.py* berfungsi untuk melatih data dengan model *Continuous Skip-Gram*. Pada awalnya *training.py* akan menginisiasi *matrix weight input* dan *output* secara *random*, dengan ukuran *matrix weight input* 4199x400 (jumlah kata unik x *hidden layer*) dan ukuran *matrix weight output* 400x4199 (*hidden layer* x jumlah kata unik). Skrip ini juga akan mengambil data kata unik, *one-hot-encode* dan paragraf besar berisi semua

kata dari tabel *dictionary* dan *big_data* pada *database* secara berurutan.

```
D:\files\Desktop\SKRIPSI\github\skripsi>trainingcbow.py
Weight Input Matrix :
[[1. 1. 2. ... 3. 2. 3.]
 [1. 3. 1. ... 3. 2. 3.]
 [3. 1. 2. ... 3. 2. 3.]
 ...
 [3. 3. 1. ... 3. 2. 1.]
 [3. 3. 1. ... 2. 3. 1.]
 [2. 1. 2. ... 1. 2. 1.]]]

Weight Output Matrix :
[[3. 1. 2. ... 3. 1. 1.]
 [3. 2. 2. ... 2. 3. 3.]
 [3. 3. 1. ... 2. 1. 1.]
 ...
 [1. 3. 1. ... 1. 1. 1.]
 [3. 1. 3. ... 3. 2. 1.]
 [2. 2. 1. ... 3. 1. 1.]]
```

Gambar 4.4 : Inisiasi *Weight* Secara *Random*

Training akan berjalan pada *big_data* perkata, kata pertama akan dicari *one-hot-encode*-nya pada *dictionary* lalu *one-hot-encode* kata akan dikalikan dengan transpos *weight matrix input*. Hasilnya (*hidden layer*), akan dikalikan dengan transpos dari *weight output matrix*. Hasil perkalian ini akan di *softmax* untuk mendapatkan prediksi. Prediksi akan dikurangi dengan *one-hot-encode* kata sebenarnya (kata tetangga dari kata *input*). Hasil pengurangan prediksi dengan kata sebenarnya akan di total, lalu hasilnya akan digunakan untuk *backpropagation*.

```

#itung expuj
#yj = exp(uj)/sum(exp(uj')) untuk j' dari satu sampe V
#shift highest value ke 0 biar gak error (nggak mempengaruhi hasil)
expuj = np.exp(uj - np.max(uj))
ycj = expuj/np.sum(expuj)

ecjcplussatu = 0
#cek one hot encode kata setelahnya
#ecj = ycj+tcj
if x < len(bigdata)-1:
    onehotencodeplussatu = dictionarykata[bigdata[x+1]]
    ecjcplussatu = ycj - onehotencodeplussatu

ecjcminsatu = 0
#cek one hot encode kata sebelumnya
#ecj = ycj-tj
if x > 0 :
    onehotencodeminsatu = dictionarykata[bigdata[x-1]]
    ecjcminsatu = ycj - onehotencodeminsatu

#ej = sum(ecj)
ej = ecjcminsatu + ecjcplussatu

```

Gambar 4.5 : Mencari *Softmax* Lalu Mengurangi Hasilnya dengan Angka Sebenarnya

Pada *backpropagation*, *hidden layer* akan dikalikan dengan total *error* dan *learning rate*, lalu hasilnya akan digunakan untuk meng-update *matrix weight output*. Sedangkan untuk *matrix weight input*, *matrix weight output* sebelum di *update* akan dikalikan dengan total *error*, lalu di transpos lalu dikalikan dengan *learning rate*. Vektor hasil akan digunakan untuk meng-update vektor pada *matrix weight input* yang merepresentasikan kata yang di-input.

Setelah *training* selesai *matrix weight input* dan *matrix weight output* yang sudah di *update* akan disimpan ke dalam *database*, ke dalam tabel *weight*. Tabel ini akan digunakan untuk *searching* pada tahap akhir. Sebelum dimasukkan ke dalam *database*, *weight matrix input* dan *matrix weight output* akan di *flatten* agar *formatting* lebih gampang pada saat *search* nanti. Tabel *dictionary* juga akan di *update* pada bagian *vector_skip_gram* dengan isi vektor pada *matrix weight input* yang merepresentasikan kata.

```

weight matrix input yang akan di masukkan ke database :
[[0.66861141 1.81910278 2.31471048 ... 3.02903372 1.12650077 1.90940147]
[2.06087423 3.30954132 0.76524658 ... 1.17991629 3.14539755 2.82222224]
[1.88055638 2.78680176 1.04443916 ... 1.10992735 1.73624741 2.297798 ]
...
[2.68702576 2.50604477 1.90868067 ... 1.42041606 3.07826243 2.88398501]
[1.85035782 0.82163707 1.45589056 ... 2.17178086 2.29993926 0.97525612]
[1.48394487 1.27781298 2.22503808 ... 2.38440162 2.15438563 2.60129766]]

weight matrix output yang akan di masukkan ke database :
[[1.85726298 2.6464074 0.44717447 ... 1.75765527 2.78731377 2.20045236]
[1.34900578 3.24814743 3.31361929 ... 0.78198922 2.91570362 0.72021436]
[1.16428053 3.21170681 2.75204446 ... 0.88634806 1.80265504 2.90655846]
...
[3.11102419 1.96730425 1.3510706 ... 0.94858325 2.85020576 1.0836745 ]
[1.62194936 1.99361892 2.94012745 ... 2.64801167 0.97338604 3.09182987]
[0.92441178 2.37766103 2.49098356 ... 1.15621836 1.82597384 1.65059268]]

```

Gambar 4.6 : *Weight Matrix Setelah Training*

Untuk *trainingcbow.py* fungsinya juga sama, yaitu *training* kata. Bedanya ia dengan menggunakan model CBOW. Dua kata dicari *one-hot-encode*-nya lalu dirata-ratakan, lalu dikalikan dengan *matrix weight input*. Hasil *hidden layer* dikalikan dengan *matrix weight output* lalu hasilnya di *softmax* untuk mendapatkan prediksi. Prediksi akan dibandingkan dengan kata sebenarnya, yaitu kata tengah dari dua kata yang *di-input*. Hasil *error*-nya akan digunakan untuk *backpropagation*. Fungsi ini juga memiliki *backpropagation* yang mirip, kecuali pada bagian *update weight matrix input* dimana dia meng-*update* dua vektor, yaitu vektor pada *matrix weight input* yang merepresentasikan dua kata *input* yang kita masukkan. Untuk *trainingcbow4kata.py*, fungsi hampir semua sama dengan *trainingcbow.py*, bedanya hanya kata *input*-nya yang empat kata dan *update* pada empat vektor yang merepresentasikan empat kata yang dimasukkan. Setelah selesai *training*, *matrix* akan di *flatten* lalu dimasukkan ke dalam *database*.

Dipilih waktu tiga jam untuk melakukan *training* kedua sistem. Digunakan *Euclidean Distance* untuk menghitung jarak dari hasil akhir (hasil prediksi *softmax*) dan hasil sebenarnya (*one-hot-encode* kata output). Pada *training* sistem *Skip-Gram*, hasil berhenti pada jarak tetangga kiri 0.87 dan tetangga kanan 0.9 setelah tiga jam dengan 43 iterasi. Sedangkan CBOW menghasilkan 0.1 untuk metode dua kata

setelah tiga jam dilatih dengan 50 iterasi dan 0.067 untuk metode empat kata setelah tiga jam dan 29 iterasi. Hasil jarak pada *Skip-Gram* dianggap terlalu jauh dan tidak layak digunakan untuk sistem pencarian.

D. Pencarian Kata Pada Dokumen

Program pencarian terdiri dari tiga kode skrip : *search.py*, *search2cbow.py* dan *search4cbow.py*. *search.py* melakukan pencarian dengan metode *Skip-Gram*, *searchcbow.py* melakukan pencarian dengan metode CBOW dua kata, dan *search4cbow.py* melakukan pencarian dengan CBOW metode empat kata. Ketiga program ini berbentuk terminal.

```
D:\files\Desktop\SKRIPSI\github\skripsi>search4cbow.py
Masukkan kata
█
```

Gambar 4.7 : Program Pencarian Berbasis Terminal

Pada *search.py*, pertama program akan mengambil *matrix weight input* dan *matrix weight output* yang sudah di *update* dari *database*. *Weight* yang sudah di *flatten* dari *database* di *format* kembali menjadi *matrix* dengan parameter *kata_unik* dan *hidden_layer* dari tabel *weight* itu juga. Skrip ini juga akan mengambil *one-hot-encode* dan semua jenis kata unik dari tabel *dictionary* pada *database*.

Program akan menampilkan *prompt user* untuk meng-*input* satu kata. Setelah kata di-*input* skrip akan mencari *one-hot-encode* dari kata *input user*. Program akan dijalankan seperti pada saat *training Skip-Gram*. *One-hot-encode* akan dikalikan dengan *matrix weight input*, lalu hasilnya, *hidden layer* akan dikalikan dengan *matrix weight output*. Lalu hasil dari perkalian ini akan di *softmax*.

Di *softmax* kita akan mencari dua nilai yang paling mendekati satu, dimana indeks dari dua nilai tersebut merupakan kata yang seharusnya kata tetangga dari kata *input*. Kata ini akan digabungkan dengan kata *input* (kata *input* di tengah dengan dua kata temu di sekitar) lalu skrip akan mengambil data dari tabel *text* pada

database, dimana tabel berisi semua sumber dan isi artikel. Di akhir, skrip akan melakukan pencarian tiga kata pada isi artikel dimana artikel dengan kemunculan kata paling tinggi akan menjadi dokumen dengan urutan paling tinggi.

```

yj = expuj/np.sum(expuj)
listonehotencodekata = list(dictionaryonehotencodekata)
katatengah = listonehotencodekata[np.argmax(yj)]
listkatatengah = []
i=0
while i <= 5 :
    i+=1
    listkatatengah.append(listonehotencodekata[np.argmax(yj)])
    yj[np.argmax(yj)] = 0

print("kata tengah merupakan : ", katatengah)
print("5 kata tengah tertinggi merupakan : ", listkatatengah)
print("tekan enter")
input()

```

Gambar 4.8 : Pencarian Kata Tengah pada CBOW dengan Mencari Indeks Angka Paling Besar pada Hasil *Softmax*

```

#print(len(listtext))
for i in range(len(listtext)):
    #print("sumber ke : ", i + 1, listtext[i][0])
    #cari kata valid di kolom content dengan menggunakan regular expression
    #itung jumlah kemunculan
    satudua = katainput[0] + " " + katatengah + " " + katainput[1]
    duasatu = katainput[1] + " " + katatengah + " " + katainput[0]
    #print(satudua)
    #print(duasatu)
    jumlahkemunculansatudua = len(re.findall('\b' + satudua + '\b',listtext[i][1]))
    jumlahkemunculanduasatu = len(re.findall('\b' + duasatu + '\b',listtext[i][1]))
    #taro [katainputvalid : jumlah kemunculan] pada dictionarykatalvalidrelevan
    dictionarykatalvalidrelevan[satudua] = (jumlahkemunculansatudua)
    dictionarykatalvalidrelevan[duasatu] = (jumlahkemunculanduasatu)
    nilaidokumen = 1 * jumlahkemunculansatudua + 1 * jumlahkemunculanduasatu
    dictionarykatalvalidrelevan["nilai dokumen"] = nilaidokumen
    #print(nilaidokumen)

```

Gambar 4.9 : Pencarian Kata pada Semua Dokumen Menggunakan *Regular Ekspression*

```
D:\files\Desktop\skripsi\github>searchcbow.py
Masukkan kata
piala inggris

kata tengah merupakan : liga
5 kata tengah tertinggi merupakan : ['liga', 'klub', 'klasemen', 'musim', 'ing', 'alphonse']
tekan enter

('https://www.indosport.com/sepakbola/20221119/piala-dunia-2022-dibuang-luis-enrique-dari-skuat-timnas-spanyol-pemain-ini-malah-girang', {'piala liga inggris': 3, 'inggris liga piala': 0, 'nilai dokumen': 3})
('https://www.indosport.com/sepakbola/20221101/duel-antarlini-laga-liga-champions-inter-milan-vs-bayern-munchen-nerazzurri-cari-kelemahan-lawan', {'piala liga inggris': 3, 'inggris liga piala': 0, 'nilai dokumen': 3})
('https://www.indosport.com/sepakbola/20221117/5-bintang-liga-italia-yang-bisa-bawa-argentina-taklukkan-piala-dunia-2022', {'piala liga inggris': 3, 'inggris liga piala': 0, 'nilai dokumen': 3})
('https://www.indosport.com/sepakbola/20221117/termasuk-lionel-messi-dan-lisandro-martinez-ini-5-pemain-kunci-argentina-juara-piala-dunia-2022', {'piala liga inggris': 3, 'inggris liga piala': 0, 'nilai dokumen': 3})
('https://www.indosport.com/sepakbola/20221110/hasil-liga-italia-inter-milan-vs-bologna-sempat-tertinggal-nerazzurri-pesta-gol', {'piala liga inggris': 3, 'inggris liga piala': 0, 'nilai dokumen': 3})
Waktu total yang dibutuhkan : 1.80668044090271
```

Gambar 4.10 : Contoh Hasil Pencarian Dokumen

searchcbow.py mirip dengan *search.py* dimana bedanya ia perlu meng-*input* dua kata dengan *output* satu kata. Ketiga kata keluaran akan digabung (kata keluaran di tengah-tengah kata *input*) lalu akan dicari dokumen mana yang memiliki kemunculan paling tinggi. *search4cbow.py* hampir sama dengan beda kata *input* yang dibutuhkan merupakan empat.

E. Pengetesan *User*

Pengetesan akan dilakukan terhadap tiga *user* terhadap tiga eksperimen, metode *Skip-Gram*, metode CBOW, dan metode gabungan. *User* akan diminta meng-*input* kata yang sama ke dalam program lalu *user* akan diminta untuk memberikan *ranking* pada lima hasil dokumen yang keluar. Selanjutnya penulis akan menilai *ranking* pencarian dokumen yang dikeluarkan sistem terhadap *ranking user*. *User* akan diminta mengisi kuisioner tentang relevansi hasil dokumen pada sistem.

```
D:\files\Desktop\skripsi\github>searchcbow.py
Masukkan kata
piala inggris

Hasil dokumen : https://www.indosport.com/sepakbola/20221119/piala-dunia-2022-dibuang-luis-enrique-dari-skuat-timnas-spanyol-pemain-ini-malah-girang
Hasil dokumen : https://www.indosport.com/sepakbola/20221101/duel-antarlini-laga-liga-champions-inter-milan-vs-bayern-munchen-nerazzurri-cari-kelemahan-lawan
Hasil dokumen : https://www.indosport.com/sepakbola/20221117/5-bintang-liga-italia-yang-bisa-bawa-argentina-taklukkan-piala-dunia-2022
Hasil dokumen : https://www.indosport.com/sepakbola/20221117/termasuk-lionel-messi-dan-lisandro-martinez-ini-5-pemain-kunci-argentina-juara-piala-dunia-2022
Hasil dokumen : https://www.indosport.com/sepakbola/20221110/hasil-liga-italia-inter-milan-vs-bologna-sempat-tertinggal-nerazzurri-pesta-gol

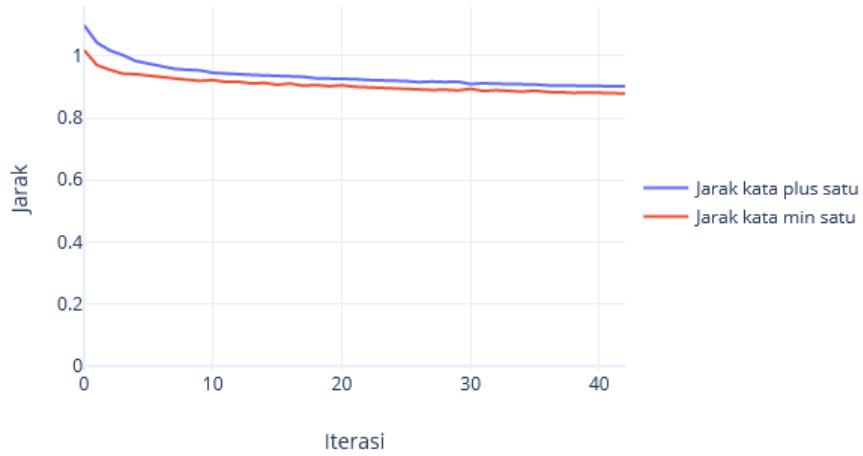
Waktu total yang dibutuhkan : 0.026414871215820312
```

Gambar 4.11 : Contoh Hasil Pencarian Dokumen oleh *User*

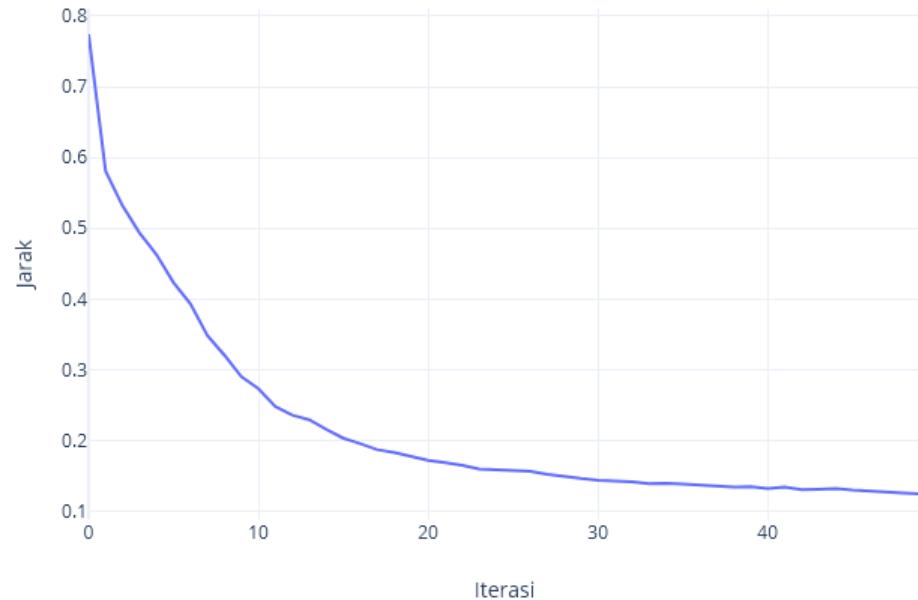
F. Analisis Hasil

Bagian ini akan membahas hasil dari tiga eksperimen yang telah dilakukan. Hasil skenario percobaan yang pertama, pada saat pelatihan proses *training Skip-Gram*, setelah tiga jam *training*, hasil yang di dapat terlalu jauh dengan hasil yang seharusnya. Hasil yang di dapat merupakan jarak antara hasil prediksi dan hasil yang sebenarnya yang di hitung dengan menggunakan *Euclidean Distance*. Jarak akhir yang di dapat merupakan 0.87 untuk tetangga kiri dan 0.9 untuk tetangga kanan setelah 43 iterasi selama tiga jam. Berdasarkan landainya grafik dan hasil jarak pada *Skip-Gram* yang dianggap terlalu jauh, maka sistem dianggap tidak cocok untuk skenario pengetesan.

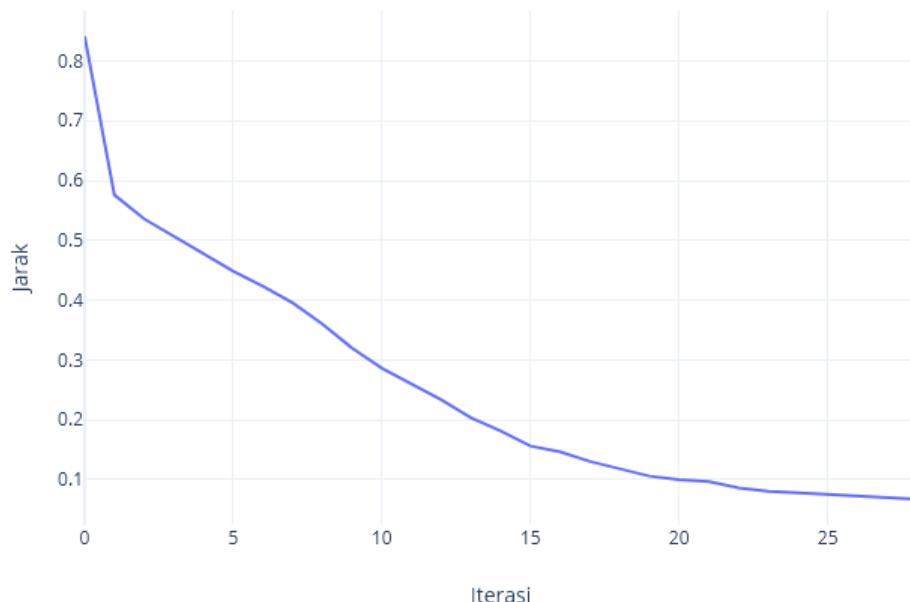
Untuk skenario CBOW, *training* sistem *input* dua kata dengan waktu tiga jam menghasilkan hasil jarak 0.1 pada iterasi 50, sedangkan *training* sistem *input* empat kata menghasilkan jarak akhir 0.067 dengan total 29 iterasi. Dengan hasil tersebut sistem dianggap layak untuk di tes ke *user*.



Gambar 4.12 : Jarak Antara Prediksi dengan Hasil Sebenarnya *Skip-Gram*



Gambar 4.13 : Jarak Antara Prediksi dengan Hasil Sebenarnya CBOW Dua Kata



Gambar 4.14 : Jarak Antara Prediksi dengan Hasil Sebenarnya CBOW Empat Kata

Untuk pengetesan relevansi pencarian terhadap *user* dengan menggunakan metode CBOW, *user* akan diberikan lima pasang kata untuk metode CBOW dua kata, dan lima dua pasang kata untuk metode CBOW empat kata. *User* akan diminta untuk memasukkan kata tersebut ke dalam mesin pencarian. Mesin pencarian akan mengeluarkan lima hasil dokumen lalu *user* diminta untuk menilai *ranking* dari dokumen keluaran sistem tersebut. Untuk hasil pencarian teks yang hanya menghasilkan jumlah dokumen yang kurang dari lima, sisa data akan diisi dengan dokumen random untuk dinilai oleh *user*, dan untuk dokumen yang memiliki jumlah hasil kemunculan kata yang sama, dokumen akan diberikan *ranking* yang sama.

Berikut akan dipaparkan hasil pencarian dengan menggunakan metode CBOW dua kata.

Tabel 4.1 : Pencarian Pertama CBOW dengan Menggunakan Kata “hasil” dan “italia”

Dokumen	<i>Ranking</i>				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/tag/13322/lautaro-martinez	1 / 2	1	4	2	66.67%
https://www.indosport.com/sepakbola/2022110/hasil-liga-italia-inter-milan-vs-bologna-sempat-tertinggal-nerazzurri-pesta-gol	1 / 2	2	1	1	100%
https://www.indosport.com/sportainment/20200324/persija-libur-ini-yang-dilakukan-marko-simic-dan-manohara	x	3	2	5	x
https://www.indosport.com/tag/5059/kenny-sansom	x	4	3	4	x

Dokumen	<i>Ranking</i>				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/multisport/20190930/naik-ring-november-2019-daud-yordan-jalani-latihan-khusus-di-bali	x	5	5	3	x

Pencarian pertama dengan kata hasil dan italia menghasilkan kata tengah liga, dan kata “hasil liga italia” ditemukan sekali pada dua *ranking* teratas dengan nilai yang sama, yaitu masing-masing dokumen ditemukan kata tersebut hanya sekali. Sedangkan tiga *ranking* dibawahnya merupakan data random karena tidak ditemukan kata tersebut pada dokumen lain. Karena hasil pertama dan hasil kedua ditemukan dengan nilai yang sama, maka hasil pertama atau kedua diberikan *ranking* 1 atau 2. Jika *user* menilai *ranking* satu atau dua pada dokumen tersebut maka hasil akan dianggap sesuai. Pada hasil kueri pertama, didapatkan dua *hit* dari total tiga user, maka didapatkan *hit rate* sebesar $\frac{2}{3}$, yang menghasilkan kesesuaian 66.67%. Hasil kesesuaian rata-rata yang di dapat dari semua kueri merupakan 83.34%

Tabel 4.2 : Pencarian Kedua CBOW dengan Menggunakan Kata “pemain” dan “milan”

Dokumen	<i>Ranking</i>				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/20221010/prediksi-liga-champions-ac-milan-vs-chelsea-menang-atau-digilas-sejarah	1 / 2 / 3	1	1	1	100%
https://www.indosport.com/sepakbola/20221012/tengah-main-bersama-chelsea-pierre-emerick-aubameyang-punya-karier-aneh-di-ac-milan	1 / 2 / 3	2	2	2	100%
https://www.indosport.com/liga-italia	1 / 2 / 3	3	3	3	100%

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sportainment/20221110/bertabur-bintang-indonesian-esports-awards-2022-bakal-lebih-semarak-dan-meriah	4 / 5	4	4	4	100%
https://www.indosport.com/multi-event/20191004/mencontek-gaya-hidup-sehat-ala-polisi	4 / 5	5	5	5	100%

Pencarian kedua dengan kata pemain dan milan menghasilkan kata tengah ac. Pencarian dengan kata “pemain ac milan” ditemukan dua kali pada tiga dokumen teratas, dan sekali pada dua dokumen terakhir. Dokumen hasil ke empat dan ke lima menghasilkan topik yang tidak sesuai dengan kata *input* tetapi memiliki hasil kemunculan satu pada tiap dokumen, hal ini disebabkan oleh sumber data yang kurang sempurna pembersihannya, dimana judul berita rekomendasi artikel lain masih berada pada isi dokumen pada saat pencarian di lakukan di sistem. Rata-rata hasil kesesuaian terakhir yang dapat merupakan 100%

Tabel 4.3 : Pencarian Ketiga CBOW dengan Menggunakan Kata “juara” dan “dunia”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/tag/13322/lautaro-martinez	1 / 2	1	3	5	33.33%
https://www.indosport.com/sepakbola/20221116/5-pemain-yang-bisa-bantu-lionel-messi-dan-argentina-juara-piala-dunia-2022	1 / 2	2	1	1	100%
https://www.indosport.com/sepakbola/20221117/termasuk-lionel-messi-dan-lisandro-martinez-ini-5-pemain-kunci-argentina-juara-piala-dunia-2022	3 / 4	3	2	2	33.33%
https://www.indosport.com/liga-italia	3 / 4	4	4	4	100%

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/multi-event/20221201/kilas-balik-setahun-sanksi-wada-gara-gara-2-atlet-pakai-doping-indonesia-kelimpungan	x	5	5	3	x

Pencarian ketiga dengan kata juara dan dunia menghasilkan kata piala. Pencarian dengan kata “juara piala dunia” menghasilkan dua dokumen teratas dengan jumlah temu dua kali, jumlah temu sekali pada dokumen nomor tiga dan empat, dan tidak ditemukan pada dokumen lima, sehingga data diisi *random*. Hasil kesesuaian rata-rata yang didapat merupakan 66.67%. Hasil pencarian pertama merupakan *landing page* untuk *tag* pemain lautaro martinez, dimana *webpage* berisi banyak teks “juara” dan “piala dunia”. *User* menganggap hasil dokumen pertama tidak mencerminkan kata “juara” dan “dunia” dibanding dokumen lain.

Tabel 4.4 : Pencarian Keempat CBOW dengan Menggunakan Kata “olahraga” dan “tinju”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/tinju/20221202/jadwal-tinju-dunia-2022-duel-tak-berimbang-tyson-fury-vs-derek-chisora	1	1	1	1	100%
https://www.indosport.com/sepakbola/20220728/prediksi-liga-1-rans-nusantara-fc-vs-pss-sleman-berburu-3-poin-perdana	x	2	3	5	x
https://www.indosport.com/sportainment/20220314/pakai-sport-bra-minimalis-saat-siram-tanaman-body-yeyen-lidya-makin-aduhai	x	3	2	4	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/ragam/20221029/top-5-news-aaron-chia-sooh-woi-yik-kandas-ada-kejutan-dari-pemain-indonesia-di-luar-negeri	x	4	4	2	x
https://www.indosport.com/sepakbola/20221123/sengaja-buang-cristiano-ronaldo-3-pemain-ini-diam-diam-sudah-dilirik-manchester-united	x	5	5	3	x

Pencarian kata keempat, yaitu olahraga dan tinju, menghasilkan kata tengah berita. Pencarian dengan menggunakan kata “olahraga berita tinju” hanya menghasilkan satu dokumen, yaitu dokumen teratas dengan jumlah temu dua kali, sisanya empat dokumen diisi dengan empat dokumen *random*. Empat berita *random* yang diisi tidak berhubungan dengan kata tinju yang menghasilkan rata-rata kesesuaian 100%,

Tabel 4.5 : Pencarian Kelima CBOW dengan Menggunakan Kata “striker” dan “italia”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/tag/13322/lautaro-martinez	1 / 2	4	2	2	66.67%
https://www.indosport.com/sepakbola/20221124/4-striker-liga-italia-yang-cocok-gantikan-cristiano-ronaldo-di-manchester-united	1 / 2	1	1	1	100%
https://www.indosport.com/sepakbola/20221106/harap-harap-cemas-reuni-dengan-arsenal-aubameyang-optimistis-bawa-chelsea-menang-di-kandang	x	2	3	5	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/multi-event/20221103/pelatihan-timnas-indonesia-u-17-bima-sakti-ajak-anak-indonesia-rutin-cuci-tangan-pakai-sabun	x	3	5	4	x
https://www.indosport.com/ragam/20221029/top-5-news-aaron-chia-sooh-woi-yik-kandas-ada-kejutan-dari-pemain-indonesia-di-luar-negeri	x	5	4	3	x

Pencarian kata kelima, dengan menggunakan kata striker dan italia, menghasilkan kata tengah liga. Pencarian dengan menggunakan kata “striker liga italia” menghasilkan dua dokumen teratas dengan jumlah kata temu masing-masing satu. Sisa tiga dokumen diisi dengan dokumen random. Walau nilai kemunculan sama, rata-rata *user* menilai hasil dokumen dua lebih tinggi dibanding hasil dokumen pertama. Hasil rata-rata kesesuaian merupakan 83.34%.

Dengan itu dari lima pertanyaan total yang diberikan, hasil kesesuaian pe-ranking-an sistem pencarian dengan menggunakan metode CBOW dua kata terhadap pe-ranking-an *user* merupakan 86.67%.

Berikut merupakan hasil pencarian dengan menggunakan metode CBOW empat kata.

Tabel 4.6 : Pencarian Pertama CBOW dengan Menggunakan Kata “emain”, “unci”, “juara”, dan “piala”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/tag/13322/lautaro-martinez	1 / 2	3	3	2	33.33%
https://www.indosport.com/sepakbola/20221117/termasuk-lionel-messi-dan-lisandro-martinez-ini-5-emain-kunci-argentina-juara-piala-dunia-2022	1 / 2	1	1	1	100%
https://www.indosport.com/raket/20220502/sederhana-namun-tetap-khidmat Begini-suasana-perayaan-idulfitri-tim-bulutangkis-indonesia-di-manila	x	2	2	3	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/otomotif/20181031/muncul-bunyi-berisik-ini-solusi-sempurna-untuk-cvt-motor-matik	x	5	5	4	x
https://www.indosport.com/multi-event/20220506/breaking-news-asian-games-hangzhou-2022-resmi-ditunda-gara-gara-covid-19	x	4	4	5	x

Pencarian kata pertama menggunakan kata pemain, kunci, juara dan piala, menghasilkan kata tengah argentina. Pencarian dengan kata “pemain kunci argentina juara piala” menghasilkan jumlah temu satu kali pada dokumen pertama dan kedua, dan tidak ditemukan pada dokumen lainnya. Hasil rata-rata kesesuaian merupakan 66.67%

Tabel 4.7 : Pencarian Kedua CBOW dengan Menggunakan Kata “chelsea”, “lepas”, “abraham”, dan “arsenal”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/20210726/chelsea-bakal-lepas-tammy-abraham-ke-arsenal-bila-hal-ini-terwujud	1	1	1	1	100%
https://www.indosport.com/sepakbola/20221117/5-bintang-liga-italia-yang-bisa-bawa-argentina-taklukkan-piala-dunia-2022	x	3	4	4	x
https://www.indosport.com/multi-event/20200317/10-jenis-olahraga-yang-dapat-dilakukan-di-rumah	x	5	5	5	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/20190221/ivan-rakitic-segera-bergabung-dengan-inter-milan	x	2	3	3	x
https://www.indosport.com/ragam/20221029/top-5-news-aaron-chia-sooh-woi-yik-kandas-ada-kejutan-dari-pemain-indonesia-di-luar-negeri	x	4	2	2	x

Pencarian kata kedua menggunakan kata chelsea, lepas, abraham, dan arsenal, menghasilkan kata tengah tammy. Pencarian kata “chelsea lepas tammy abraham arsenal” menghasilkan dua kemunculan pada dokumen pertama dan nol kemunculan pada dokumen lainnya. Hasil rata-rata kesesuaian merupakan 100%

Tabel 4.8 : Pencarian Ketiga CBOW dengan Menggunakan Kata “cristiano”, “ronaldo”, “manchester”, dan “united”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/20221123/sengaja-buang-cristiano-ronaldo-3-pemain-ini-diam-diam-sudah-dilirik-manchester-united	1	1	1	1	100%
https://www.indosport.com/tag/5059/kenny-sansom	x	3	3	5	x
https://www.indosport.com/ragam/20221125/skateboard-senayan-komunitas-skate-asal-ibu-kota-sejak-1994	x	4	4	4	x
https://www.indosport.com/esports/20200326/salut-rrq-kampanye-anti-netizen-toxic-di-mpl-indonesia-season-5	x	5	5	3	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sportainment/20200324/persija-libur-ini-yang-dilakukan-marko-simic-dan-manohara	x	2	2	2	x

Pencarian ketiga menggunakan kata cristiano, ronaldo, manchester, united, menghasilkan kata tengah cabut. Pencarian dengan menggunakan kata “cristiano ronaldo cabut manchester united” menghasilkan jumlah kemunculan satu kali pada dokumen pertama dan nol kemunculan pada dokumen sisa. Hasil rata-rata kesesuaian merupakan 100%

Tabel 4.9 : Pencarian Keempat CBOW dengan Menggunakan Kata “asam”, “lambung”, “olahraga”, dan “cocok”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/multi-event/20190105/5-cara-jitu-berolahraga-bagi-penderita-asam-lambung	1	1	1	1	100%

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sportainment/20190622/tips-menggunakan-senjata-m249-di-game-esports-pubg-mobile	x	4	5	5	x
https://www.indosport.com/ragam/20221104/top-5-news-prediksi-pemain-timnas-jerman-di-piala-dunia-2022-ac milan-kunci-tiket-16-besar	x	2	4	4	x
https://www.indosport.com/raket	x	3	3	2	x
https://www.indosport.com/sepakbola/20221117/termasuk-lionel-messi-dan-lisandro-martinez-ini-5-pemain-kunci-argentina-juara-piala-dunia-2022	x	5	2	3	x

Pencarian keempat menggunakan kata asam, lambung, olahraga, cocok, menghasilkan kata tengah baca. Pencarian dengan menggunakan kata “asam lambung berita olahraga cocok” menghasilkan jumlah kemunculan satu kali pada dokumen pertama dan nol pada dokumen lainnya. Hasil rata-rata kesesuaian merupakan 100%

Tabel 4.10 : Pencarian Kelima CBOW dengan Menggunakan Kata “inter”, “milan”, “posisi”, dan “klasemen”

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/2022110/hasil-liga-italia-inter-milan-vs-bologna-sempat-tertinggal-nerazzurri-pesta-gol	1	3	1	1	66.67%
https://www.indosport.com/otomotif/20181031/muncul-bunyi-berisik-ini-solusi-sempurna-untuk-cvt-motor-matik	x	5	5	5	x
https://www.indosport.com/sepakbola/20190219/profil-pieter-tanuri-bos-bali-united-calon-komisaris-pt-lib	x	4	4	3	x

Dokumen	Ranking				Kesesuaian
	Sistem	User 1	User 2	User 3	
https://www.indosport.com/sepakbola/20221101/duel-antarlini-laga-liga-champions-inter-milan-vs-bayern-munchen-nerazzurri-cari-kelemahan-lawan	x	1	2	2	x
https://www.indosport.com/sepakbola/20221117/5-bintang-liga-italia-yang-bisa-bawa-argentina-taklukkan-piala-dunia-2022	x	2	3	4	x

Pencarian kelima dengan menggunakan kata inter, milan, posisi, klasemen, menghasilkan kata tengah melecit. Pencarian menggunakan kata “inter milan melecit posisi klasemen” menghasilkan kemunculan sekali pada dokumen pertama. Hasil kesesuaian rata-rata merupakan 66.67%.

Dengan itu rata-rata kemunculan pada sistem CBOW empat kata merupakan 86.67%. Kebanyakan pencarian dari CBOW empat kata hanya menghasilkan satu dokumen karena kata yang dicari perlu banyak.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan tiga eksperimen yang telah dikerjakan, kesimpulan dari analisis hasil, implementasi, dan evaluasi program dipaparkan sebagai berikut.

1. Hasil eksperimen pertama dianggap gagal, dikarenakan hasil akhir jarak antara vektor prediksi dan vektor kata yang sebenarnya merupakan 0.87 untuk tetangga kiri dan 0.9 untuk tetangga kanan, yang dianggap tidak layak untuk dijadikan sistem pencarian.
2. Hasil eksperimen kedua menunjukkan bahwa *training* sistem berhasil, yang ditunjukkan oleh hasil jarak 0.1 pada CBOW dua kata dan 0.067 pada CBOW empat kata setelah tiga jam training. Lalu berdasarkan pencarian dengan penggabungan kata *input* dan kata keluaran sistem, sistem menghasilkan rata-rata kesesuaian 86.67% untuk kedua sistem, CBOW dua kata dan CBOW empat kata. Dengan ini dapat disimpulkan bahwa pencarian menggunakan metode Continuous-Bag-of-Words dapat menghasilkan pencarian yang relevan.
3. Hasil eksperimen ketiga dianggap gagal karena sistem *Skip-Gram* tidak dapat digunakan.
4. Dikarenakan *training* waktu yang lama (tiga jam untuk mencapai dekat dengan nol pada CBOW dengan total data 38925 kata, 70 artikel), maka *neural network* ini dapat dianggap tidak bagus untuk di implementasikan pada mesin pencarian, apalagi ditambah dengan banyaknya data dokumen yang harus di proses dalam kumpulan data yang selalu mengekspansi.

B. Saran

Berdasarkan kesimpulan di atas, saran yang dapat diberikan penulis untuk penelitian selanjutnya yaitu sebagai berikut.

1. Menggunakan metode lain untuk mesin pencarian.
2. Menambahkan cara untuk membedakan *ranking* dari hasil pencarian dengan jumlah kata temu yang sama.
3. Merapihkan data *crawl* lebih baik lagi, karena terlihat dari hasil pencarian, data yang di *crawl* masih ada kalimat dan kata yang di proses yang tidak merepresentasikan judul artikel (contohnya judul rekomendasi *link* ke artikel lain).
4. Menggunakan gabungan kata *input* dan kata mirip dengan banyak variasi untuk ekspansi percobaan pencarian.
5. Menggunakan banyak kategori tema untuk melihat efeknya pada hasil pencarian.

DAFTAR PUSTAKA

- Al-Saqqa, 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., Karafiat, M., Burget, L., Cernocky, 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., & Mathematiques, 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
- Tala, F. (2003, 12). A study of stemming effects on information retrieval in bahasa indonesia.
- 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

LAMPIRAN

Lampiran 1 Formulir Test *User*

Formulir Test *User*

A. Pengujian Jalannya Program Pencarian Teks CBOW

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1.	Setelah program dijalankan, program akan menunjukkan prompt untuk meng- <i>input</i> kata.			
2.	Meng- <i>input</i> dua kata pada skrip search2cbow.py akan menyebabkan program mengembalikan lima hasil dokumen.			
3.	Meng- <i>input</i> empat kata pada skrip search4cbow.py kata akan menyebabkan program mengembalikan lima hasil dokumen.			

B. Pengujian Skenario CBOW Dua Kata

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "hasil" dan "italia", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "emain" dan "milan", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "juara" dan "dunia", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "olahraga" dan "tinju", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "striker" dan "italia", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

C. Pengujian Skenario CBOW Empat Kata

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "emain", "kunci", "juara", dan "piala", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "chelsea", "lepas", "abraham", dan "arsenal", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "cristiano", "ronaldo", "manchester", dan "united", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	User memasukkan kata "asam", "lambung", "olahraga", dan "cocok", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut user.			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut user.			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut user.			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut user.			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut user.			

No.	Skenario Pengujian	Kesesuaian		Kesimpulan / Urutan Seharusnya
		Sesuai	Tidak Sesuai	
1.	<i>User</i> memasukkan kata "inter", "milan", "posisi", dan "klasemen", lalu program akan mengembalikan lima hasil dokumen.			
2.	Dokumen keluaran program urutan nomor satu merupakan hasil nomor satu menurut <i>user</i> .			
3.	Dokumen keluaran program urutan nomor dua merupakan hasil nomor dua menurut <i>user</i> .			
4.	Dokumen keluaran program urutan nomor tiga merupakan hasil nomor tiga menurut <i>user</i> .			
5.	Dokumen keluaran program urutan nomor empat merupakan hasil nomor empat menurut <i>user</i> .			
6.	Dokumen keluaran program urutan nomor lima merupakan hasil nomor lima menurut <i>user</i> .			

Saya yang bertanda tangan di bawah ini, menyatakan bahwa pada hari
tanggal telah melakukan pengujian program dengan
requirement pengujian sebagai berikut:

1. Jalannya aplikasi.
2. Pe-ranking-an dokumen yang dikeluarkan aplikasi.

Jakarta,.....

(.....)

Lampiran 2 Source Code onehotencode.py

```

import random
import mysql.connector
import unidecode
import time
import re

startAwalTime = time.time()
startTime = startAwalTime

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="dbcrawl"
)

sourcecursor = sourcedb.cursor()

#ambil data dari dbcrawl, hasil tools fathan
sourcecursor.execute(
    "SELECT base_url,content_text FROM page_information")

myresult = sourcecursor.fetchall()

tupledatabase = myresult

targetdb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

targetcursor = targetdb.cursor()

#inisiasi string kosong untuk campuran semua string nanti
bigdata = ""

#list index dokumen yang udah dipilih
sampledDokumen = []

```

```
#stopwords
with open('stopwords.txt', 'r') as file:
    stopwords = file.read().split()

print("cleaning double whitespace...")
#tuple database isinya [sumber artikel(link)][isi artikel]
for iterasi in range(70):

    x = random.randint(0, len(tupledatabase)-1)
    while x in sampledDokumen :
        x = random.randint(0, len(tupledatabase)-1)
    sampledDokumen.append(x)

#ubah data character unicode
stringvalue = unidecode.unidecode(tupledatabase[x][1])
stringvalue = stringvalue.lower()
stringvalue = stringvalue.replace(","," ")
stringvalue = stringvalue.replace("."," ")
stringvalue = stringvalue.replace("\\""," ")
stringvalue = stringvalue.replace("#"," ")
stringvalue = stringvalue.replace("&"," ")
stringvalue = stringvalue.replace("'", "")
stringvalue = stringvalue.replace("( "," ")
stringvalue = stringvalue.replace(")"," ")
stringvalue = stringvalue.replace("@"," ")
stringvalue = stringvalue.replace("/"," ")
stringvalue = stringvalue.replace(":"," ")
stringvalue = stringvalue.replace("*"," ")
stringvalue = stringvalue.replace("["," ")
stringvalue = stringvalue.replace("]"," ")
stringvalue = stringvalue.replace("?"," ")
stringvalue = stringvalue.replace("-"," ")
stringvalue = stringvalue.replace("%"," ")
stringvalue = stringvalue.replace("$"," ")
stringvalue = stringvalue.replace("!"," ")
stringvalue = stringvalue.replace("<"," ")
stringvalue = stringvalue.replace(">"," ")
stringvalue = stringvalue.replace("="," ")
stringvalue = stringvalue.replace("|"," ")
stringvalue = stringvalue.replace("_"," ")
stringvalue = stringvalue.replace(";""," ")
```

```

#hapus stopwords
for nomor in range(len(stopwords)) :
    stringvalue = re.sub('\\\b'+stopwords[nomor]+ '\\b', ''
, stringvalue)

stringvaluesplit = stringvalue.split()
realstringvalue = []
#hapus kata yang isinya number doang
for z in range(len(stringvaluesplit)) :
    if stringvaluesplit[z].isdigit() == False:
        realstringvalue.append(stringvaluesplit[z])

#hapus whitespace di paragraph isi artikel kalo dia double/lebih
stringvalue = ' '.join(realstringvalue)

bigdata += stringvalue

#pindahin data dari database fathan + string yang udah dibersihin
#double/lebih whitespacenya ke database baru (tempat processing da
ta kita)
sql = "INSERT INTO text (sumber_url, content) VALUES (%s, %s)"
val = (tupledatabase[x][0],stringvalue)
targetcursor.execute(sql, val)
targetdb.commit()
print('jumlah data : ' + str(len(tupledatabase)))
print('Waktu yang dibutuhkan : ' + str(time.time() - startTime))
print()
startTime = time.time()

print("compiling data...")
#pindahin kumpulan semua string ke table baru
sql = "INSERT INTO big_data (compiled_string) VALUES (%s)"
val = (bigdata,)
targetcursor.execute(sql, val)
targetdb.commit()
print('Waktu yang dibutuhkan : ' + str(time.time() - startTime))
print()
startTime = time.time()

```

```

#perlu bikin biar wordnya gak duplicate dulu baru dimasukkan ke dalam database kayaknya
print("creating unique word dictionary...")
#bikin database dictionary unique word
listKata = bigdata.split()
listKata.sort()
listKataUnik = list(dict.fromkeys(listKata))
print('Jumlah Kata Tak Unik : ' + str(len(listKata)))
print('Jumlah Kata Unik : ' + str(len(listKataUnik)))
print('Waktu yang dibutuhkan : ' + str(time.time() - startTime))
print()
startTime = time.time()

#tuple berisi (kataUnik, one_hot_encode)
kataUnikDanOneHotEncode = []
#kita gak perlu peduli ordered atau nggaknya, one hot encode + kata langsung di ram
print("one-hot-encoding words...")
onehotencode = []
i = 0
for x in range(len(listKataUnik)):
    while i < len(listKataUnik) :
        onehotencode.append(0)
        i+=1
    onehotencode[x] = 1
    kataUnikDanOneHotEncode.append((listKataUnik[x], str(onehotencode)))
    i=0
    onehotencode = []
print('Waktu yang dibutuhkan : ' + str(time.time() - startTime))
print()
startTime = time.time()

print("saving unique word to database...")
#masukkan kata unique ke dalam database
sql =
"INSERT INTO dictionary (kata , one_hot_encode) VALUES (%s, %s)"
val = kataUnikDanOneHotEncode
targetcursor.executemany(sql, val)
targetdb.commit()
print('Waktu yang dibutuhkan : ' + str(time.time() - startTime))
print()
startTime = time.time()

waktuJalan = (time.time() - startAwalTime)
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))

#MULAI TRAINING

```

Lampiran 3 *Source Code* training.py

```

import numpy as np
import mysql.connector
import time
import sys

jumlahhiddenlayer = 400

learningRate = 0.7

jumlahIterasi = 1000

startAwalTime = time.time()

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

sourcecursor = sourcedb.cursor()

#ambil data dari dbcrawl, hasil tools fathan
sourcecursor.execute("SELECT kata,one_hot_encode FROM dictionary")

myresult = sourcecursor.fetchall()
jumlahkatadictionary = sourcecursor.rowcount

#Inisiasi Weight Input (lower bound 1, higher bound 4)(aneh, lower
#bound di include tapi higher bound di exclude)
#W=VxN
weightinputmatrix = np.random.randint(1, 4
, size=(jumlahkatadictionary, jumlahhiddenlayer))
weightinputmatrix = weightinputmatrix.astype(np.float64)
print("Weight Input Matrix :\n", weightinputmatrix)

#Inisiasi Weight Output (lower bound 1, higher bound 4)
#W'=NxV
weightoutputmatrix = np.random.randint(1, 4
, size=(jumlahhiddenlayer, jumlahkatadictionary))
weightoutputmatrix = weightoutputmatrix.astype(np.float64)
print("Weight Output Matrix :\n", weightoutputmatrix)

```

```

#ambil bigdata id paling terakhir
sourcecursor.execute(
"SELECT compiled_string FROM big_data WHERE ID = (SELECT MAX(ID) F
ROM big_data)"
)

rawbigdata = sourcecursor.fetchone()
bigdata = rawbigdata[0]
bigdata = bigdata.split()

dictionarykata = {}
#bikin dictionary kata dan one hot encode
startTime = time.time()
for i in range(jumlahkataadictionary):
    #ubah semua one hot encode ke array
    onehotencodekata = str(myresult[i][1])

#format dari [1, 0, 0, 0, 0, 0] jadi 100000 biar gampang di listnya
a
    onehotencodekata = onehotencodekata.replace(" ", "")
    onehotencodekata = onehotencodekata.replace(",", "")
    onehotencodekata = onehotencodekata.replace("[", "")
    onehotencodekata = onehotencodekata.replace("]", "")
    #ubah ke list biar gampang mapping ke array
    dictionarykata[myresult[i][0
]] = np.asarray(list(onehotencodekata)).astype(np.float64)
waktuarrayonehotencode = time.time() - startTime

totalwaktuonehotencode = 0
totalwaktuhiddenlayer = 0
totalwaktuuj = 0
totalwaktuyj = 0
totalwaktuecjplussatu = 0
totalwaktuecjminsatu = 0
totalwaktuej = 0
totalwaktuperkalianweightoutput = 0
totalwaktupenguranganweightoutput = 0
totalwaktuupdateweightinput = 0
listdictionarykata = list(dictionarykata)

bufferweightoutput = np.empty((jumlahhiddenlayer, jumlahkataadictio
nary)).astype(np.float64)

```

```

#Skip-Gram
#looping di bigdata
loop = True
iterasi = 0
while loop == True:
    iterasi = iterasi + 1
    listerror= np.empty((0,1),np.float64)
    listerrorminsatu= np.empty((0,1),np.float64)
    listerrorplussatu= np.empty((0,1),np.float64)
    print("iterasi ke : ",iterasi)
    for x in range(len(bigdata)):
        startTime = time.time()
        if x== 0 or x == len(bigdata)-1 :
            print("kata ke", x+1 , " dari ", len(bigdata))
        onehotencodekatatarget = dictionarykata[bigdata[x]]
        waktuonehotencode = time.time() - startTime
        totalwaktuonehotencode += waktuonehotencode

        #hidden layer
        #h = (W)T*x
        startTime = time.time()
        hiddenLayer = np.dot(np.transpose(weightinputmatrix),oneho
        tencodekatatarget)
        waktuhiddenlayer = time.time() - startTime
        totalwaktuhiddenlayer += waktuhiddenlayer

#itung ucj, hasil perkalian antara weight output matrix dengan hid
den layer
    #uj = (vwj)T * h
    startTime = time.time()
    uj = np.dot(np.transpose(weightoutputmatrix),hiddenLayer)
    waktuuj = time.time() - startTime
    totalwaktuuj += waktuuj

    #itung expuj
    #yj = exp(uj)/sum(exp(uj')) untuk j' dari satu sampe V
    startTime = time.time()

#shift highest value ke 0 biar gak error (nggak mempengaruhi hasi
1)
    expuj = np.exp(uj - np.max(uj))
    ycj = expuj/np.sum(expuj)
    waktuyj = time.time() - startTime
    totalwaktuyj += waktuyj

```

```

ecjcplussatu = 0
errorplussatu = 0
#cek one hot encode kata setelahnya
#ecj = ycj-tcj
if x < len(bigdata)-1:
    startTime = time.time()
    onehotencodeplussatu = dictionarykata[bigdata[x+1]]
    ecjcplussatu = ycj - onehotencodeplussatu
    waktuecjplussatu = time.time() - startTime
    totalwaktuecjplussatu += waktuecjplussatu

ecjcminsatu = 0
errorminsatu = 0
#cek one hot encode kata sebelumnya
#ecj = ycj-tj
if x > 0 :
    startTime = time.time()
    onehotencodeminsatu = dictionarykata[bigdata[x-1]]
    ecjcminsatu = ycj - onehotencodeminsatu
    waktuecjminsatu = time.time() - startTime
    totalwaktuecjminsatu += waktuecjminsatu

#ej = sum(ecj)
startTime = time.time()
ej = ecjcminsatu + ecjcplussatu
waktuej = time.time() - startTime
if x > 0 and x < len(bigdata)-1:
    errorplus = np.linalg.norm(ycj-onehotencodeplussatu)
    errormin = np.linalg.norm(ycj-onehotencodeminsatu)
    totalerror = errorplus + errormin
    listerrorminsatu = np.append(listerrorminsatu,np.array([[error
min]]),axis=
0)
    listerrorplussatu = np.append(listerrorplussatu,np.array([[err
orplus]]),axis=
0)
    c = 2
    error = -(c * uj) + c * np.log(np.sum(expuj))
elif x == len(bigdata)-1 :
    totalerror = np.linalg.norm(ycj-onehotencodeminsatu)
    errormin = np.linalg.norm(ycj-onehotencodeminsatu)
    listerrorminsatu = np.append(listerrorminsatu,np.array([[error
min]]),axis=
0)
    c = 1
    error = -(c * uj) + c * np.log(np.sum(expuj))

```

```

    elif x == 0 :
        totalerror = np.linalg.norm(ycj-onehotencodeplussatu)
        errorplus = np.linalg.norm(ycj-onehotencodeplussatu)
        listerrorplussatu = np.append(listerrorplussatu,np.array([[err
orplus]]),axis=
0)
        c = 1
        error = -(c * uj) + (c * np.log(np.sum(expuj)))
        listerror = np.append(listerror,np.array([[totalerror]]),axis=0)

        totalwaktuej += waktuej

        #update weight output matrix
        #W'ij(new) = W'ij(old) - n * ej * h
        #ej = sum(ecj)
        startTime = time.time()
        #pakai buffer buat multiply dan outer, biar cepet
        np.multiply(learningRate, np.outer(hiddenLayer , ej, bufferweighto
utput), bufferweightoutput)
        waktuperkalianweightoutput = time.time() - startTime
        totalwaktuperkalianweightoutput += waktuperkalianweightoutput

        #update weight input matrix
        #Vwi(new) = Vwi(old) - n * (EH)T
        #EHi = sum (EIj * W'ij)
        #EIj = sum(ecj)
        #EH berarti perkalian matrix weight output dengan sum error yang m
enghasilkan N dimension vector
        #N dimension berarti dia sejumlah hidden layer
        startTime = time.time()
        EH = np.dot(weightoutputmatrix,ej)
        indexkatatarget = listdictionarykata.index(bigdata[x])
        weightinputmatrix[indexkatatarget] = weightinputmatrix[indexkatata
rget] - learningRate * np.transpose(EH)
        waktuupdateweightinput = time.time() - startTime
        totalwaktuupdateweightinput += waktuupdateweightinput

        #update weight output lanjutan, ini bagian kurangin doang
        startTime = time.time()
        weightoutputmatrix -= bufferweightoutput
        waktupenguranganweightoutput = time.time() - startTime
        totalwaktupenguranganweightoutput += waktupenguranganweightoutput

```

```

ratarataerrorminsatu = np.mean(listerrorminsatu)
ratarataerrorplussatu = np.mean(listerrorplussatu)
ratarataerror = np.mean(listerror)
print("rata-rata error : ", ratarataerror)
print("error : ", error)
print("rata-rata error min satu : ", ratarataerrorminsatu)
print("rata-rata error plus satu : ", ratarataerrorplussatu)
if time.time() - startAwalTime >= 10800:
    loop = False

sourcedb.ping(reconnect=True)

#input weight matrix input dan output ke dalem database
startTime = time.time()
sql =
"INSERT INTO weight (metode, matrix_weight_input, matrix_weight_ou
tput) VALUES (%s, %s, %s)"
#set threshold sebelum di simpen ke database biar gak di truncate
#supress biar dia gak di singkat jadi e-01
np.set_printoptions(threshold=sys.maxsize, suppress=True)
val = ("skip-gram",
       np.array2string(weightinputmatrix), np.array2string(weightoutput
matrix))
sourcecursor.execute(sql, val)
sourcedb.commit()
print(
'Waktu yang dibutuhkan untuk memasukkan weight ke database : '
+ str(time.time() - startTime))
print()

#input vektor kata ke dalem database setelah semua data selesai di
train
#tuple berisi (vektor kata, kata)
startTime = time.time()
vektordankata = []
for x in range(jumlahkataadictionary):
    vektordankata.append((str(weightinputmatrix[x]), listdictionary
kata[x]))
sql =
"UPDATE dictionary SET vector_skip_gram = %s WHERE kata = %s"
val = vektordankata
sourcecursor.executemany(sql, val)
sourcedb.commit()
print(
'Waktu yang dibutuhkan untuk memasukkan vektor per kata ke databas
e: '
+ str(time.time() - startTime))
print()
print()
waktuJalan = (time.time() - startAwalTime)

```

```
print('Total waktu ubah one hot encode ke array : '
    + str(waktuarrayonehotencode))
print()
print('Total waktu one hot encode : '
    + str(totalwaktuonehotencode))
print()
print('Total waktu hidden layer : ' + str(totalwaktuhiddenlayer))
print()
print('Total waktu uj : ' + str(totalwaktuuj))
print()
print('Total waktu yj : ' + str(totalwaktuyj))
print()
print('Total waktu ejplussatu : ' + str(totalwaktuecjplussatu))
print()
print('Total waktu ejminsatu : ' + str(totalwaktuecjminsatu))
print()
print('Total waktu perkalian weight output : '
    + str(totalwaktuperkalianweightoutput))
print()
print('Total waktu pengurangan weight output : '
    + str(totalwaktupenguranganweightoutput))
print()
print('Total waktu update weight input : '
    + str(totalwaktuupdateweightinput))
print()
print('Jumlah Iterasi : ' + str(iterasi))
print()
print()
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))
```

Lampiran 4 Source Code trainingbow.py

```

import numpy as np
import mysql.connector
import time
import sys

jumlahhiddenlayer = 400

learningRate = 0.05

startAwalTime = time.time()

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

sourcecursor = sourcedb.cursor()

#ambil data dari dbcrawl, hasil tools fathan
sourcecursor.execute("SELECT kata,one_hot_encode FROM dictionary")

myresult = sourcecursor.fetchall()
jumlahkatadictionary = sourcecursor.rowcount

#Inisiasi Weight Input (lower bound 1, higher bound 4)(aneh, lower
#bound di include tapi higher bound di exclude)
#W=VxN
weightinputmatrix = np.random.randint(1, 4
, size=(jumlahkatadictionary, jumlahhiddenlayer))
weightinputmatrix = weightinputmatrix.astype(np.float64)

#Inisiasi Weight Output (lower bound 1, higher bound 4)
#W'=NxV
weightoutputmatrix = np.random.randint(1, 4
, size=(jumlahhiddenlayer, jumlahkatadictionary))
weightoutputmatrix = weightoutputmatrix.astype(np.float64)

#ambil bigdata id paling terakhir
sourcecursor.execute(
"SELECT compiled_string FROM big_data WHERE ID = (SELECT MAX(ID) F
ROM big_data)"
)

```

```

rawbigdata = sourcecursor.fetchone()
bigdata = rawbigdata[0]
bigdata = bigdata.split()

dictionarykata = {}
#bikin dictionary kata dan one hot encode
startTime = time.time()
for i in range(jumlahkatadictionary):
    #ubah semua one hot encode ke array
    onehotencodekata = str(myresult[i][1])

#format dari [1, 0, 0, 0, 0, 0] jadi 100000 biar gampang di listnya
a
    onehotencodekata = onehotencodekata.replace(" ", "")
    onehotencodekata = onehotencodekata.replace(",", "")
    onehotencodekata = onehotencodekata.replace("[", "")
    onehotencodekata = onehotencodekata.replace("]", "")
    #ubah ke list biar gampang mapping ke array
    dictionarykata[myresult[i][0]
]] = np.asarray(list(onehotencodekata)).astype(np.float64)
waktuarrayonehotencode = time.time() - startTime

listdictionarykata = list(dictionarykata)

bufferweightoutput = np.empty((jumlahhiddenlayer, jumlahkatadictio
nary)).astype(np.float64)

#CBOW
#looping di bigdata
loop = True
iterasi = 0
while loop == True :
    listerror= np.empty((0,1),np.float64)
    print("iterasi ke : ",iterasi + 1)
    iterasi += 1
    for x in range(len(bigdata)-2):
        startTime = time.time()
        if x== 0 or x == len(bigdata)-1 :
            print("kata konteks pertama :", bigdata[x])
            print("kata ke", x+1 , " dari ", len(bigdata))
            onehotencodekatakontekssatu = dictionarykata[bigdata[x]]

```

```

startTime = time.time()
if x== 0 or x == len(bigdata)-1 :
    print("kata konteks kedua :", bigdata[x+2])
    print("kata ke", x+3 , " dari ", len(bigdata))
onehotencodekatakonteksdua = dictionarykata[bigdata[x+2]]

#h = (1 / jumlah kata konteks) * (W)T * sum(onehotencodeinput)
startTime = time.time()
hiddenlayer = (1/2
) * np.dot(np.transpose(weightinputmatrix),(onehotencodekatakontek
ssatu + onehotencodekatakonteksdua)

#uj = (vwj')T * h
startTime = time.time()
uj = np.dot(np.transpose(weightoutputmatrix),hiddenlayer)

startTime = time.time()
#yj = exp(uj)/sum(exp(uj')) untuk j' dari satu sampe V
expuj = np.exp(uj - np.max(uj))
yj = expuj/np.sum(expuj)

#ej = yj-tj
startTime = time.time()
onehotencodekatatarget = dictionarykata[bigdata[x+1]]
ej = yj - onehotencodekatatarget
error = np.linalg.norm(yj-onehotencodekatatarget)
listerror = np.append(listerror,np.array([[error]]),axis=0)

#update weight matrix output
#w'ij(new) = w'ij(old) - n.ej.hi
startTime = time.time()
np.multiply(learningRate, np.outer(hiddenlayer , ej, bufferweighto
utput), bufferweightoutput)

#update weight matrix input
#vwi,c (new) = vwi,c(old) - 1/C * n * (EH)T untuk c dari 1 sampe C
startTime = time.time()
EH = np.dot(weightoutputmatrix,ej)
indexkontekskatasatu = listdictionarykata.index(bigdata[x])
indexkontekskatadua = listdictionarykata.index(bigdata[x+2])
weightinputmatrix[indexkontekskatasatu] = weightinputmatrix[indexk
ontekskatasatu] - (
1/2) * learningRate * np.transpose(EH)
weightinputmatrix[indexkontekskatadua] = weightinputmatrix[indexko
ntekskatadua] - (
1/2) * learningRate * np.transpose(EH)

```

```

        #update weight output lanjutan, ini bagian kurangin doang
        startTime = time.time()
        weightoutputmatrix -= bufferweightoutput

        ratarataerror = np.mean(listerror)
        print("rata-rata error : ", ratarataerror)
        if ratarataerror <= 0.001 or time.time() - startAwalTime >=
10800:
            loop = False

sourcedb.ping(reconnect=True)

#input weight matrix input dan output ke dalem database
startTime = time.time()
print("weight matrix input yang akan di masukkan ke database :\n"
, weightinputmatrix)
print()
print("weight matrix output yang akan di masukkan ke database :\n"
, weightoutputmatrix)
print()
sql =
"INSERT INTO weight (metode, matrix_weight_input, matrix_weight_ou
tput, kata_unik, hidden_layer) VALUES (%s,%s,%s,%s,%s)"
#set threshold sebelum di simpen ke database biar gak di truncate
#supress biar dia gak di singkat jadi e-01
np.set_printoptions(threshold=sys.maxsize, suppress=True)
val = ("cbow"
, np.array2string(weightinputmatrix.flatten()), np.array2string(we
ightoutputmatrix.flatten()), jumlahkatadictionary, jumlahhiddenlaye
r)
sourcecursor.execute(sql, val)
sourcedb.commit()
print(
'Waktu yang dibutuhkan untuk memasukkan weight ke database : '
+ str(time.time() - startTime))
print()

```

```
#input vektor kata ke dalam database setelah semua data selesai di
train
#tuple berisi (vektor kata, kata)
startTime = time.time()
vektordankata = []
for x in range(jumlahkatadictionary):
    vektordankata.append((str(weightinputmatrix[x]),myresult[x][0]
)))

sql = "UPDATE dictionary SET vector_cbow = %s WHERE kata = %s"
val = vektordankata
sourcecursor.executemany(sql, val)
sourcedb.commit()
print(
'Waktu yang dibutuhkan untuk memasukkan vektor per kata ke database
:
+
+ str(time.time() - startTime))
print()

print('Total iterasi : ' + str(iterasi))
print()

waktuJalan = (time.time() - startAwalTime)
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))
```

Lampiran 5 Source Code trainingbow4kata.py

```

import numpy as np
import mysql.connector
import time
import sys

jumlahhiddenlayer = 400

learningRate = 0.05

startAwalTime = time.time()

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

sourcecursor = sourcedb.cursor()

#ambil data dari dbcrawl, hasil tools fathan
sourcecursor.execute("SELECT kata,one_hot_encode FROM dictionary")

myresult = sourcecursor.fetchall()
jumlahkatadictionary = sourcecursor.rowcount

#Inisiasi Weight Input (lower bound 1, higher bound 4)(aneh, lower
#bound di include tapi higher bound di exclude)
#W=VxN
weightinputmatrix = np.random.randint(1, 4
, size=(jumlahkatadictionary, jumlahhiddenlayer))
weightinputmatrix = weightinputmatrix.astype(np.float64)

#Inisiasi Weight Output (lower bound 1, higher bound 4)
#W'=NxV
weightoutputmatrix = np.random.randint(1, 4
, size=(jumlahhiddenlayer, jumlahkatadictionary))
weightoutputmatrix = weightoutputmatrix.astype(np.float64)

#ambil bigdata id paling terakhir
sourcecursor.execute(
"SELECT compiled_string FROM big_data WHERE ID = (SELECT MAX(ID) F
ROM big_data)"
)

```

```

rawbigdata = sourcecursor.fetchone()
bigdata = rawbigdata[0]
bigdata = bigdata.split()

dictionarykata = {}
#bikin dictionary kata dan one hot encode
startTime = time.time()
for i in range(jumlahkatadictionary):
    #ubah semua one hot encode ke array
    onehotencodekata = str(myresult[i][1])

#format dari [1, 0, 0, 0, 0, 0] jadi 100000 biar gampang di listnya
a
    onehotencodekata = onehotencodekata.replace(" ", "")
    onehotencodekata = onehotencodekata.replace(",", "")
    onehotencodekata = onehotencodekata.replace("[", "")
    onehotencodekata = onehotencodekata.replace("]", "")
    #ubah ke list biar gampang mapping ke array
    dictionarykata[myresult[i][0]
]] = np.asarray(list(onehotencodekata)).astype(np.float64)
waktuarrayonehotencode = time.time() - startTime

listdictionarykata = list(dictionarykata)

bufferweightoutput = np.empty((jumlahhiddenlayer, jumlahkatadictio
nary)).astype(np.float64)

#CBOW
#looping di bigdata
loop = True
iterasi = 0
while loop == True :
    listerror= np.empty((0,1),np.float64)
    print("iterasi ke : ",iterasi + 1)
    iterasi += 1
    for x in range(len(bigdata)-5):
        startTime = time.time()
        if x== 0 or x == len(bigdata)-1 :
            print("kata konteks pertama :", bigdata[x])
            print("kata ke", x+1 , " dari ", len(bigdata))
        onehotencodekatakontekssatu = dictionarykata[bigdata[x]]

```

```

startTime = time.time()
if x== 0 or x == len(bigdata)-1 :
    print("kata konteks kedua :", bigdata[x+1])
    print("kata ke", x+2 , " dari ", len(bigdata))
onehotencodekatakonteksdua = dictionarykata[bigdata[x+1]]


startTime = time.time()
if x== 0 or x == len(bigdata)-1 :
    print("kata konteks ketiga :", bigdata[x+3])
    print("kata ke", x+4 , " dari ", len(bigdata))
onehotencodekatakontekstiga = dictionarykata[bigdata[x+3]]


startTime = time.time()
if x== 0 or x == len(bigdata)-1 :
    print("kata konteks keempat :", bigdata[x+4])
    print("kata ke", x+5 , " dari ", len(bigdata))
onehotencodekatakonteksempat = dictionarykata[bigdata[x+4]]


#h = (1 / jumlah kata konteks) * (W)T * sum(onehotencodeinput)
startTime = time.time()
hiddenlayer = (1/4
) * np.dot(np.transpose(weightinputmatrix),(onehotencodekatakontek
ssatu + onehotencodekatakonteksdua+onehotencodekatakontekstiga+one
hotencodekatakonteksempat))

#uj = (vwj')T * h
startTime = time.time()
uj = np.dot(np.transpose(weightoutputmatrix),hiddenlayer)

startTime = time.time()
#yj = exp(uj)/sum(exp(uj')) untuk j' dari satu sampe V
expuj = np.exp(uj - np.max(uj))
yj = expuj/np.sum(expuj)


#ej = yj-tj
startTime = time.time()
onehotencodekatatarget = dictionarykata[bigdata[x+2]]
ej = yj - onehotencodekatatarget
error = np.linalg.norm(yj-onehotencodekatatarget)
listerror = np.append(listerror,np.array([[error]]),axis=0)

```

```

        #update weight matrix output
        #w'ij(new) = w'ij(old) - n.ej.hi
        startTime = time.time()
        np.multiply(learningRate, np.outer(hiddenlayer , ej, bufferweightoutput), bufferweightoutput)

        #update weight matrix input

        #vwi,c (new) = vwi,c(old) - 1/C * n * (EH)T untuk c dari 1 sampe C
        startTime = time.time()
        EH = np.dot(weightoutputmatrix,ej)
        indexkontekskatasatu = listdictionarykata.index(bigdata[x])
        indexkontekskatadua = listdictionarykata.index(bigdata[x+1])
        indexkontekskatatiga = listdictionarykata.index(bigdata[x+3])
        indexkontekskataempat = listdictionarykata.index(bigdata[x+4])
        weightinputmatrix[indexkontekskatasatu] = weightinputmatrix[x[indexkontekskatasatu]] - (
        1/4) * learningRate * np.transpose(EH)
        weightinputmatrix[indexkontekskatadua] = weightinputmatrix[x[indexkontekskatadua]] - (
        1/4) * learningRate * np.transpose(EH)
        weightinputmatrix[indexkontekskatatiga] = weightinputmatrix[x[indexkontekskatatiga]] - (
        1/4) * learningRate * np.transpose(EH)
        weightinputmatrix[indexkontekskataempat] = weightinputmatrix[x[indexkontekskataempat]] - (
        1/4) * learningRate * np.transpose(EH)

        #update weight output lanjutan, ini bagian kurangin doang
        startTime = time.time()
        weightoutputmatrix -= bufferweightoutput

        ratarataerror = np.mean(listerror)
        print("rata-rata error : ", ratarataerror)
        if ratarataerror <= 0.001 or time.time() - startAwalTime >=
10800:
            loop = False

sourcedb.ping(reconnect=True)

```

```

#input weight matrix input dan output ke dalam database
startTime = time.time()
print("weight matrix input yang akan di masukkan ke database :\n"
, weightinputmatrix)
print()
print("weight matrix output yang akan di masukkan ke database :\n"
, weightoutputmatrix)
print()
sql =
"INSERT INTO weight (metode, matrix_weight_input, matrix_weight_ou
tput, kata_unik, hidden_layer) VALUES (%s,%s,%s,%s,%s)"
#set threshold sebelum di simpen ke database biar gak di truncate
#suppress biar dia gak di singkat jadi e-01
np.set_printoptions(threshold=sys.maxsize, suppress=True)
val = ("cbow4kata"
, np.array2string(weightinputmatrix.flatten()), np.array2string(we
ightoutputmatrix.flatten()),jumlahkataadictionary,jumlahhiddenlaye
r)
sourcecursor.execute(sql, val)
sourcedb.commit()
print(
'Waktu yang dibutuhkan untuk memasukkan weight ke database : '
+ str(time.time() - startTime))
print()

#input vektor kata ke dalam database setelah semua data selesai di
train
#tuple berisi (vektor kata, kata)
startTime = time.time()
vektordankata = []
for x in range(jumlahkataadictionary):
    vektordankata.append((str(weightinputmatrix[x]),myresult[x][0
])))
    print('Total iterasi : ' + str(iterasi))
    print()

waktuJalan = (time.time() - startAwalTime)
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))

```

Lampiran 6 Source Code search2cbow.py

```

import mysql.connector
import numpy as np
import time
import re
import random

katainput = input('Masukkan kata \n')
startAwalTime = time.time()
print()

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

sourcecursor = sourcedb.cursor()

sourcecursor.execute(
"SELECT kata, one_hot_encode, vector_cbow FROM dictionary")

myresult = sourcecursor.fetchall()

dictionaryonehotencodekata = {}
#bikin dictionary kata dan vektor skip gram

sourcecursor.execute(
"SELECT matrix_weight_input,matrix_weight_output,kata_unik,hidden_
layer FROM weight WHERE metode='cbow' GROUP BY ID = (SELECT MAX(I
D) FROM weight)"
)
weight = sourcecursor.fetchone()
matrixweightinput = weight[0]
matrixweightinput = matrixweightinput.replace("[", "")
matrixweightinput = matrixweightinput.replace("]", "")
#weight = kataunik x hiddenlayer
matrixweightinput = np.fromstring(matrixweightinput.strip('\n'
),sep=' ').reshape((weight[2], weight[3]))

matrixweightoutput = weight[1]
matrixweightoutput = matrixweightoutput.replace("[", "")
matrixweightoutput = matrixweightoutput.replace("]", "")
#weight = hiddenlayer x kataunik
matrixweightoutput = np.fromstring(matrixweightoutput.strip('\n'
),sep=' ').reshape((weight[3], weight[2]))

```

```

for i in range(len(myresult)):
    onehotencode = myresult[i][1]
    onehotencode = onehotencode.replace(" ", "")
    onehotencode = onehotencode.replace(",", "")
    onehotencode = onehotencode.replace("[", "")
    onehotencode = onehotencode.replace("]", "")
    onehotencode = np.asarray(list(onehotencode)).astype(np.float64)
    dictionaryonehotencodemata[myresult[i][0]] = onehotencode

katainput = katainput.split()
if len(katainput) != 2 :
    print(
"jumlah kata yang dimasukkan tidak sesuai, silahkan jalankan ulang
program"
)
    print()
    quit()

onehotencodemata = dictionaryonehotencodemata.get(katainput[0])
onehotencodemata2 = dictionaryonehotencodemata.get(katainput[1])
if onehotencodemata is not None and onehotencodemata2 is not None:
    katainput[0]
    katainput[1]
    hiddenlayer = (1/2
) * np.dot(np.transpose(matrixweightinput),(onehotencodemata + one
hotencodemata2))
    uj = np.dot(np.transpose(matrixweightoutput),hiddenlayer)
    expuj = np.exp(uj - np.max(uj))
    yj = expuj/np.sum(expuj)
    listonehotencodemata = list(dictionaryonehotencodemata)
    katatengah = listonehotencodemata[np.argmax(yj)]
    listkatatengah = []
    i=0
    while i <= 5 :
        i+=1
        listkatatengah.append(listonehotencodemata[np.argmax(yj)])
        yj[np.argmax(yj)] = 0
    #print("kata tengah merupakan : ", katatengah)
    #print("5 kata tengah tertinggi merupakan : ", listkatatengah)
    #print("tekan enter")
    #input()

```

```

    elif onehotencodekata is None :
        print("kata " + "\"" + katainput[0] + "\"" + " tidak ditemukan")
        print()
        quit()
    elif onehotencodekata2 is None :
        print("kata " + "\"" + katainput[1] + "\"" + " tidak ditemukan")
        print()
        quit()

sourcecursor.execute("SELECT sumber_url,content FROM text")

dictionarykemunculan= {}
dictionarykatavalidrelevan = {}

listtext = sourcecursor.fetchall()
#listtext[index][kolom]
nilaidokumen = 0

for i in range(len(listtext)):

    #cari kata valid di kolom content dengan menggunakan regular expression
    #itung jumlah kemunculan
    satudua = katainput[0] + " " + katabengah + " " + katainput[1]
    duasatu = katainput[1] + " " + katabengah + " " + katainput[0]
    jumlahkemunculansatudua = len(re.findall('\b'+ satudua +'\b',
    ,listtext[i][1]))
    jumlahkemunculanduasatu = len(re.findall('\b'+ duasatu +'\b',
    ,listtext[i][1]))

    #taro [katainputvalid : jumlah kemunculan] pada dictionarykatavalidrelevan
    dictionarykatavalidrelevan[satudua] = (jumlahkemunculansatudua)
    dictionarykatavalidrelevan[duasatu] = (jumlahkemunculanduasatu)
    nilaidokumen = 1 * jumlahkemunculansatudua + 1
    * jumlahkemunculanduasatu
    dictionarykatavalidrelevan["nilai dokumen"] = nilaidokumen

    #bikin dictionary nested [sumber : {kata input : jumlah, kata relevan : jumlah, nilai dokumen : nilai}]
    dictionarykemunculan[listtext[i][0]
    ]] = dictionarykatavalidrelevan
    dictionarykatavalidrelevan= {}

```

```
listdictionarykemunculan = list(dictionarykemunculan)
limadokumenpalingrelevan = sorted(dictionarykemunculan.items(), key
=
lambda item: item[1]["nilai dokumen"], reverse=True)[0:5]
for i in range(len(limadokumenpalingrelevan)):
    if limadokumenpalingrelevan[i][1]["nilai dokumen"] == 0 :

#print("Hasil dokumen :", limadokumenpalingrelevan[random.randrange
(1,70)][0])
    print("Hasil dokumen :"
, listdictionarykemunculan[random.randrange(1,70)])
else :
#print("Hasil dokumen :", limadokumenpalingrelevan[i][0])
    print(limadokumenpalingrelevan[i])

print()

waktuJalan = (time.time() - startAwalTime)
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))
```

Lampiran 7 Source Code search4cbow.py

```

import mysql.connector
import numpy as np
import time
import re
import random

katainput = input('Masukkan kata \n')
print()

sourcedb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="skripsi"
)

sourcecursor = sourcedb.cursor()

sourcecursor.execute(
"SELECT kata, one_hot_encode, vector_cbow_4kata FROM dictionary")

myresult = sourcecursor.fetchall()

dictionaryonehotencodekata = {}
#bikin dictionary kata dan vektor skip gram

sourcecursor.execute(
"SELECT matrix_weight_input,matrix_weight_output,kata_unik,hidden_
layer FROM weight WHERE metode='cbow4kata' GROUP BY ID = (SELECT M
AX(ID) FROM weight)"
)
weight = sourcecursor.fetchone()
matrixweightinput = weight[0]
matrixweightinput = matrixweightinput.replace("[", "")
matrixweightinput = matrixweightinput.replace("]", "")
#weight = kataunik x hiddenlayer
matrixweightinput = np.fromstring(matrixweightinput.strip('\n'
),sep=' ').reshape((weight[2], weight[3]))

matrixweightoutput = weight[1]
matrixweightoutput = matrixweightoutput.replace("[", "")
matrixweightoutput = matrixweightoutput.replace("]", "")
#weight = hiddenlayer x kataunik
matrixweightoutput = np.fromstring(matrixweightoutput.strip('\n'
),sep=' ').reshape((weight[3], weight[2]))

```

```

for i in range(len(myresult)):
    onehotencode = myresult[i][1]
    onehotencode = onehotencode.replace(" ", "")
    onehotencode = onehotencode.replace(",", "")
    onehotencode = onehotencode.replace("[", "")
    onehotencode = onehotencode.replace("]", "")
    onehotencode = np.asarray(list(onehotencode)).astype(np.float64)
    dictionaryonehotencodemata[myresult[i][0]] = onehotencode

startAwalTime = time.time()
katainput = katainput.split()
if len(katainput) != 4 :
    print(
"jumlah kata yang dimasukkan tidak sesuai, silahkan jalankan ulang
program"
)
    print()
    quit()

onehotencodemata = dictionaryonehotencodemata.get(katainput[0])
onehotencodemata2 = dictionaryonehotencodemata.get(katainput[1])
onehotencodemata3 = dictionaryonehotencodemata.get(katainput[2])
onehotencodemata4 = dictionaryonehotencodemata.get(katainput[3])
if onehotencodemata is not None and onehotencodemata2 is not None:
    katainput[0]
    katainput[1]
    hiddenlayer = (1/4
) * np.dot(np.transpose(matrixweightinput),(onehotencodemata + one
hotencodemata2 + onehotencodemata3 +onehotencodemata4))
    uj = np.dot(np.transpose(matrixweightoutput),hiddenlayer)
    expuj = np.exp(uj - np.max(uj))
    yj = expuj/np.sum(expuj)
    listonehotencodemata = list(dictionaryonehotencodemata)
    katatengah = listonehotencodemata[np.argmax(yj)]
    listkatatengah = []
    i=0
    while i <= 5 :
        i+=1
        listkatatengah.append(listonehotencodemata[np.argmax(yj)])
        yj[np.argmax(yj)] = 0
    print("kata tengah merupakan : ", katatengah)
    print("5 kata tengah tertinggi merupakan : ", listkatatengah)
    #print("tekan enter")
    #input()

```

```

    elif onehotencodekata is None :
        print("kata " + "\"" + kataininput[0] + "\"" + " tidak ditemukan")
        print()
        quit()
    elif onehotencodekata2 is None :
        print("kata " + "\"" + kataininput[1] + "\"" + " tidak ditemukan")
        print()
        quit()
    elif onehotencodekata2 is None :
        print("kata " + "\"" + kataininput[2] + "\"" + " tidak ditemukan")
        print()
        quit()
    elif onehotencodekata2 is None :
        print("kata " + "\"" + kataininput[3] + "\"" + " tidak ditemukan")
        print()
        quit()

sourcecursor.execute("SELECT sumber_url,content FROM text")

dictionarykemunculan= {}
dictionarykatavalidrelevan = {}

listtext = sourcecursor.fetchall()
#listtext[index][kolom]
nilaidokumen = 0

for i in range(len(listtext)):

    #cari kata valid di kolom content dengan menggunakan regular expression
    #itung jumlah kemunculan
    katagabungan = kataininput[0] + " " + kataininput[1] + " "
    + katatengah + " " + kataininput[2] + " " + kataininput[3]
    jumlahkemunculangabungan = len(re.findall('\b'+ katagabungan +
    '\b',listtext[i][1]))

    #taro [katainputvalid : jumlah kemunculan] pada dictionarykatavalidrelevan
    dictionarykatavalidrelevan[katagabungan] = (jumlahkemunculangabungan)
    nilaidokumen = 1 * jumlahkemunculangabungan
    dictionarykatavalidrelevan["nilai dokumen"] = nilaidokumen

```

```
#bikin dictionary nested [sumber : {kata input : jumlah, kata relevan : jumlah, nilai dokumen : nilai}]
    dictionarykemunculan[listtext[i][0
]] = dictionarykatavalidrelevan
    dictionarykatavalidrelevan= {}

listdictionarykemunculan = list(dictionarykemunculan)
limadokumenpalingrelevan = sorted(dictionarykemunculan.items(), key=
y=
lambda item: item[1]["nilai dokumen"], reverse=True)[0:5]
r = list(range(5))
random.shuffle(r)
for i in r:
    if limadokumenpalingrelevan[i][1]["nilai dokumen"] == 0 :

#print("Hasil dokumen :", limadokumenpalingrelevan[random.randrange(1,70)][0])
    print("Hasil dokumen :"
, listdictionarykemunculan[random.randrange(1,70)])
else :
    #print("Hasil dokumen :", limadokumenpalingrelevan[i][0])
    print(limadokumenpalingrelevan[i])

print()

waktuJalan = (time.time() - startAwalTime)
print('Waktu total yang dibutuhkan : ' + str(waktuJalan))
```

RIWAYAT HIDUP

MUHAMMAD ZALGHORNAIN. Lahir di Jakarta, 19 Mei 1997, merupakan anak kedua dari pasangan Bapak Ir. Mochammad Nadjib Bachmid dan Ibu Wirdalita, S.Sos.

Penulis mengawali pendidikan di TK Islam Assalam II, Tambun, Bekasi pada tahun 2002-2003. Kemudian melanjutkan pendidikan di SD Jaya Suti Abadi, Tambun, Bekasi pada tahun 2003-2009. Selanjutnya penulis melanjutkan studi di SMPN 182, Pancoran, Jakarta Selatan pada tahun 2009-2012. Kemudian meneruskan ke SMAN 14, Jakarta Timur pada tahun 2012-2015. Pada tahun 2015 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ) melalui jalur SBMPTN. Di awal tahun 2023 (14 Februari 2023) penulis dinyatakan lulus dan mendapat gelar Sarjana Komputer (S.Kom), Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta.