Seminar Nasional Sains Data 2024 (SENADA 2024)
UPN "Veteran" Jawa Timur
E-ISSN 2808-5841
P-ISSN 2808-7283

Analisis Perbandingan FastText dan Word2Vec pada Sistem Temu Balik Informasi

Rosni Lumbantoruan[1], Maria Puspita Sari Nababan[2], Letare Aiglien Saragih[3]
[1, 2, 3]*Sistem Informasi, Institut Teknologi Del*
[1]rosni@del.ac.id
[2]iss18064@students.del.ac.id
[3]iss18019@students.del.ac.id
Corresponding author email: rosni@del.ac.id

Abstract: *Information retrieval with the approach of machine learning commonly depends on the usage of word-embedding in representing the natural language for both documents and user's query. Thus, applying the precise word-embedding becomes an essential factor in increasing the performance of the information retrieval system. In this research, we compare two commonly used approaches of word-embedding, FastText and Word2Vec. We conduct these two approaches using two different datasets namely Internet News and Movie Plots. The results show that there are different characteristics of each embedding where FastText gets the advantage of n-gram to get more variation of texts to capture more similar words. Meanwhile, Word2Vec commonly find the similar text based on their co-appearance on the documents.*
Keywords: *word-embedding, FastText, Word2Vex, Information Retrieval, Semantical meaning*

Abstrak: Sistem temu balik informasi dengan menggunakan pendekatan pembelajaran mesin pada umumnya memanfaatkan *word embedding* dalam merepresentasikan dokumen dan kueri pengguna. Pemilihan *word embedding* menjadi salah satu faktor kunci yang mempengaruhi kinerja sistem temu balik informasi, khususnya untuk mengolah teks atau kalimat dengan karakteristik data yang tidak terstruktur. Pada penelitian ini, *word embedding* yang paling sering digunakan yaitu FastText dan Word2Vec dibandingkan dalam hal menangkap dan mengembalikan makna semantik kata. Pada penelitian ini, eksperimen untuk membandingkan kedua pendekatan dilakukan dengan menerapkan masing-masing pendekatan pada dua dataset yang berbeda yaitu Internet News dan Movie Plots. Hasil eksperimen menunjukkan bahwa kedua pendekatan memiliki karakteristik masinng-masing, Dimana FastText dengan bantuan representasi kata dengan n-gram mampu menangkap kata yang memiliki kesamaan dari sisi susunan karakter sedangkan Word2Vec mencari kemiripan dengan kata lain berdasarkan keseringan kata tersebut muncul secara bersamaan dengan kata lain pada dokumen.
Kata kunci: *word embedding*, FastText, Word2Vec, sistem temu balik informasi, makna semantik

I. PENDAHULUAN

Pada sistem temu balik informasi, *word embedding* digunakan untuk merepresentasikan kedekatan sebuah kata dengan kata lain pada dokumen atau kueri. Pada umumnya, kedekatan kata merepresentasikan kemiripan kata berdasarkan makna kata baik semantik maupun kontekstual. Contoh kedekatan semantik antara kata kendaraan dengan alat transportasi atau secara kontekstual kedekatan antara mobil dengan pesawat terbang. Kemiripan antar kata akan sangat bermanfaat pada sistem temu balik informasi, khususnya untuk mencari kedekatan dokumen dengan kata pada kueri yang diberikan oleh pengguna. Dengan demikian, sistem temu balik informasi mampu mengembalikan dokumen yang paling mirip dengan kueri yang diberikan oleh pengguna [1].

Proses pencarian kemiripan antara kueri dengan dokumen menjadi salah satu komponen paling vital pada sistem temu balik informasi, diluar pengindeksan dan penyimpanan data. Ditambah lagi, sistem temu balik informasi pada umumnya berkaitan dengan data teks yang tidak terstruktur, sehingga peranan *word embedding* untuk mencari hubungan dan kedekatan antar kata menjadi sangat penting bagi para peneliti sistem temu balik karena mampu memberikan dampak yang positif dalam meningkatkan kinerja pengembalian dokumen [2]. Misalnya, pengguna menggunakan kata "bulan purnama", maka dokumen yang dikembalikan oleh sistem temu balik informasi seyogyanya adalah dokumen-dokumen yang mengandung kata "bulan purnama" atau dokumen yang relevan dengan kedua kata tersebut, misalnya yang mengandung kata "bulan penuh", "fase bulan".

Dalam penerapannya terdapat beberapa pendekatan *word embedding,* diantaranya adalah FastText dan Word2Vec. Pada penelitian ini, kedua *word-embedding* ini akan dibandingkan kinerjanya dalam menangkap hubungan antar kata dan mengembalikan makna semantik kata yang sesuai dengan inputan yang diberikan. Secara garis besar, Word2Vec menggunakan sekumpulan teks yang besar (korpus) sebagai data latih untuk membangun *vocabulary* dan menghasilkan ruang vektor yag dapat berjumlah beberapa ratus dimensi, dengan setiap kata unik pada korpus berupa vektor [3]. Salah satu kelemahan dari Word2Vec adalah ketidakmampuannya untuk merepresentasikan kata yang tidak pernah dijumpai sebelumnya pada teks atau data yang digunakan. Kelemahan ini dapat ditangani dengan FastText, dengan menyimpan kata dalam bentu *n-gram* sehingga dihasilkan representasi kata yang lebih beragam dalam bentuk vektor [3].

Dengan demikian, maka kontribusi dari penelitian ini adalah melakukan analisis perbandingan dari *word embedding*, FastText dan Word2Vec, dalam mengembalikan teks atau dokumen yang sesuai dengan kueri yang diberikan berdasarkan kedekatannya dari makna semantik dan makna kontekstualnya.

Selengkapnya, penelitian ini akan membahas tentang penelitian yang relevan dengan perbandingan Word2Vec dan Fastext pada Bab 2, dilanjutkan dengan metodologi penelitian pada Bab 3. Bab IV akan membahas terkait skenario eksperimen dan bagaimana eksperimen dijalankan serta pada Bab V akan dibahas hasil dan pembahasan. Sebagai penutup, Bab VII akan membahas kesimpulan dan saran untuk penelitian berikutnya.

II. PENELITIAN TERKAIT

Pada bagian ini akan dijelaskan penelitian terkait untuk metode yang akan digunakan pada penelitian yaitu FastText dan Word2Vec.

FastText merupakan *word vector open source* untuk kategorisasi teks dengan pembelajaran tersupervisi atau *supervised learning*. FastText tidak hanya memperhitungkan elemen seperti tata bahasa dan urutan kata dari teks tetapi mampu menangani ketidakseimbangan sampel [4]. FastText menghasilkan representasi vektor untuk kata maupun frasa pada *input layer* yang kemudian dipetakan selanjutnya pada *hidden layer* melalui transformasi linier. Pada penelitian [5], dilakukan perbandingan antara Word2Vec dan FastText dalam memengaruhi kinerja pencarian informasi sistem temu balik informasi. Pada percobaan yang dilakukan, diperoleh hasil yang menunjukkan bahwa performa FastText lebih baik untuk data koleksi eksternal dan sebaliknya, Word2Vec umumnya bekerja dengan baik pada data yang terlebih dahulu dikembalikan ke kata dasarnya atau *stemmed collection.* Hal ini berbanding terbalik dengan FastText. Dengan demikian disimpulkan bahwa dengan karakteristik yang melekat pada algoritma *embbeding* dimana FastText terlihat berkinerja lebih baik untuk tugas-tugas lain ketika dilatih tentang koleksi yang belum diproses dan Word2Vec lebih baik untuk tugas-tugas yang memerlukan pemrosesan data terlebih dahulu, khususnya *stemming*.

Pada percobaan lain [6], dilakukan perbandingan antara FastText dan Word2Vec dalam kasus *Named Entity Recognition* (NER). Hasil yang diamati bahwa terdapat perbedaan antara kinerja dari teknik *word embedding* yang digunakan. Secara khusus, Word2vec menunjukkan kinerja yang lebih baik daripada FastText dan Glove untuk kedua teknik *deep learning*, menggunakan NER untuk bahasa Urdu. Perbandingan antara FastText dan Word2Vec juga dilakukan pada penelitian [5] dimana pada kasus ini dilakukan untuk melihat kinerja kedua teknik tersebut untuk klasifikasi teks. Setiap teknik diimplementasikan pada model yang digunakan untuk klasifikasi teks dan dinilai berdasarkan hasil evaluasi dari model tersebut. Hasil yang diperoleh berasal dari evaluasi yang telah dilatih menggunakan

precision (P), recall (R), dan F-Measure (F). Kinerja model dievaluasi dengan menggunakan dua dataset yang berbeda yaitu 20 newsgroup dan Reuters. Hasil menunjukkan model yang menggunakan Word2Vec tidak mampu merepresentasikan vektor dari kata yang tidak ada dalam korpus (*out of vocabulary*) sedangkan FastText menunjukkan keandalannya untuk menangani permasalahan *out of vocabulary* dengan mengembalikan kata yang relevan atau masih berkaitan dengan kata tersebut.

Walaupun FastText menunjukkan kinerja yang paling baik, namun perbedaan kinerja tersebut tidak signifikan, sehingga disimpulkan teknik *word embedding* memiliki kinerja yang kompetitif dan bergantung terhadap dataset yang digunakan. Pada penelitian ini, kami akan membahas lebih dalam penyebab perbedaan kinerja dari kedua *word embedding*, Word2Vec dan FastText serta bagaimana kedua teknik tersebut berbeda dalam mencari kata yang mirip pada dataset yang diberikan.

III. METODOLOGI PENELITIAN

Pada bab ini dijelaskan terkait kedua pendekatan *word embedding* yang digunakan yaitu Word2Vec dan FastText, mulai dari tahapan persiapan data yaitu teknik *text preprocessing* yang digunakan, penggunaan *word embedding* untuk merepresentasikan kata yang ada pada dokumen dan kueri. Kemudian, dijelaskan tentang dua percobaan yang membandingkan hasil yang diperoleh dari penggunaan kedua metode *word embedding* tersebut.

III.1. Dataset

Relasi antar kata dengan penerapan *word embedding* pada penelitian ini akan ditelusuri dengan menggunakan dua dataset yaitu: 1) Internet News data with Readers Engagement dan; 2) Wikipedia Movie Plots. Kedekatan antar kata dihitung menggunakan *cosine similarity*. Setiap dataset memiliki konteks dan jumlah atribut yang berbeda yang diuraikan sebagai berikut.

Internet News Data with Readers Engagement[1], berisi artikel dari beberapa penerbit terkenal dan terdaftar berdasarkan popularitas di situs web penerbit. Dataset ini terdiri dari 15 atribut dengan jumlah data sebanyak 23.535 data.

Wikipedia Movie Plots[2], berisi dattentang ringkasan deskripsi plot yang diambil dari Wikipedia. Dataset berisi deskripsi 34.886 film dari seluruh dunia. Data yang digunakan dari dataset ini adalah atribut yang berisi teks panjang dan akan digunakan pada *clustering*dokumen. Lebih detail terkait kedua data ini dirangkumkan pada table 1 berikut.

Tabel 1. Dataset

Dataset	Atribut	Total Data	Size (MB)
Internet News Data with Readers Engagement	15	23.535	7,69
Wikipedia Movie Plots	8	34.886	81,19

III.2. Data and Text Preprocessing

Data yang ditemukan dalam kehidupan nyata seringkali tidak lengkap, tidak konsisten, atau terdapat banyak *noise*. Hal ini dapat membuat pemahaman data menjadi sulit. Untuk itu diperlukan suatu proses yang dapat digunakan untuk mengubah data mentah menjadi format yang dapat dipahami, yaitu melalui teknik *data preprocessing*. Teknik ini mengacu pada beberapa langkah yang dilakukan untuk meningkatkan kualitas data mentah, dan juga sering diperlukan untuk memastikan validitas dan

[1] https://www.kaggle.com/datasets/szymonjanowski/internet-articles-data-with-users-engagement/

[2] https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots

reliabilitas hasil analisis data. Tahap *data preprocessing* yang dilakukan pada penelitian ini adalah sebagai berikut:

III.2.1. Data Cleaning

Proses ini dilakukan untuk mengatasi data yang tidak lengkap, tidak sempurna, atau salah. Pembersihan data dapat dilakukan dengan berbagai cara, yaitu dengan mengidentifikasi apakah ada nilai yang hilang sehingga dapat disesuaikan. Teknik lain adalah menghapus nilai duplikat (nilai yang sama) atau menghapus kolom tertentu yang tidak diperlukan.

III.2.2. Data Reduction

Proses ini dilakukan untuk mereduksi data yang dimiliki sehingga hanya data yang diperlukan saja yang diperoleh. Salah satu cara yang dapat dilakukan untuk mereduksi data adalah dengan menghilangkan atribut-atribut yang tidak digunakan dalam pemodelan.

Text Preprocessing merupakan tahapan yang digunakan untuk mengolah data, khususnya data dalam bentuk teks. Beberapa operasi yang dilakukan dengan memanfaatkan data teks memerlukan *tahapan preprocessing* yang bertujuan untuk menyusun teks dan mengekstraksi fitur. Selain itu, tujuan dari *tahapan preprocessing* ini adalah untuk mempersiapkan representasi teks sehingga dapat menghasilkan model yang baik melalui data yang bersih. Penggunaan tahapan *preprocessing* dapat disesuaikan dengan kebutuhan model serta pencapaian data yang akan dibersihkan dan diolah. Beberapa tahapan *preprocessing* yang akan dilakukan adalah sebagai berikut [7]:

III.2.3. Case Folding

Operasi ini digunakan untuk mengubah semua huruf menjadi huruf kecil. Hal ini bertujuan untuk menghindari pendefinisian 2 kata yang sama menjadi berbeda karena perbedaan penggunaan huruf kecil dan huruf besar dalam teks.

III.2.4. Stop-word Removal

Dalam berkomunikasi, terutama melalui teks, kata-kata tertentu sering digunakan untuk melengkapi kalimat, tetapi jika diterjemahkan dalam arti yang terpisah, mereka tidak memiliki arti apa pun. Kata-kata ini disebut sebagai *stopwords*. Contohnya adalah penggunaan "the", "and" atau "a". Operasi penghapusan *stopwords* dapat digunakan untuk menghilangkan kata-kata yang termasuk dalam *stopwords* sehingga memungkinkan untuk lebih fokus pada kata-kata yang lebih penting dan bermakna.

III.2.5. Tokenization

Teks yang akan dipelajari perlu dipecah dari kalimat ke kata demi kata. Operasi ini disebut dengan *tokenization. Tokenization* diperlukan untuk memotong teks menjadi bagian-bagian yang lebih kecil yang disebut token.

III.2.6. Lemmatization

Operasi ini merupakan proses yang termasuk dalam *stemming* tetapi didasarkan pada kamus dengan memanfaatkan kosakata kata untuk menghilangkan imbuhan dari teks sehingga dapat kembali ke bentuk dasarnya.

Tabel 2 berikut menampilkan data setelah melalui data dan *text-preprocessing*. Pengurangan dataset terjadi sebagai akibat dari penghapusan data duplikat, data dengan nilai nol dan penghapusan kolom yang tidak diperlukan dalam penelitian.

Tabel 2. Dataset setelah data dan *text-preprocessing*

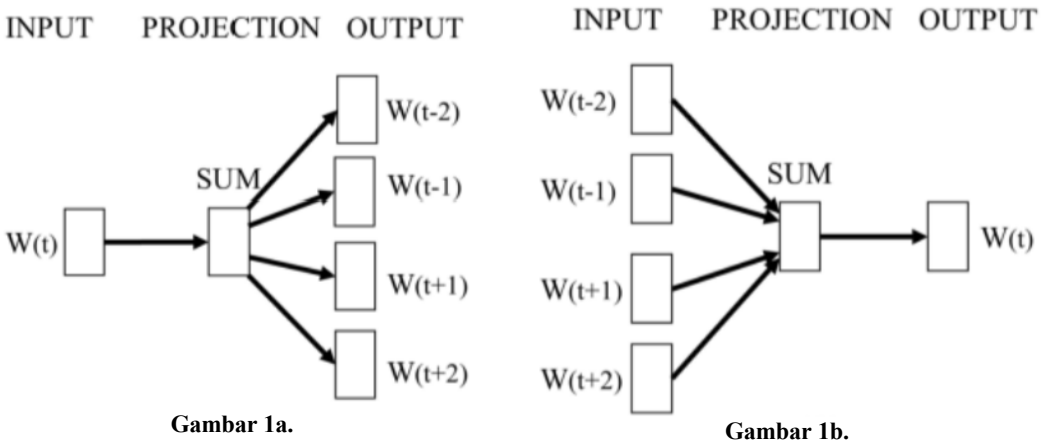
Dataset	Atribut	Total Data
Internet News Data with Readers Engagement	1	9.145
Wikipedia Movie Plots	1	33.869

III.3. Word Embedding

Kata yang telah melalui tahapan *text preprocessing* selanjutnya akan diolah untuk mendapatkan hasil dalam bentuk vektor. Metode yang digunakan untuk mengubah kata menjadi representasi vektor adalah *word embedding*. Teknik *word embedding* yang digunakan pada penelitian ini adalah FastText dan Word2Vec. Data yang akan digunakan dalam *word* embedding berasal dari hasil akhir *text preprocessing*. Setiap kata akan ditampilkan dalam potongan yang disebut token. *Word embedding* mengubah setiap token kedalam bentuk vektor. Terdapat beberapa penelitian sebelunnya yang telah melakukan perbandingan antara Word2Vec dan FastText, diantaranya [8], [9].

FastText merupakan perpanjangan dari *library* Word2vec yang didasarkan pada kontinuitas skip-gram dimana setiap kata akan direpresentasikan sebagai karakter n-gram. Arsitektur skip gram *negative sampling* (SGNS) dari FastText dapat mengidentifikasi kata-kata konteks yang diberikan kata target sehingga kata-kata tersebut akan dipecah menjadi beberapa n-gram dan kemudian menjadi vektor dengan masing-masing n-gram. Jumlah representasi vektor dari semua n-gram atau rata-ratanya akan mewakili kata legenda. Algoritma FastText mengasumsikan sebuah kata terdiri dari karakter n-gram dan panjang *n* dapat berubah dari satu menjadi panjang kata. Pendekatan FastText, yang menyimpan vektor kata sebagai karakter n-gram, memungkinkan metode ini untuk menemukan representasi vektor untuk kata-kata yang tidak langsung ditemukan dalam kamus.

Berikutnya adalah metode Word2vec dimana pada metode ini terdapat 2 arsitektur yaitu Skip-gram dan CBOW [10], [11].



Gambar 1a. **Gambar 1b.**

Gambar 1. merupakan contoh gambar ganda : (1a) Arsitektur Skip-Gram (1b) Arsitektur CBOW

Bagian a dari gambar 1 menunjukkan arsitektur Skip-gram Word2Vec yang bertujuan untuk memprediksi konteks (*output*) di sekitar kata saat ini (*input*). Data masukan Skip-gram dalam bentuk *n-hot encoded vector*. Saat proses pelatihan berjalan, kata yang diinput akan bernilai 1 sedangkan kata lainnya akan bernilai 0. Karena targetnya hanya 1 kata, maka output target akan berupa *one-hot encoded vector*.

Secara arsitektural, Skip-Gram menggunakan kata-kata saat ini (sebagai *input*) untuk memprediksi konteks sekitarnya (sebagai *target*), dimana Skip-Gram akan mempelajari distribusi probabilitas kata-

kata dalam konteks dengan *windows size* yang telah ditentukan. Misalnya, konteks yang digunakan saat ini adalah "the best revenge is massive success" dengan ukuran *windows* = 2. Untuk merepresentasikan konteks ke dalam arsitektur Skip-Gram, kita harus mengubah setiap kata menjadi vektor *one-hot encoding* seperti yang ditunjukkan pada tabel 3 berikut.

Tabel 3. Vektor kata dengan *one-hot encoding*

Term	Vektor
the	[1,0,0,0,0,0]
best	[0,1,0,0,0,0]
revenge	[0,0,1,0,0,0]
is	[0,0,0,1,0,0]
massive	[0,0,0,0,1,0]
success	[0,0,0,0,0,1]

Gambar 1 bagian b menunjukkan arsitektur Word2Vec CBOW yang merupakan kebalikan dari skip-gram Word2vec. Tujuannya adalah untuk memprediksi kata (*output*) ketika diberikan konteks di sekitar kata (*input*). Prosesnya tidak jauh berbeda dengan arsitektur Skip-gram. Perbedaannya hanya terletak pada input data berupa *n-hot encoded vector* dan target *output* berupa *one-hot encoded vector*. Setelah training hingga mencapai error minimum, kita dapat mengambil vektor representasi kata dengan mengalikan *one-hot encoded vector* dari setiap kata dengan bobot W.

IV. METODE EVALUASI

Eksperimen yang dilakukan bertujuan untuk melihat relasi antar kata pada dokumen yang akan dicoba melalui beberapa eksperimen. Setiap percobaan memiliki proses dan hasil tersendiri. Pada bagian ini akan dijelaskan komponen dan hasil yang diperoleh dari percobaan yang dilakukan. Ada dua bagian yaitu lingkungan percobaan yang menggambarkan dataset dan metode yang akan di-gunakan, dan hasil percobaan yang menggambarkan hasil dari setiap percobaan. Tabel 4 menampilkan metode eksperimen yang akan dilakukan pada penelitian ini.

Tabel 4. Metode eksperimen

Dataset	Metode	Text Preprocessing
Internet News	FastText	Yes
	Word2Vec	Yes
Movie Plots	FastText	Yes
	Word2Vec	Yes

Daftar dengan nomor dapat ditambahkan sebagai berikut:

1. Penelitian ini akan menggunakan 2 jenis dataset, yaitu: dataset Internet News dan dataset Movie Plots. Kedua set data akan diproses dan dimodelkan secara terpisah. Setiap set data akan diproses terlebih dahulu untuk memastikan data bersih dan cukup baik untuk digunakan. Adapun rangkaian *preprocessing* data dan teks yang dilakukan antara lain *removing duplicate data*, *removing Null values*, *removing punctuation*, *tokenization*, *stop-word removal*, *lemmatization*.
2. Pada percobaan menggunakan FastText, dataset yang telah diproses pada data and text preprocessing selanjutnya akan digunakan sebagai input pada layer FastText. FastText akan mengubah teks input menjadi vektor;

3. Pada percobaan menggunakan Word2Vec, dataset yang telah diproses pada data and *text preprocessing*, selanjutnya akan digunakan sebagai input pada layer Word2Vec. Word2Vec akan mengubah teks masukan menjadi vektor.

V. HASIL EKSPERIMEN

Pengaruh dari *word embedding* Word2Vec dan FastText dibandingkan dengan penerapan kedua *word embedding* pada dua dataset yang berbeda yaitu Internet News dan Movie Plots. Kedua dataset terlebih dahulu melalui data dan *text preprocessing*. Hasil dan pembahasan dari perbandingan kedua metode tersebut dapat dilihat sub-bab berikut.

V.1. Perbandingan FastText dan Word2Vec pada dataset Internet News

Tabel 5. FastText vs Word2Vec pada dataset Internet News

Contoh kata	Metode	Kata dengan makna semantik terdekat	Kedekatan dengan *cosine similarity*
moon	FastText	typhoon	0,999
		Typhoon	0,999
		Pokemon	0,999
		Gordon	0,998
		Solomon	0,998
	Word2Vec	different	0,999
		making	0,999
		usually	0,999
		sometimes	0,999
		problem	0,999

Pada tabel 5 dapat dilihat bahwa FastText dapat menemukan kata-kata yang lebih mirip dengan kata "moon" dibandingkan Word2Vec. Dari hasil penerapan model FastText dianalisis bahwa pengembalian kata diperoleh dari hasil pemecahan kata menjadi karakter bi-gram [3].Kata-kata yang dimunculkan berasal dari proses yang menggunakan karakter n-gram untuk mewakil setiap *embedding*. Pada proses ini digunakan parameter *min n* dan *max n*, dimana *min n* mendefinisikan karakter minimum n-gram dan *max n* mewakili karakter maksimum n-gram. Sehingga kata yang dikembalikan dianggap sebagai kata yang paling mirip dengan inputan kueri yang diberikan. Untuk menemukan kata yang terdekat perlu dilakukan penghitungan skor kesamaan antar kata. Kata-kata yang diwakili oleh vektor kata kontinu sehingga dapat diterapkan *similarity* untuk kata tersebut. Secara khusus digunakan kosinus sudut antara dua vektor. Kemiripan ini dihitung untuk semua kata dalam kosakata, dan 5 kata yang paling mirip akan ditampilkan. Jika kata tersebut muncul dalam kosa kata, kata tersebut akan muncul di atas. Misalnya kata "moon", dipecah menjadi karakter bi-gram: "mo", "oo", dan "on". Kemudian model akan mencari kata kata yang mengandung karakter "mo", "oo", dan "on" pada daftar kata dan kamus FasText dan mengembalikan kata -kata dengan nilai vektor tertinggi.

Berdasarkan hasil yang diperoleh dari perbandingan model 1 dan model 2, dapat dilihat bahwa pada model yang menggunakan Fast Text akan mengembalikan term yang berkaitan dengan karakter n-gram dari term inputan dan berkaitan dengan konteks dari term tersebut. Contohnya untuk term "moon" memiliki kemiripan teratas dengan term "typhoon". Jika dilihat dari segi karakter n-gram, baik term "typhoon" dan "moon" memiliki kesamaan karakter n-gram yaitu "oon". Sedangkan jika dilihat dari segi konteks, term "typhoon" dan "moon" memiliki hubungan dimana penggabungan kedua term tersebut memiliki makna yang mengacu pada objek tertentu. Sementara dari hasil penerapan model Word2Vec dianalisis bahwa pengembalian kata diperoleh dari hasil pengecekan kata sebelum dan sesudah dari posisi kueri pada dokumen atau teks [3]. Misalnya terdapat kueri "moon", kemudian pada

dokumen terdapat teks "moon usually looks beautiful". Maka Word2Vec akan mengeksplorasi isi dokumen untuk mencari letak kata "moon" untuk kemudian mengembalikan kata sesudah atau sebelum kata yang dicari. Dalam hal ini "usually" akan menjadi output dari kueri.

V.2. Perbandingan FastText dan Word2Vec pada dataset Movie Plots

Selanjutnya tabel 6 menampilkan perbandingan FastText dan Word2Vec pada dataset Movie Plots.

Tabel 6. FastText vs Word2Vec pada dataset Movie Plots

Contoh kata	Metode	Kata dengan makna semantik terdekat	Kedekatan dengan *cosine similarity*
moon	FastText	full moon	0,743
		moonlight	0,737
		moonlighting	0,713
		mooning	0,691
		eclipsed	0,687
	Word2Vec	eclipse	0,782
		cloud	0,772
		meteor	0,742
		dinosaur	0,735
		universe	0,731

Pembahasan perbandingan FastText dan Word2Vec sebagaimana pada subbab 5.1 dapat juga dilihat pada tabel 6 diatas. Dengan Word2Vec, kesamaan semantik dari kata dilihat dari hubungan kata yang pada kueri yaitu "moon" dengan kata lain yang sering muncul bersamaan dengan kata kueri tersebut. Misalnya, pada hasil ini diperoleh bahwa kata "eclipse", "cloud" sering muncul bersamaan dengan kata "moon" seperti kalimat "the cloud hides away the moon", sehingga kata "moon" dan "cloud" dianggap memiliki kedekatan. Sementara itu, pada FastText digunakan pemisahan dengan n-gram untuk mengindentifikasi bahwa moon dan moonlight masih memiliki karakter n-gram yang berdekatan yaitu kata "moon".

VI. KESIMPULAN

Berdasarkan hasil analisis eksperimen yang dilakukan pada kedua perbandingan metode *word embedding*, FastText dan Word2Vec dapat disimpulkan bahwa dari sisi hasil *embedding* output yang dihasilkan oleh kedua pendekatan berbeda. FastText mengasumsikan sebuah kata terdiri dari karakter n-gram dan panjang n dapat berubah dari satu menjadi panjang kata atau sesuai kebutuhan sistem. Pendekatan FastText, yang menyimpan vektor kata sebagai karakter n-gram, memungkinkan metode ini untuk menemukan representasi vektor untuk kata-kata yang tidak langsung ditemukan dalam kamus. Dilain sisi, Word2Vec memprediksi konteks kata dengan cara melihat kedekatan sebuah kata dengan kata lain yang posisinya berada di sebelum atau sesudah kata tersebut. *Word embedding* dapat digunakan untuk menggambarkan kedekatan sebuah kata atau dokumen namun kedekatan tersebut adalah kedekatan kontekstual sesuai dengan data latih yang digunakan sehingga seringkali kedekatan tersebut bukan merupakan makna sebuah kata.

Kedua *word embedding* dapat digunakan sesuai dengan kebutuhan dari aplikasi yang akan dibangun, apakah akan menggunakan FastText atau Word2Vec atau mungkin menggunakan penggabungan keduanya untuk dapat menangkap kedekatan kata berdasarkan kemiripan karakternya dan seringnya suatu kata dan kata lain muncul secara bersamaan pada dokumen.

UCAPAN TERIMA KASIH

Tim penulis mengucapkan terima kasih kepada Institut Teknologi Del melalui Lembaga Penelitian dan Pengabdian Masyarakat (LPPM) yang telah mendukung sepenuhnya berjalannya penelitian ini.

REFERENSI

1. E. S. Qorina, "Analisis perbandingan metode fast text dan word2vec pada query kesamaan semantik Sistem temu kembali informasi sirah nabawi," Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta.
2. D. Roy, D. Ganguly, S. Bhatia, S. Bedathur, and M. Mitra, "Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1835–1838, doi: 10.1145/3269206.3269277.
3. E. M. DHARMA, F. L. GAOL, H. L. H. S. WARNARS, and B. SOEWITO, "The Accuracy Comparison Among Word2vec, Glove, And Fasttext Towards Convolution Neural Network (CNN) Text Classification," *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 2, 2022.
4. K. Wang, Y. Cui, J. Hu, Y. Zhang, W. Zhao, and L. Feng, "Cyberbullying Detection, Based on the FastText and Word Similarity Schemes," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 20, no. 1, Nov. 2020, doi: 10.1145/3398191.
5. Nurdin, B. A. S. Aji, A. Bustamin, and Z. Abidin, "Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks," *J. Tekno Kompak*, vol. 14, no. 2, pp. 74–79, 2020.
6. S. Kanwal, K. Malik, K. Shahzad, F. Aslam, and Z. Nawaz, "Urdu Named Entity Recognition: Corpus Generation and Deep Learning Applications," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 19, no. 1, Jun. 2019, doi: 10.1145/3329710.
7. V. Raunak, V. Gupta, and F. Metze, "Effective dimensionality reduction for word embeddings," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 235–243.
8. S. Khomsah, R. D. Ramadhani, S. Wijaya, and others, "The accuracy comparison between Word2Vec and FastText On sentiment analysis of hotel reviews," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 6, no. 3, pp. 352–358, 2022.
9. R. Lumbantoruan, R. U. Siregar, I. Manik, N. Tambunan, and H. Simanjuntak, "Analysis comparison of FastText and Word2vec for detecting offensive language," in *2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM)*, 2022, pp. 1–8.
10. T. Menon, "Empirical analysis of CBOW and skip gram NLP models," 2020.
11. K. Choudhary and R. Beniwal, "Xplore Word Embedding Using CBOW Model and Skip-Gram Model," in *2021 7th International Conference on Signal Processing and Communication (ICSC)*, 2021, pp. 267–270.