



UNIVERSITAS INDONESIA

**ANALISIS KINERJA BERT SEBAGAI METODE
REPRESENTASI TEKS UNTUK *TEXT CLUSTERING***

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

**ALVIN SUBAKTI
1706031821**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2021**

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.



HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama : Alvin Subakti
NPM : 1706031821
Program Studi : Matematika
Judul Skripsi : Analisis Kinerja BERT sebagai Metode Representasi Teks untuk *Text Clustering*

Telah berhasil dipertahankan di hadapan Dewan Pengaji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia

DEWAN PENGUJI

Pembimbing I	: Dr.rer.nat. Hendri Murfi, S.Si., M.Kom.	()
Pembimbing II	: Dra. Nora Hariadi, M.Si.	()
Pengaji	: Dr. Kiki Ariyanti Sugeng	()
Pengaji	: Dr. Helen Burhan, S.Si., M.Si.	()

Ditetapkan di : Depok

Tanggal : 3 Agustus 2021

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala berkat dan penyertaan-Nya selama perkuliahan dan penulisan tugas akhir ini. Penulisan tugas akhir beserta perkuliahan selama ini tentunya dibantu oleh berbagai pihak. Dengan demikian, penulis ingin mengucapkan terima kasih terhadap bantuan yang telah diberikan oleh pihak tersebut, yaitu:

1. Bapak Dr. rer. nat. Hendri Murfi S.Si., M.Kom. selaku Dosen Pembimbing I. Terima kasih karena telah meluangkan waktu dan tenaga untuk membimbing dan mengarahkan penulis dari sejak pemilihan topik hingga penulisan tugas akhir ini selesai. Terima kasih atas semua ilmu, pemikiran, masukan, dan nasehat yang telah diberikan sehingga penulis dapat mengangkat dan menyelesaikan tugas akhir mengenai *text clustering* ini.
2. Ibu Dra. Nora Hariadi, M.Si. selaku Dosen Pembimbing II. Terima kasih karena telah meluangkan waktu dan tenaga untuk memberikan kritik dan masukan yang berharga selama penulisan tugas akhir ini. Terima kasih juga atas segala pengertian dan nasihat yang telah diberikan bahkan ketika penulis banyak melakukan kesalahan.
3. Ibu Dr. Kiki Ariyanti Sugeng dan Ibu Dr. Helen Burhan, S.Si., M.Si. selaku Dosen Penguji. Terima kasih atas waktu yang telah diluangkan untuk menguji dan memberikan saran bagi penulis, sehingga tugas akhir ini menjadi lebih baik lagi.
4. Bapak Rudy Hamidi, Ibu Ida Nomina, dan Felix Subakti selaku keluarga penulis. Terima kasih karena selalu memberikan dukungan dalam berbagai bentuk selama masa perkuliahan hingga masa penulisan tugas akhir.
5. Ibu Dr. Dra. Dian Lestari, D.E.A. selaku Kepala Departemen Matematika FMIPA UI. Terima kasih atas segala masukan dan dukungan yang diberikan ke penulis selama masa perkuliahan, terutama ketika penulis sedang mempersiapkan diri untuk mengikuti kegiatan pertukaran mahasiswa.
6. Bapak Dr. Hengki Tasman, S.Si., M.Si. selaku Kepala Program Studi Matematika FMIPA UI. Terima kasih atas segala masukan dan dukungan yang diberikan ke penulis selama masa perkuliahan, terutama ketika penulis sedang mempersiapkan diri untuk mengikuti kegiatan pertukaran mahasiswa.

7. Ibu Dra. Siti Aminah, M.Kom. selaku Pembimbing Akademis penulis. Terima kasih atas masukan dan dukungan yang diberikan sehingga rencana perkuliahan penulis dapat berjalan dengan baik.
8. Dosen-dosen Departemen Matematika FMIPA UI yang telah mengajarkan ilmu kepada penulis selama berkuliah di Departemen Matematika FMIPA UI. Terima kasih atas ilmu matematika yang sangat menarik dan berguna bagi penulis.
9. Juan Daniel selaku sahabat penulis yang paling ambisius. Terima kasih karena telah selalu menjadi teman bagi penulis sejak zaman mahasiswa baru. Terima kasih karena selalu bisa menjadi tempat untuk bercerita maupun bertengkar, dan selalu mau memberikan kritik dan bimbingan dalam berbagai hal, termasuk dalam penulisan tugas akhir ini.
10. Gita Kartika, Mega Fransiska, Yusril Rais Anwar, David Setiawan, dan Deni selaku teman-teman penulis. Terima kasih karena selalu mau membantu penulis ketika kesulitan dalam pekulianan. Terima kasih juga karena sudah menjadi teman yang ambisius sehingga juga memacu semangat penulis untuk menjadi ambisius.
11. Syarifah Ramadhan selaku sahabat se-apartemen penulis. Terima kasih karena telah menjadi teman cerita yang baik dan teman yang selalu memberikan dukungan dalam segala aktivitas.
12. Kak Rilo dan Kak Naufal selaku kakak tingkat penulis yang mengerjakan tugas akhir mengenai pendekstrian topik. Terima kasih telah membimbing penulis dan mau menjawab pertanyaan penulis ketika kesulitan memahami materi pendekstrian topik.
13. Semua pasangan dan teman lomba penulis yang tidak bisa disebutkan satu-satu. Terima kasih karena telah mau menemani penulis dalam mengikuti perlombaan sehingga penulis bisa mendapat pengalaman berharga.
14. Alva Andhika Sa'id dan Allissa Rahman serta teman-teman lain selaku teman magang penulis. Terima kasih karena sudah menjadi tim magang yang baik dan teman cerita di sela-sela mengerjakan tugas magang sehingga penulis mendapatkan banyak ilmu berharga.
15. Teman – teman di Discord yang tidak bisa disebutkan satu-satu. Terima kasih karena sering menemani malam penulis sehingga penulis tidak bosan di rumah

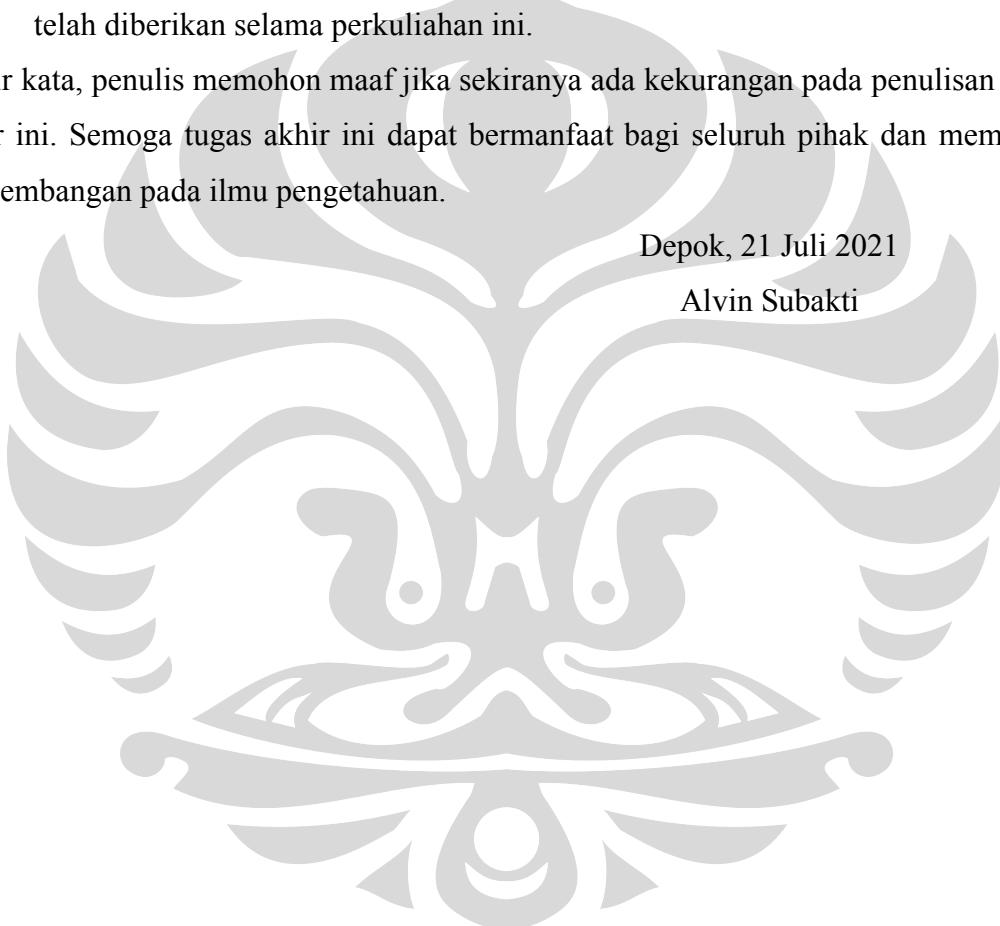
saja. Terima kasih karena bisa menjadi tempat untuk mencari hiburan sekaligus meminta masukan sehingga penulis bisa menyelesaikan tugas akhir ini.

16. Teman – teman dari tim asisten laboratorium komputasi Departemen Matematika FMIPA UI yang telah memberikan kesempatan kepada penulis untuk bergabung ke dalam tim sehingga penulis memiliki kesempatan untuk mengembangkan *skill*-nya dalam bidang komputasi dan akhirnya memotivasi penulis untuk mengajukan topik dalam bidang komputasi sebagai tugas akhir
17. Teman-teman INFIN17E. Terima kasih atas bantuan, hiburan, dan dukungan yang telah diberikan selama perkuliahan ini.

Akhir kata, penulis memohon maaf jika sekiranya ada kekurangan pada penulisan tugas akhir ini. Semoga tugas akhir ini dapat bermanfaat bagi seluruh pihak dan membawa pengembangan pada ilmu pengetahuan.

Depok, 21 Juli 2021

Alvin Subakti



HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Alvin Subakti

NPM : 1706031821

Program Studi : Matematika

Fakultas : Fakultas Matematika dan Ilmu Pengetahuan Alam

Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

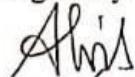
Analisis Kinerja BERT sebagai Metode Representasi Teks untuk Text Clustering beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal: 12 Juli 2021

Yang menyatakan



(Alvin Subakti)

ABSTRAK

Nama : Alvin Subakti
Program Studi : Matematika
Judul : Analisis Kinerja BERT sebagai Metode Representasi Teks untuk *Text Clustering*
Pembimbing : 1. Dr.rer.nat. Hendri Murfi, S.Si., M.Kom.
 2. Dra. Nora Hariadi, M.Si.

Text clustering adalah teknik pengelompokan teks sehingga teks di dalam kelompok yang sama memiliki tingkat similaritas yang lebih tinggi satu sama lain dibandingkan dengan teks pada kelompok yang berbeda. Proses pengelompokan teks secara manual membutuhkan waktu dan sumber daya yang banyak sehingga digunakan *machine learning* untuk melakukan pengelompokan secara otomatis. Representasi dari teks perlu diekstraksi sebelum dimasukkan ke dalam model *machine learning*. Metode yang umumnya digunakan untuk mengekstraksi representasi data teks adalah TFIDF. Namun, metode TFIDF memiliki kekurangan yaitu tidak memperhatikan posisi dan konteks penggunaan kata. Model BERT adalah model yang dapat menghasilkan representasi kata yang bergantung pada posisi dan konteks penggunaan suatu kata dalam kalimat. Penelitian ini menganalisis kinerja model BERT sebagai metode representasi data teks dengan membandingkan model BERT dengan TFIDF. Selain itu, penelitian ini juga mengimplementasikan dan membandingkan kinerja metode ekstraksi dan normalisasi fitur yang berbeda pada representasi teks yang dihasilkan model BERT. Metode ekstraksi fitur yang digunakan adalah *max* dan *mean pooling*. Sementara itu, metode normalisasi fitur yang digunakan adalah *identity*, *layer*, *standard*, dan *min-max normalization*. Representasi teks yang diperoleh dimasukkan ke dalam 4 algoritma *clustering* berbeda, yaitu *k-means clustering*, *eigenspace-based fuzzy c-means*, *deep embedded clustering*, dan *improved deep embedded clustering*. Kinerja representasi teks dievaluasi dengan menggunakan metrik *clustering accuracy*, *normalized mutual information*, dan *adjusted rand index*. Hasil simulasi menunjukkan representasi data teks yang dihasilkan model BERT mampu mengungguli representasi yang dihasilkan TFIDF pada 28 dari 36 metrik. Selain itu, implementasi ekstraksi dan normalisasi fitur yang berbeda pada model BERT memberikan kinerja yang berbeda-beda dan perlu disesuaikan dengan algoritma yang digunakan.

Kata Kunci:
BERT, Representasi Teks, *Text clustering*

ABSTRACT

Name	:	Alvin Subakti
Study Program	:	Mathematics
Title	:	Performance Analysis of BERT as a Text Representation Method for Text Clustering
Counsellor	:	1. Dr.rer.nat. Hendri Murfi, S.Si., M.Kom. 2. Dra. Nora Hariadi, M.Si.

Text clustering is a task of grouping a set of texts in a way such that text in the same group will be more similar toward each other than to those from different group. The process of grouping text manually requires significant amount of time and labor. Therefore, automation utilizing machine learning is necessary. Text representation needs to be extracted to become the input for machine learning models. The common method used to represent textual data is TFIDF. However, TFIDF cannot consider the position and context of a word in a sentence. BERT model has the capability to produce text representation that incorporate position and context of a word in a sentence. This research analyzed the performance of BERT model as a text representation method by comparing it with TFIDF. Moreover, various feature extraction and normalization methods are also applied in text representation from BERT model. Feature extraction methods used are max and mean pooling. On the other hand, feature normalization methods used are identity, layer, standard, and min-max normalization. Text representation obtained become an input for 4 clustering algorithms, k-means clustering, eigenspace-based fuzzy c-means, deep embedded clustering, and improved deep embedded clustering. Performance of text representations in text clustering are evaluated utilizing clustering accuracy, normalized mutual information, and adjusted rand index. Simulation results showed that text representation obtained from BERT model outperforms representation from TFIDF in 28 out of 36 metrics. Furthermore, different feature extraction and normalization produced varied performances. The usage of these feature extraction and normalization must be altered depending on the text clustering algorithm used.

Key words:

BERT, Text Representation, Text Clustering

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERNYATAAN ORISINALITAS	iii
HALAMAN PENGESAHAN	v
KATA PENGANTAR	vii
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH	xi
ABSTRAK.....	xiii
ABSTRACT	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR.....	xix
DAFTAR TABEL	xxi
DAFTAR LAMPIRAN	xxiii
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah	3
1.3. Tujuan Penelitian	3
1.4. Metodologi Penelitian.....	3
1.5. Batasan Masalah	4
BAB 2 LANDASAN TEORI	5
2.1. <i>Term Frequency Inverse Document Frequency</i>	5
2.2. <i>Truncated Singular Value Decomposition</i>	6
2.3. <i>Autoencoder</i>	7
2.3.1. <i>Artificial Neural Network</i>	8
2.3.1.1. Model <i>Artificial Neural Network</i>	10
2.3.1.2. Fungsi Aktivasi	12
2.3.1.3. Fungsi <i>Loss</i>	13
2.3.1.4. <i>Error Backpropagation</i>	13
2.3.1.5. Optimisasi.....	16
2.3.2. Arsitektur <i>Autoencoder</i>	18
2.4. <i>Clustering</i>	19
2.4.1. <i>K-Means Clustering</i>	20
2.4.2. <i>Eigenspace Based Fuzzy C Means</i>	22
2.4.3. <i>Deep Embedded Clustering</i>	27
2.4.3.1. Inisialisasi Parameter DEC.....	29
2.4.3.2. Optimisasi Parameter DEC	31
2.4.4. <i>Improved Deep Embedded Clustering</i>	34
2.4.4.1. Inisialisasi dan <i>Clustering Loss</i>	36
2.4.4.2. Pelestarian Struktur Lokal	36
BAB 3 BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER	40
3.1. <i>Transformer</i>	40
3.1.1. Arsitektur Model <i>Transformer</i>	40
3.1.2. <i>Attention</i>	41
3.1.2.1. <i>Multi-head attention</i>	45
3.1.2.2. Penerapan <i>Attention</i> dalam Model <i>Transfomer</i>	47

3.1.3. <i>Position-wise Feed Forward Network</i>	47
3.1.4. Koneksi Residu dan Normalisasi	48
3.1.5. <i>Positional Encoding</i>	50
3.2. <i>Bidirectional Encoder Representation from Transformer</i>	50
3.2.2. Representasi Input.....	52
3.2.3. <i>Pre-training BERT</i>	55
3.2.3.1. <i>Masked Language Modelling</i>	56
3.2.3.2. <i>Next Sentence Prediction</i>	56
3.2.3.3. <i>Data Pre-training</i>	57
3.3. <i>Bidirectional Encoder Representation from Transformer</i> Sebagai Metode Representasi Teks	58
3.3.1. Pendekatan Berbasis Fitur dengan BERT	58
3.3.2. Ekstraksi Fitur	59
3.3.3. Normalisasi Fitur.....	61
BAB 4 HASIL SIMULASI DAN PEMBAHASAN	64
4.1. Sepsifikasi Mesin dan Perangkat Lunak	64
4.2. Tahapan Umum Simulasi.....	65
4.3. Data.....	66
4.3.1. Pengambilan Data	66
4.3.2. Pra-Pengolahan Data.....	69
4.4. Representasi Data Teks.....	70
4.4.1. Representasi Data Teks dengan Metode TFIDF	70
4.4.2. Representasi Data Teks dengan Metode BERT	73
4.5. Simulasi <i>Text Clustering</i>	76
4.5.1. Model <i>K-Means Clustering</i>	76
4.5.2. Model <i>Eigenspace-based Fuzzy C-Means</i>	76
4.5.3. Model <i>Deep Embedded Clustering</i>	77
4.5.4. Model <i>Improved Deep Embedded Clustering</i>	79
4.6. Metrik Evaluasi.....	80
4.6.1. <i>Clustering Accuracy</i>	81
4.6.2. <i>Normalized Mutual Information</i>	82
4.6.3. <i>Adjusted Rand Index</i>	83
4.7. Hasil dan Analisis Simulasi	84
4.7.1. Performa pada Model <i>K-Means Clustering</i>	85
4.7.2. Performa pada Model <i>Eigenbased-space Fuzzy C-Means</i>	88
4.7.3. Performa pada Model <i>Deep Embedded Clustering</i>	90
4.7.4. Performa pada Model <i>Improved Deep Embedded Clustering</i>	92
4.7.5. Analisis Perbandingan Representasi Data Teks Antara Metode TFIDF Dengan Metode BERT	95
4.7.6. Analisis Perbandingan Metode Ekstraksi dan Normalisasi Fitur yang Berbeda pada Representasi Data Teks Dengan Model BERT	97
BAB 5 PENUTUP	99
5.1. Kesimpulan	99
5.2. Saran	99
DAFTAR PUSTAKA	101

DAFTAR GAMBAR

Gambar 2.1. Ilustrasi dekomposisi matriks dengan SVD	6
Gambar 2.2. Ilustrasi dekomposisi matriks dengan TSVD.....	7
Gambar 2.3. (a) Struktur neuron biologis (b) Mekanisme perpindahan sinyal di antara dua neuron	8
Gambar 2.4. (a) Jaringan Neuron Secara Biologis (b) Jaringan NN.....	9
Gambar 2.5. (a) <i>Single layer perceptron</i> (b) <i>Multilayer perceptron</i>	9
Gambar 2.6. Struktur MLP	10
Gambar 2.7. Ilustrasi perhitungan nilai δj	15
Gambar 2.8. (a) Arsitektur <i>deep autoencoder</i> (b) Arsitektur DEC.....	28
Gambar 2.9. Struktur <i>denoising autoencoder</i>	30
Gambar 2.10. Proses pembelajaran <i>autoencoder</i> dengan <i>greedy layer-wise pretraining</i> (a) <i>Autoencoder</i> yang dibangun (b) Pembangunan lapisan 1 <i>encoder</i> (c) Pembangunan lapisan <i>code</i> (d) tahap <i>finetuning</i> dari <i>autoencoder</i>	30
Gambar 2.11. Ilustrasi struktur model <i>improved deep embedded clustering</i>	35
Gambar 3.1. Ilustrasi arsitektur <i>transformer</i>	41
Gambar 3.2. Ilustrasi penentuan nilai vektor <i>query</i> , <i>key</i> , dan <i>value</i>	42
Gambar 3.3. Ilustrasi fungsi <i>attention</i>	43
Gambar 3.4. Ilustrasi <i>scaled dot product</i>	44
Gambar 3.5. Ilustrasi <i>multi-head attention</i>	46
Gambar 3.6. Ilustrasi koneksi residual dan normalisasi pada <i>transformer</i>	49
Gambar 3.7. Ilustrasi penjumlahan antara <i>input embeddings</i> dan <i>positional encodings</i>	50
Gambar 3.8. Ilustrasi model BERT.....	51
Gambar 3.9. Pemetaan <i>word embedding</i>	53
Gambar 3.10. Representasi input model BERT	53
Gambar 3.11. Pemetaan <i>segment embedding</i>	54
Gambar 3.12. Pemetaan <i>position embedding</i>	55
Gambar 3.13. Alur pembelajaran <i>masked language modelling</i>	56
Gambar 3.14. Alur pembelajaran <i>next sentence prediction</i>	57
Gambar 3.15. Ilustrasi metode representasi teks dengan model BERT	58
Gambar 4.1. Tahapan umum simulasi	65
Gambar 4.2. <i>Loss pre-training autoencoder</i> pada model DEC	78
Gambar 4.3. Tabel kontingensi antara dua partisi	81
Sumber: (Hubert & Arabie, 1985), telah diolah kembali	81
Gambar 4.4. Visualisasi t-SNE <i>ground-truth label AG news</i> dengan representasi data teks (a) TFIDF (b) BERT	96
Gambar 4.5. Visualisasi t-SNE representasi data teks <i>R2</i> dengan metode (a) TFIDF (b) model BERT	97

Gambar 4.6. Diagram kotak 20 representasi data *AG news* dengan metode BERT dan *mean pooling* (a) sebelum *min-max normalization* (b) sesudah *min-max normalization*.

..... 98



DAFTAR TABEL

Tabel 2.1. Algoritma <i>k-means clustering</i>	22
Tabel 2.2. Algoritma <i>eigenspace-based fuzzy c-means</i>	27
Tabel 2.3. Algoritma <i>deep embedded clustering</i>	33
Tabel 2.4. Algoritma <i>improved deep embedded clustering</i>	37
Tabel 4.1. Spesifikasi mesin dan perangkat lunak.....	64
Tabel 4.2. Deskripsi singkat data.....	66
Tabel 4.3. Distribusi kelas pada data <i>R2</i>	67
Tabel 4.4. Contoh data teks <i>AG news</i>	67
Tabel 4.5. Contoh data teks <i>R2</i>	68
Tabel 4.6. Contoh data teks <i>Yahoo! Answers</i>	68
Tabel 4.7. Contoh pembersihan data.....	69
Tabel 4.8. Contoh tokenisasi data pada metode TFIDF.....	70
Tabel 4.9. Dokumen untuk ilustrasi TF-IDF	70
Tabel 4.10. Ilustrasi perhitungan TFIDF	71
Tabel 4.11. Hasil akhir metode TFIDF	72
Tabel 4.12. Contoh tokenisasi data pada metode BERT	73
Tabel 4.13. Contoh <i>padding</i> dan <i>truncating</i> data pada metode BERT	74
Tabel 4.14. Contoh <i>encoding</i> data pada metode BERT	75
Tabel 4.15. Hyperparameter <i>k-means clustering</i>	76
Tabel 4.16. Hyperparameter <i>eigen-space based fuzzy c-means</i>	77
Tabel 4.17. Hyperparameter <i>deep embedded clustering</i>	77
Tabel 4.18. Hyperparameter <i>improved deep embedded clustering</i>	79
Tabel 4.19. Deskripsi penyederhanaan	85
Tabel 4.20. Hasil evaluasi <i>k-means clustering</i> pada data <i>AG news</i>	86
Tabel 4.21. Hasil evaluasi <i>k-means clustering</i> pada data <i>Yahoo! Answers</i>	86
Tabel 4.22. Hasil evaluasi <i>k-means clustering</i> pada data <i>R2</i>	87
Tabel 4.23. Hasil evaluasi <i>eigenspace-based fuzzy c-means</i> pada data <i>AG news</i>	88
Tabel 4.24. Hasil evaluasi <i>eigenspace-based fuzzy c-means</i> pada data <i>Yahoo! Answers</i>	89
Tabel 4.25. Hasil evaluasi <i>eigenspace-based fuzzy c-means</i> pada data <i>R2</i>	90
Tabel 4.26. Hasil evaluasi <i>deep embedded clustering</i> pada data <i>AG news</i>	90
Tabel 4.27. Hasil evaluasi <i>deep embedded clustering</i> pada data <i>Yahoo! Answers</i>	91
Tabel 4.28. Hasil evaluasi <i>deep embedded clustering</i> pada data <i>R2</i>	92
Tabel 4.29. Hasil evaluasi <i>improved deep embedded clustering</i> pada data <i>AG news</i> ...	93
Tabel 4.30. Hasil evaluasi <i>improved deep embedded clustering</i> pada data <i>Yahoo! Answers</i>	94
Tabel 4.31. Hasil evaluasi <i>improved deep embedded clustering</i> pada data <i>R2</i>	94

DAFTAR LAMPIRAN

Lampiran 1. Fungsi Pengambilan Representasi Teks dengan Model BERT	107
Lampiran 2. Program <i>Text Clustering</i>	109



BAB 1

PENDAHULUAN

1.1. Latar Belakang

Teknologi informasi memiliki peranan penting dalam aktivitas manusia di kehidupan sehari-hari. Seiring dengan perkembangan zaman, teknologi informasi juga mengalami pekembangan yang sangat cepat. Perkembangan teknologi informasi ini didukung oleh meningkatnya ketersedian internet. Contohnya di negara Indonesia, jumlah pengguna internet di Indonesia mencapai 175,4 juta orang pada tahun 2020. Jumlah pengguna ini setara dengan 64% populasi Indonesia dan mengalami peningkatan sebesar 17% apabila dibandingkan dengan tahun 2019 (Ramadhan, 2020). Salah satu dampak dari peningkatan ketersediaan internet adalah adanya peningkatan jumlah berita yang tersedia secara daring (Roser, Ritchie, & Ortiz-Ospina, 2021). Kondisi *lockdown* ketika pandemi Covid-19 juga mengakibatkan transformasi *digital* yang lebih cepat, ditunjukkan dengan menurunnya konsumsi media massa cetak seperti koran dan meningkatnya konsumsi media *digital* seperti televisi atau berita *online* (Dams, 2020)

Pengelompokan terhadap berita yang tersebar di internet dapat memudahkan pengguna ketika ingin mencari berita tertentu. Salah satu contoh pengelompokan yang dapat dilakukan adalah pengelompokan berita berdasarkan topik yang sedang dibahas. Pengelompokan berita dapat dilakukan dengan menganalisis teks dalam berita dan menentukan topik yang terkandung dalam teks tersebut. Namun, banyaknya berita yang tersedia di internet mengakibatkan proses pengelompokan data teks secara manual sulit dilakukan. Hal ini dikarenakan pengelompokan data teks secara manual membutuhkan sumber daya manusia yang banyak dan membutuhkan waktu yang lama. Oleh karena itu, dibutuhkan metode dan algoritma yang dapat digunakan untuk mengolah dan melakukan analisis terhadap data teks secara otomatis, salah satunya adalah dengan menggunakan *machine learning*.

Machine learning terbagi menjadi dua jenis berdasarkan metode pembelajarannya, yaitu *supervised learning* dan *unsupervised learning*. *Supervised learning* adalah pembelajaran menggunakan data yang memiliki *label*, sementara *unsupervised learning* adalah pembelajaran menggunakan data yang tidak memiliki *label* (Bishop, 2006). Data teks yang tersedia di internet umumnya tidak memiliki *label*. Selain itu, aktivitas

pelabelan data teks juga membutuhkan sumber daya manusia yang banyak dan mahal (Cloud Google, 2021). Oleh karena dua alasan tersebut, metode *unsupervised learning* cocok digunakan untuk menentukan kelompok pada data teks.

Text Clustering adalah proses pengelompokan teks yang serupa dari sekumpulan teks. *Text clustering* memiliki beberapa tingkatan granularitas, yaitu tingkat dokumen, paragraf, kalimat, atau frasa. Beberapa aplikasi dari metode *text clustering* adalah pada pengorganisasian buku, peringkasan korpus, dan pengklasifikasian dokumen (Aggarwal & Zhai, 2012). Hingga sekarang ini, berbagai algoritma *unsupervised learning* telah diimplementasikan untuk melakukan *text clustering*. Beberapa contoh algoritma yang sudah pernah diterapkan adalah *k-means clustering* (Xiong, Huang, Lv, & Li, 2016), *eigenspace-based fuzzy c-means* (EFCM) (Murfi, 2018), *deep embedded clustering* (DEC) (Xie, Girshick, & Farhadi, 2016), dan *improved deep embedded clustering* (IDEC) (Guo, Gao, Liu, & Yin, 2017).

Salah satu tahapan awal pada *text clustering* adalah merepresentasikan teks dalam bentuk suatu vektor numerik (Guan *et al.*, 2020). Hal ini dikarenakan model tidak dapat langsung memproses data dalam bentuk teks sehingga perlu dilakukan transformasi ke bentuk numerik terlebih dahulu. Selain itu, proses merepresentasikan teks juga dapat mempelajari pola dari input. *Representation learning* adalah metode yang secara otomatis mengubah data mentah berbentuk teks menjadi representasinya. Metode *representation learning* yang umumnya digunakan adalah metode *bag of words* seperti *Term Frequency-Inverse Document Frequency* (TFIDF) dan metode *sequence of words* seperti word2vec dan *Bidirectional Encoder Representations from Transformers* (BERT).

Model BERT adalah *pretrained language model* yang diteliti Devlin *et al.* (2018). Model BERT memanfaatkan arsitektur model *transformer* dan dapat mencapai performa *State Of The Art* (SOTA) untuk beberapa permasalahan *Natural Language Processing* (NLP). Model BERT dapat digunakan dengan 2 pendekatan, yaitu pendekatan berbasis fitur dan pendekatan berbasis *fine tuning*. Pada pendekatan berbasis fitur, model BERT merepresentasikan data teks menjadi fitur tetap menggunakan model *pre-trained*. Model BERT dapat menghasilkan representasi yang bergantung pada posisi dan konteks penggunaan suatu kata dalam kalimat. Hingga saat ini, beberapa penelitian sudah menerapkan pendekatan berbasis fitur untuk memperoleh representasi teks dari model BERT. Beberapa penerapannya adalah pendekripsi *toxic speech* (d'Sa, Illina, & Fohr,

2020) dan klasifikasi teks (Ye *et al.*, 2020; Yu, Wang, & Jiang, 2021). Sebagian besar penelitian yang dilakukan adalah mengembangkan representasi data teks dari model BERT dalam menyelesaikan masalah *supervised learning*. Namun, masih jarang ada penelitian yang berfokus pada pengembangan metode representasi tersebut dalam menyelesaikan masalah *unsupervised learning*. Pada penelitian ini, model BERT diimplementasikan sebagai metode representasi teks dalam menyelesaikan masalah *unsupervised learning* yaitu *text clustering*.

1.2. Rumusan Masalah

Berikut ini merupakan masalah yang akan dibahas dalam skripsi ini:

1. Bagaimana kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*?
2. Bagaimana pengaruh metode ekstraksi dan normalisasi yang berbeda terhadap kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut :

1. Menganalisis kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*.
2. Membandingkan pengaruh metode ekstraksi dan normalisasi yang berbeda terhadap kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*.

1.4. Metodologi Penelitian

Metode yang digunakan pada skripsi ini adalah sebagai berikut

1. Studi literatur

Penelitian ini diawali dengan melakukan studi literatur terkait model dan konsep yang digunakan dalam penelitian ini. Beberapa topik yang dipelajari adalah model BERT, *representation learning*, *k-means clustering*, EFCM, DEC, dan IDEC. Studi dilakukan dengan mempelajari buku dan penelitian yang membahas topik yang disebutkan sebelumnya. Hasil studi literatur digunakan sebagai landasan teori pada penelitian ini.

2. Pengumpulan data

Data yang digunakan pada penelitian ini adalah 3 dari 5 data yang digunakan pada penelitian Guan *et al.* (2020). Data yang digunakan bersumber dari 3 media berita *online* yang berbeda, yaitu *AG News*, *Yahoo! Answers*, dan *Reuters*.

3. Implementasi dan simulasi program

Implementasi pada penelitian ini diterapkan menggunakan Bahasa pemrograman Python. Tahapan simulasi terdiri atas persiapan data, dilanjutkan dengan pengambilan representasi data teks, dilanjutkan dengan implementasi representasi data teks dalam *text clustering*. Hasil simulasi dievaluasi menggunakan 3 metrik evaluasi.

4. Analisis hasil simulasi

Penelitian ini difokuskan pada 2 buah analisis. Analisis pertama adalah analisis perbandingan kinerja kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*. Sementara analisis kedua adalah analisis pengaruh metode ekstraksi dan normalisasi yang berbeda terhadap kinerja model BERT sebagai representasi teks dalam melakukan *text clustering*.

1.5. Batasan Masalah

Adapun batasan masalah yang digunakan pada skripsi ini adalah sebagai berikut:

1. Data yang digunakan adalah data *AG news*, *R2*, dan *Yahoo! Answers* yang digunakan penelitian Guan *et al.* (2020).
2. Metode *clustering* yang akan digunakan untuk dievaluasi performanya adalah *k-means clustering*, EFCM, DEC, dan IDEC.
3. Metrik yang digunakan untuk evaluasi adalah *clustering accuracy*, *normalized mutual information*, dan *adjusted rand index*.

BAB 2

LANDASAN TEORI

2.1. *Term Frequency Inverse Document Frequency*

Term Frequency Inverse Document Frequency (TFIDF) adalah salah satu metode representasi kata yang dapat memberikan bobot konstan pada setiap kata. Penjelasan mengenai TFIDF merujuk pada penelitian Ramos (2003) kecuali disebutkan yang lain. Secara umum, representasi dari metode TFIDF mengimplikasikan tingkat relevansi suatu kata terhadap dokumen tertentu. TFIDF memperhitungkan dua hal, yaitu frekuensi kata (*term frequency*) dan invers dari frekuensi kemunculan kata pada dokumen (*inverse document frequency*).

Penentuan nilai representasi numerik dari suatu kata t pada suatu dokumen d dengan metode TFIDF dapat ditentukan dengan persamaan (2.1):

$$w_{t,d} = tf_{t,d} \times \log\left(\frac{N}{df_t}\right), \quad (2.1)$$

dengan $tf_{t,d}$ merepresentasikan frekuensi kata t pada dokumen d , N merepresentasikan banyaknya dokumen, dan df_t merepresentasikan frekuensi dokumen yang mengandung kata t .

Misalkan df_t mendekati nilai frekuensi dokumen N . Jika $0 < \log(N/df_t) < c$ dengan c suatu nilai konstanta yang sangat kecil, maka $w_{t,d}$ akan cenderung lebih kecil dari $tf_{t,d}$ namun tetap memiliki nilai yang positif. Hal ini mengimplikasikan bahwa kemunculan kata t relatif banyak namun masih dianggap penting dalam keseluruhan dokumen N . Kata seperti preposisi atau kata ganti juga memiliki nilai representasi TFIDF yang sangat rendah. Hal ini mengimplikasikan kata-kata tersebut tidak memiliki kepentingan dalam dokumen dan dapat diabaikan. Sementara itu, kata dengan kemunculan yang banyak di suatu dokumen tapi sedikit pada dokumen lainnya akan berakibat pada nilai $f_{t,d}$ dan $\log\left(\frac{N}{df_t}\right)$ yang relatif besar sehingga bobot yang diperoleh juga menjadi besar. Hal ini mengimplikasikan kata tersebut dianggap penting dan merepresentasikan dokumen dimana kata tersebut sering muncul.

Hasil representasi data teks dari *representation learning* seperti TFIDF dapat digunakan sebagai input dari berbagai algoritma *machine learning*, salah satunya adalah pada algoritma-algoritma *text clustering*. Selanjutnya akan dijelaskan terlebih dahulu

beberapa konsep yang diimplementasikan pada algoritma *text clustering* yang digunakan pada penelitian ini. Beberapa konsep tersebut adalah *truncated singular decomposition* pada Subbab 2.2 dan *autoencoder* pada Subbab 2.3.

2.2. Truncated Singular Value Decomposition

Pertama akan dijelaskan terlebih dahulu mengenai *singular value decomposition* (SVD). SVD adalah salah satu metode untuk mendekomposisikan matriks dengan matriks yang bukan matriks persegi. Misalkan terdapat matriks A berukuran $m \times n$, maka dekomposisi matriks A dengan menggunakan SVD dapat direpresentasikan pada persamaan (2.2):

$$A = U\Sigma V^T, \quad (2.2)$$

dengan U adalah matriks ortogonal berukuran $m \times m$, Σ adalah matriks berukuran $m \times n$, dan V adalah matriks ortogonal berukuran $n \times n$.

Dekomposisi matriks A dengan menggunakan SVD dapat diilustrasikan sesuai dengan Gambar 2.1:

$$A = U\Sigma V^T = [\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_m] \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_p \\ 0_{(m-p) \times p} & & & 0_{(m-p) \times (n-p)} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}$$

Gambar 2.1. Ilustrasi dekomposisi matriks dengan SVD

Sumber: (Anton & Rorres, 2013), telah diolah kembali

Misalkan λ adalah suatu nilai eigen dan σ adalah akar dari nilai eigen, $\sqrt{\lambda}$. Untuk suatu matriks A berukuran $m \times n$, matriks $A^T A$ memiliki paling banyak n nilai eigen. Entri tak nol pada diagonal dari matriks Σ merupakan *singular value*, yaitu σ terurut dari nilai eigen matriks $A^T A$. Sehingga $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq \sigma_p > 0$, dengan p adalah banyaknya nilai eigen tak nol dari $A^T A$. Vektor \mathbf{v}_i berdimensi n pada matriks V merupakan vektor eigen dari matriks $A^T A$ yang juga sudah diurutkan sesuai dengan urutan *singular value* pada matriks Σ . Himpunan vektor $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p\}$ adalah basis ortonormal dari ruang kolom A . Di sisi lain, himpunan vektor $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p, \mathbf{u}_{p+1}, \dots, \mathbf{u}_m\}$ adalah perluasan dari basis ortonormal $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p\}$ untuk R^m . Sehingga himpunan $\{\mathbf{u}_{p+1}, \dots, \mathbf{u}_m\}$ adalah himpunan vektor satuan yang saling bebas linier baik antara elemen dalam himpunan tersebut maupun dengan setiap elemen pada himpunan $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p\}$.

Pada metode SVD, vektor \mathbf{u}_i akan berperan sebagai basis baru yang dapat merepresentasikan A . Kemudian σ_i menunjukkan besarnya variansi A yang dijelaskan oleh vektor \mathbf{u}_i . Terakhir vektor \mathbf{v}_i merupakan kombinasi linier dari $\mathbf{u}_i \sigma_i$. Karena entri diagonal matriks Σ adalah *singular value* yang sudah diurutkan dari terbesar hingga terkecil, maka untuk suatu nilai k dengan $k \leq p$, dekomposisi yang ditunjukkan pada Gambar 2.2 merupakan aproksimasi terbaik dari matriks A yang berdimensi k , dengan entri dari setiap matriks pada Gambar 2.2 berasal dari entri pada Gambar 2.1.

$$A \approx [\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k \end{bmatrix} [\mathbf{v}_1^T \mathbf{v}_2^T \dots \mathbf{v}_k^T]$$

Gambar 2.2. Ilustrasi dekomposisi matriks dengan TSVD

Sumber: (Anton & Rorres, 2013), telah diolah kembali

Aproksimasi terbaik ini disebabkan oleh pengurutan *singular value* yang merupakan representasi seberapa besar variansi dari A terjelaskan. Pengambilan sebanyak k *singular value* menghilangkan $(p - k)$ *singular value* yang paling tidak signifikan. Pengambilan ini disebut dengan metode *Truncated Singular Value Decomposition* (TSVD) (Burden *et al.*, 2011) yang merupakan salah satu metode yang berguna untuk mereduksi dimensi dari dekomposisi suatu matriks A . Aproksimasi matriks A dengan menggunakan metode TSVD dapat direpresentasikan pada persamaan (2.3):

$$A \approx \tilde{U} \tilde{\Sigma} \tilde{V}^T, \quad (2.3)$$

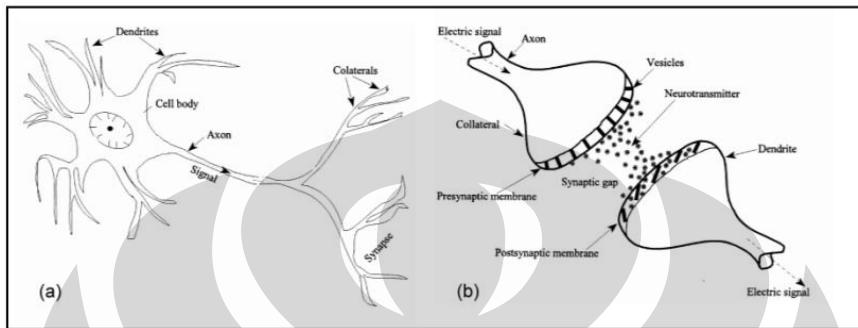
Dengan \tilde{U} adalah matriks berukuran $m \times k$, $\tilde{\Sigma}$ adalah matriks berukuran $k \times k$, dan \tilde{V}^T adalah matriks berukuran $k \times n$.

2.3. Autoencoder

Pada penelitian ini, digunakan metode reduksi dimensi dengan *autoencoder* (Hinton & Salakhutdinov, 2006). Bagian ini tersusun atas penjelasan mengenai *artificial neural network* yang merupakan komponen pembentuk *autoencoder*, kemudian dilanjutkan dengan penjelasan arsitektur model *autoencoder*.

2.3.1. Artificial Neural Network

Artificial neural network (NN) merupakan algoritma yang bertujuan untuk mempelajari pola pada data dengan pendekatan matematis yang mengikuti proses penerimaan informasi dan pemrosesan pada otak manusia (Basheer & Hajmeer, 2000). Oleh karena itu, perlu diketahui terlebih dahulu cara kerja otak manusia dalam mempelajari informasi tertentu.

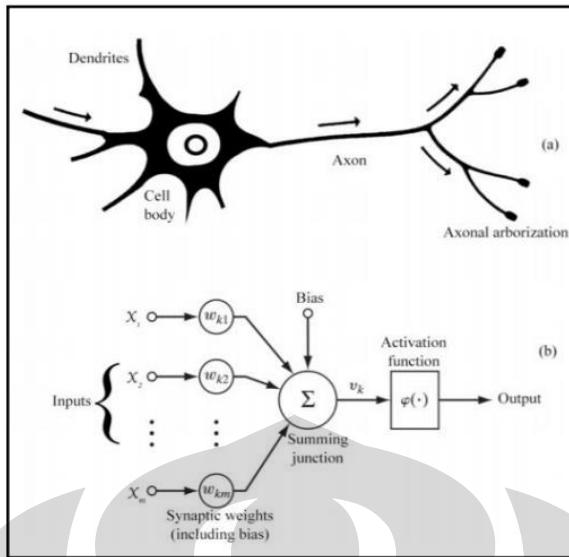


Gambar 2.3. (a) Struktur neuron biologis (b) Mekanisme perpindahan sinyal di antara dua neuron

Sumber: (Basheer & Hajmeer, 2000), telah diolah kembali

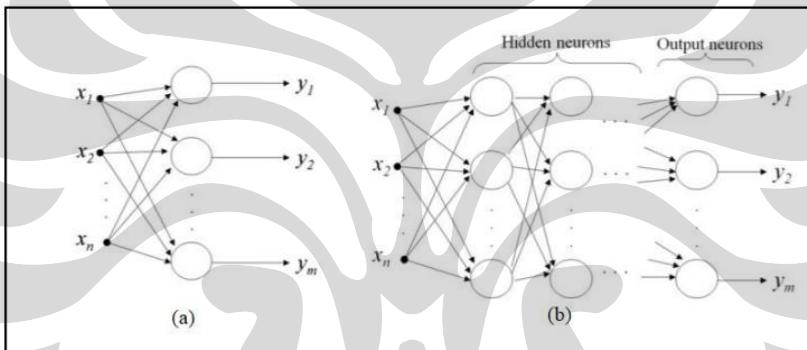
Gambar 2.3a. menerangkan struktur neuron otak manusia. Neuron terdiri dari tiga komponen, yaitu dendrit, badan sel (*cell body*), dan *axon*. Informasi yang diterima dari luar otak memiliki bentuk berupa sinyal listrik pada saraf. Dendrit menerima sinyal listrik dari neuron untuk kemudian dikirim ke badan sel, yang berlanjut ke *axon* dan akhirnya dikirimkan ke neuron terdekat melalui sinapsis atau celah mikroskopik (Basheer & Hajmeer, 2000). Ketika sinyal listrik menyebar melalui celah sinapsis, *neurotransmitter* dilepaskan dari vesikel. Hal ini ditunjukkan pada Gambar 2.3b (Basheer & Hajmeer, 2000). *Neurotransmitter* akhirnya akan mendekati dendrit pada neuron dan menghasilkan sinyal listrik baru yang kemudian kembali berpindah ke neuron lain. Metode ini menjadi inspirasi bagi mekanisme NN.

Gambar 2.4. menunjukkan similaritas antara mekanisme NN dengan jaringan neuron biologis. Bobot pada setiap sambungan NN analog dengan synapsis, input pada NN analog dengan sinyal listrik yang masuk ke dendrit, fungsi aktivasi pada NN analog dengan badan sel, dan *output* pada NN analog dengan sinyal listrik yang diteruskan *axon* (Akgun & Demir, 2018). NN adalah salah satu metode untuk menyelesaikan masalah klasifikasi, sehingga sering dikategorisasikan dalam *supervised learning*.



Gambar 2.4. (a) Jaringan Neuron Secara Biologis (b) Jaringan NN

Sumber: (Akgun & Demir, 2018), telah diolah kembali



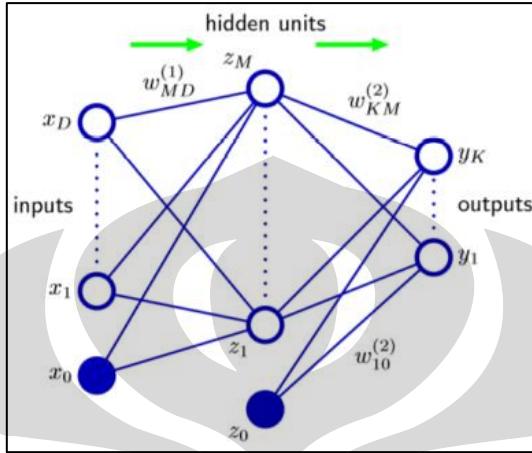
Gambar 2.5. (a) Single layer perceptron (b) Multilayer perceptron

Sumber: (Vanneschi & Castelli, 2019), telah diolah kembali

NN dapat dibagi menjadi 2 jenis berdasarkan hubungan antara neuron, yakni *feedforward neural network* (FNN) dan *recurrent neural network* (RNN). Pada FNN, *output* tidak dimasukkan kembali ke dalam model karena tidak ada hubungan rekursif antara neuron-neuron. Sebaliknya, neuron pada RNN memiliki hubungan rekursif (Goodfellow, Bengio, & Courville, 2016). *Single layer perceptron* adalah model yang mengandung sejumlah neuron pada lapisan input dan diproses langsung menuju lapisan *output* (Bishop, 2006). Sementara itu, FNN juga dapat dikenal sebagai *multilayer Perceptron* (MLP) karena setiap langkahnya menyerupai model *single layer perceptron*. Gambar 2.5. menunjukkan perbedaan antara MLP dengan *single layer perceptron*, pada penelitian ini digunakan NN berjenis MLP.

2.3.1.1. Model Artificial Neural Network

Model NN tersusun atas tiga bagian, yaitu lapisan input, lapisan tersembunyi, dan lapisan *output*. Bagian lapisan tersembunyi bisa memiliki lebih dari satu lapisan. Model disebut sebagai *deep neural network* (DNN) apabila memiliki lebih dari satu lapisan tersembunyi. Visualisasi struktur model NN dapat dilihat pada Gambar 2.6.



Gambar 2.6. Struktur MLP

Sumber: (Bishop, 2006)

Model NN memiliki kemampuan untuk memprediksi target bila suatu vektor data berdimensi D , $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$, dimasukkan ke dalam lapisan input kemudian diproses pada lapisan tersembunyi. Pada model NN, setiap neuron yang terhubung memiliki parameter bobot. Misalkan bobot dari sambungan neuron ke- i pada lapisan pertama dengan neuron ke- j pada lapisan kedua dinotasikan dengan $w_{ji}^{(1)}$, dan setiap neuron yang tidak berada pada lapisan input memiliki parameter bias yang dinotasikan dengan $w_{j0}^{(1)}$.

Input dari neuron $z_j, j = 1, \dots, M$ pada Gambar 2.6. dapat diekspresikan pada persamaan (2.4):

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad (2.4)$$

dengan M menyatakan banyak neuron di lapisan tersembunyi.

Variabel a_j pada persamaan (2.4) disebut dengan nilai aktivasi. Nilai aktivasi akan ditransformasi oleh sebuah fungsi nonlinier terturunkan $h(\cdot)$ yang disebut dengan fungsi aktivasi. Nilai aktivasi yang telah ditransformasi dengan menggunakan $h(\cdot)$ akan

berperan sebagai *output* dari lapisan tersembunyi. *Output* dari neuron $z_j, j = 1, \dots, M$, dapat diekspresikan dalam persamaan (2.5):

$$z_j = h(a_j). \quad (2.5)$$

Output dari suatu lapisan tersembunyi menjadi input untuk lapisan tersembunyi berikutnya apabila masih ada lapisan tersembunyi atau menjadi input untuk lapisan *output* apabila sudah tidak ada lapisan tersembunyi lain. Misalkan neuron pada lapisan *output* direpresentasikan sebagai $y_k, k = 1, \dots, K$, dengan K adalah banyak kemungkinan *output*. Misalkan bobot dari sambungan neuron ke- j pada lapisan ke- m dengan neuron ke- k pada lapisan ke- $(m - 1)$ dinotasikan dengan $w_{kj}^{(m)}$, dan parameter bias pada lapisan ke- m yang dinotasikan dengan $w_{k0}^{(m)}$. Input berupa nilai aktivasi yang diberikan adalah fungsi dari z_j seperti yang ditunjukkan pada persamaan (2.6):

$$a_k = \sum_{j=1}^M w_{kj}^{(m)} z_j + w_{k0}^{(m)}. \quad (2.6)$$

Nilai a_k akan kembali ditransformasi oleh fungsi aktivasi $l(\cdot)$ dan menjadi *output* pada lapisan *output*. Hasil yang diperoleh disebut sebagai *output unit activation* dinotasikan dengan y_k . *Output* dari model NN diberikan oleh persamaan (2.7):

$$y_k = l(a_k). \quad (2.7)$$

Apabila persamaan (2.4) sampai (2.7) digabungkan maka untuk suatu MLP dengan banyak keseluruhan lapisan sebelum lapisan *output* adalah N , akan diperoleh *output* $y_k(\mathbf{x}, \mathbf{w})$, dengan parameter bobot dan bias tergabung dalam vektor \mathbf{w} , berupa persamaan (2.8):

$$y_k(\mathbf{x}, \mathbf{w}) = l\left(\sum_{j=1}^M w_{kj}^{(N)} h\left(\sum_{i=1}^D w_{ji}^{(N-1)} z_i + w_{j0}^{(N-1)}\right) + w_{k0}^{(N)}\right), \quad (2.8)$$

dengan M menyatakan banyak neuron pada lapisan ke- N , dan D menyatakan banyak neuron pada lapisan ke- $(N - 1)$. Nilai tetap z_i merupakan entri ke- i dari vektor \mathbf{x} apabila lapisan ke- $(N - 1)$ merupakan lapisan input dan merupakan nilai aktivasi ke- i apabila lapisan ke- $(N - 1)$ merupakan lapisan tersembunyi. Sehingga dapat disimpulkan bahwa NN adalah fungsi nonlinier yang memetakan himpunan input $\{x_i\}$ ke *output* $\{y_k\}$ yang

dipengaruhi oleh bobot w . Proses pemetaan ini juga disebut dengan *forward propagation* (Bishop, 2006).

Seperti yang dijelaskan sebelumnya, pada setiap lapisan pada model NN, dibutuhkan fungsi aktivasi untuk memperoleh *output* pada setiap lapisan. Oleh karena itu, pada Subbab 2.3.1.2, diberikan penjelasan mengenai berbagai fungsi aktivasi yang umumnya digunakan.

2.3.1.2. Fungsi Aktivasi

Fungsi aktivasi mengaktifkan neuron sehingga input berubah menjadi *output*. persamaan (2.9) sampai (2.13) adalah beberapa contoh fungsi yang umumnya digunakan sebagai fungsi aktivasi (Bishop, 2006; Goodfellow, Bengio, & Courville, 2016).

a. Fungsi Identitas

$$f(a) = a; \quad f(a) \in \mathbb{R}, \quad (2.9)$$

b. Fungsi Logistic Sigmoid

$$f(a) = \frac{1}{1 + \exp(-a)}; \quad f(a) \in (0,1), \quad (2.10)$$

c. Fungsi Hyperbolic Tangent

$$f(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}; \quad f(a) \in (-1,1), \quad (2.11)$$

d. Fungsi Softmax

$$f(a_i) = \frac{\exp(a_i)}{\sum_{j=1}^k \exp(a_j)}; \quad i = 1, 2, \dots, k; \quad f(a_i) \in (0,1], \quad (2.12)$$

e. Fungsi Rectified Linear Unit (ReLU)

$$f(a) = \max\{0, a\}; \quad f(a) \in [0, \infty). \quad (2.13)$$

Pada bagian ini sudah dijelaskan mengenai mekanisme kerja NN dalam memprediksi *target* suatu data. Pada bagian-bagian selanjutnya akan dijelaskan metode pembelajaran model NN sehingga menghasilkan model yang lebih baik untuk memprediksi *target* data.

Selanjutnya diberikan penjelasan mengenai fungsi *loss* yang bertujuan untuk mengukur kinerja model dengan cara membandingkan *output* dari fungsi aktivasi dengan *output* target.

2.3.1.3. Fungsi *Loss*

Performa model NN dapat dievaluasi berdasarkan jarak *target* hasil prediksi model dengan *target* sebenarnya. Fungsi *loss* yang umumnya digunakan pada model NN adalah fungsi *Mean-of-square error* (MSE) dan fungsi *cross-entropy* (Bishop, 2006). MSE digunakan jika *target* yang diprediksi bernilai riil seperti dalam masalah regresi. Sedangkan *cross-entropy* digunakan jika *target* yang diprediksi bernilai biner. Fungsi MSE digunakan sebagai fungsi *loss* pada penelitian ini.

Misalkan terdapat N pasangan data dan *target* yang dinotasikan dengan $\{\mathbf{x}_n, t_n\}$, $n = 1, 2, \dots, N$. Prediksi dari t_n dilakukan dengan menggunakan data \mathbf{x}_n dan model $y(\cdot)$. Misalkan *output* prediksi dinotasikan dengan $y(\mathbf{x}_n, \mathbf{w})$ dengan \mathbf{w} merupakan parameter bobot dalam $y(\cdot)$. MSE dapat diekspresikan dalam persamaan (2.14):

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \|y(\mathbf{x}_n, \mathbf{w}) - t_n\|^2. \quad (2.14)$$

Dengan meminimumkan fungsi *loss* MSE pada persamaan (2.14), model NN diharapkan menghasilkan prediksi target yang memiliki jarak terdekat dengan target sebenarnya. Oleh sebab itu, pembelajaran model NN memiliki tujuan mencari vektor bobot \mathbf{w} yang meminimumkan $\mathcal{L}(\mathbf{w})$. Pencarian nilai \mathbf{w} dilakukan dengan metode iteratif yang akan dijelaskan pada Subbab 2.3.1.4 dan 2.3.1.5.

2.3.1.4. *Error Backpropagation*

Error backpropagation (*backprop*) adalah algoritma pencarian gradien fungsi *loss* terhadap bobot dari NN. *Backprop* memiliki 2 keunggulan yaitu efektif secara komputasi dan fleksibel untuk diaplikasikan ke segala jenis fungsi *loss* (Bishop, 2006). Misalkan \mathcal{L} adalah fungsi *loss* pada persamaan (2.14). Gradien dari fungsi *loss* terhadap suatu bobot w_{ji} dapat diekspresikan dalam persamaan (2.15):

$$\frac{\partial \mathcal{L}}{\partial w_{ji}}. \quad (2.15)$$

Didefinisikan juga proses *forward propagation* dengan fungsi *loss* $\mathcal{L}(\mathbf{w})$, yaitu setiap neuron menghitung nilai aktivasi sesuai dengan persamaan (2.16) (Bishop, 2006):

$$a_j = \sum_i w_{ji} z_i, \quad (2.16)$$

dengan z_i adalah input neuron ke- i dan w_{ji} adalah bobot pada dari penghubung neuron i dengan neuron j . Bobot w_{j0} atau bias juga ditambahkan dan terhubung dengan neuron ke- j dengan $z_0 = 1$. Nilai aktivasi yang diperoleh dari persamaan (2.16) selanjutnya ditransformasikan menggunakan fungsi aktivasi $h(\cdot)$ sehingga *output* dari neuron ke- j adalah $z_j = h(a_j)$.

Turunan pertama dari fungsi *loss* \mathcal{L} terhadap w_{ji} ditentukan dalam proses *backprop*. Fungsi \mathcal{L} bergantung pada w_{ji} saat melewati fungsi aktivasi a_j , sehingga aturan rantai untuk turunan parsial digunakan untuk mencari turunan pertama fungsi \mathcal{L} terhadap w_{ji} . Turunan pertama fungsi \mathcal{L} terhadap w_{ji} ditunjukkan oleh persamaan (2.17):

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = \frac{\partial \mathcal{L}}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \quad (2.17)$$

Selanjutnya, turunan pertama dari a_j pada persamaan (2.16) terhadap w_{ji} dapat dilihat pada persamaan (2.18):

$$\frac{\partial a_j}{\partial w_{ji}} = z_i. \quad (2.18)$$

Kemudian, parameter δ_j didefinisikan dahulu seperti pada persamaan (2.19):

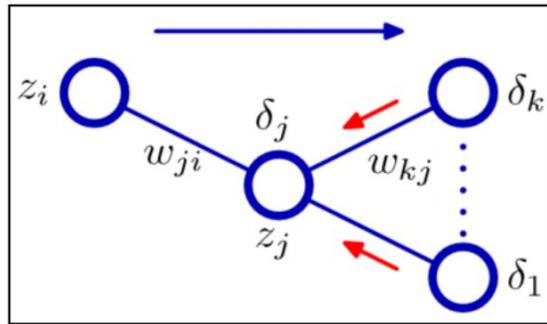
$$\delta_j = \frac{\partial \mathcal{L}}{\partial a_j}, \quad (2.19)$$

dan dengan mensubsitusikan persamaan (2.18) dan (2.19) pada (2.17), diperoleh persamaan (2.20):

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = \delta_j z_i. \quad (2.20)$$

Berdasarkan persamaan (2.20), dapat diketahui bahwa turunan dari fungsi \mathcal{L} terhadap w_{ji} sama dengan perkalian antara input dari neuron ke- i , z_i , dengan nilai δ_j . Hal ini juga menunjukkan bahwa δ_j untuk setiap neuron pada lapisan tersembunyi dan lapisan *output* adalah nilai yang perlu dicari untuk menentukan nilai turunan fungsi *loss* terhadap w_{ji} .

Perhitungan nilai δ_j diilustrasikan pada Gambar 2.7.



Gambar 2.7. Ilustrasi perhitungan nilai δ_j

Sumber: (Bishop, 2006)

Misalkan y_k adalah *output* dari neuron ke- k pada lapisan *output*, dan t_k adalah nilai *target* sebenarnya. Nilai δ_k pada lapisan *output* pada Gambar 2.7 yang terdiri dari K neuron dapat didefinisikan dengan persamaan (2.21) (Bishop, 2006):

$$\delta_k = y_k - t_k. \quad (2.21)$$

Selanjutnya, dengan mensubsitusikan $z_i = h(a_i)$ pada persamaan (2.16), dapat diperoleh $a_j = \sum_i w_{ji} h(a_i)$. Hal ini menunjukkan bahwa suatu unit aktivasi neuron ke- j , a_j , bergantung pada unit aktivasi neuron ke- i , a_i , yang terhubung dengan neuron ke- j . Oleh sebab itu, turunan pertama dari unit aktivasi a_j terhadap a_i dapat ditunjukkan pada persamaan (2.22):

$$\frac{\partial a_j}{\partial a_i} = w_{ji} h'(a_i). \quad (2.22)$$

Kemudian dengan mengacu pada Gambar 2.7 bahwa suatu neuron j tersambung dengan neuron $1, 2, \dots, k$, dapat dilakukan penurunan untuk menentukan nilai δ_j :

$$\begin{aligned} \delta_j &= \frac{\partial \mathcal{L}}{\partial a_j} \\ &= \sum_{i=1}^k \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial a_j} \\ &= \sum_{i=1}^k \delta_i w_{ij} h'(a_i) \end{aligned}$$

sehingga diperoleh persamaan (2.23):

$$\delta_j = \sum_{i=1}^k (y_i - t_i) w_{ij} h'(a_i). \quad (2.23)$$

Persamaan (2.23) disebut sebagai persamaan *backprop*. Pergerakan indeks i pada persamaan (2.23) mengimplikasikan aliran informasi yang berjalan mundur. Hal ini berkebalikan dengan pergerakan indeks pada *forward propagation* yang mengimplikasikan aliran informasi yang bergerak maju. Tanda panah pada Gambar 2.7 mengilustrasikan pergerakan *backprop* dan *forward propagation*. Tanda panah biru adalah aliran dari *forward propagation* sementara tanda panah merah adalah aliran informasi *backprop*. Hasil dari gradien yang diperoleh dari *backpropagation* kemudian dapat digunakan untuk memperbarui bobot menggunakan fungsi optimisasi. Pada Subbab 2.3.1.5 diberikan penjelasan mengenai beberapa jenis fungsi optimisasi yang umumnya digunakan.

2.3.1.5. Optimisasi

Pada bagian ini dijelaskan bagaimana memperbarui bobot menggunakan beberapa metode optimisasi yang umumnya digunakan pada pembelajaran model NN. Tiga metode yang dijelaskan pada bagian ini adalah *gradient descent*, *stochastic gradient descent*, dan *adaptive moment estimation*.

Gradient descent adalah salah satu metode optimisasi yang paling pertama digunakan dalam proses pembelajaran bobot pada model NN. *Gradient descent* memperbarui bobot secara iteratif hingga mencapai nilai optimal yang meminimumkan fungsi *loss*. Aturan pembaharuan dari metode *gradient descent* dapat dilihat pada persamaan (2.24):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \mathcal{L}(\mathbf{w}^{(\tau)}), \quad (2.24)$$

dengan η disebut sebagai *learning rate*. Metode *gradient descent* kurang efisien karena diperlukan penggunaan seluruh *dataset* untuk setiap pembaharuan bobot (Bishop, 2006). Sehingga digunakan metode lain yang merupakan pengembangan dari *gradient descent* yaitu *stochastic gradient descent* (SGD).

Dalam metode SGD, setiap pembaharuan bobot tidak menggunakan seluruh *dataset*, melainkan menggunakan subhimpunan dari data yang dipilih secara acak tanpa pengembalian (Ruder, 2016) dengan pembaharuan bobot SGD mengikuti persamaan (2.25):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \mathcal{L}^*(\mathbf{w}^{(\tau)}), \quad (2.25)$$

dengan $\nabla\mathcal{L}^*(\mathbf{w}^{(\tau)})$ adalah nilai gradien fungsi *loss* untuk suatu subhimpunan data. Pembaharuan bobot dalam satu iterasi dilakukan terus menerus terhadap subhimpunan data yang berbeda-beda menggunakan persamaan (2.25) sampai semua data sudah digunakan. Penggunaan metode SGD mengurangi beban komputasi karena penggunaan data yang lebih kecil ketika melakukan pembaharuan bobot (Ruder, 2016).

Metode *gradient descent* maupun SGD cukup sensitif terhadap pemilihan *learning rate*. *Learning rate* yang terlalu besar mengakibatkan metode menjadi divergen atau hanya berfluktuasi di sekitar titik minimum. Sebaliknya, *learning rate* yang terlalu kecil mengakibatkan konvergensi metode menjadi lambat. Oleh karena itu, dalam laju pembelajaran *gradient descent* dan SGD dapat digunakan momentum yang berguna untuk mempercepat konvergensi (Ruder, 2016).

Perpindahan suku $\mathbf{w}^{(\tau)}$ pada persamaan (2.24) menghasilkan persamaan (2.26).

$$\mathbf{w}^{(\tau+1)} - \mathbf{w}^{(\tau)} = -\eta \nabla\mathcal{L}(\mathbf{w}^{(\tau)}), \quad (2.26)$$

Vektor $\mathbf{w}^{(\tau+1)} - \mathbf{w}^{(\tau)}$ pada persamaan (2.26) didefinisikan ulang sebagai vektor $\Delta\mathbf{w}^{(\tau+1)}$. Selanjutnya, satu suku tambahan yang disebut suku momentum $p\Delta\mathbf{w}^{(\tau)}$ ditambahkan pada persamaan (2.26) untuk mempercepat konvergensi *gradient descent* atau SGD, sehingga diperoleh persamaan (2.27):

$$\Delta\mathbf{w}^{(\tau+1)} = -\eta \nabla\mathcal{L}(\mathbf{w}^{(\tau)}) + p\Delta\mathbf{w}^{(\tau)}, \quad (2.27)$$

dengan p adalah parameter momentum (Ruder, 2016).

Metode *adaptive moment estimation* (Adam) adalah metode pembaharuan bobot yang merupakan pengembangan dari *gradient descent* dan SGD. Sesuai dengan namanya, metode Adam menggunakan momentum yang disertakan dengan laju pembelajaran adaptif. Aturan pembaharuan bobot pada metode Adam mengikuti persamaan (2.28):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau+1)}} + \epsilon} \hat{\mathbf{m}}^{(\tau+1)}, \quad (2.28)$$

dengan nilai-nilai parameter pada persamaan (2.28) dapat diperoleh melalui persamaan (2.29) sampai (2.32):

$$\hat{\mathbf{v}}^{(\tau+1)} = \frac{\mathbf{v}^{(\tau+1)}}{1 - \beta_2^{\tau+1}}, \quad (2.29)$$

$$\mathbf{v}^{(\tau+1)} = \beta_2 \mathbf{v}^{(\tau)} + (1 - \beta_2) (\nabla\mathcal{L}(\mathbf{w}^{(\tau)}))^2, \quad (2.30)$$

$$\hat{\mathbf{m}}^{(\tau+1)} = \frac{\mathbf{m}^{(\tau+1)}}{1 - \beta_1^{\tau+1}}, \quad (2.31)$$

$$\mathbf{m}^{(\tau+1)} = \beta_1 \mathbf{m}^{(\tau)} + (1 - \beta_1) \nabla \mathcal{L}(\mathbf{w}^{(\tau)}), \quad (2.32)$$

dengan $\beta_1, \beta_2, \epsilon$ adalah suatu konstanta bernilai tetap. Persamaan (2.32) mengimplikasikan bahwa $\mathbf{m}^{(\tau+1)}$ adalah momentum yang mempertimbangkan nilai gradien fungsi *loss* pada iterasi sebelumnya sedemikian sehingga minimum *loss* lokal dapat dihindari. Persamaan (2.30) mengimplikasikan $\mathbf{v}^{(\tau+1)}$ adalah laju pembelajaran adaptif yang mempertimbangkan kuadrat dari nilai gradien fungsi *loss* pada iterasi sebelumnya. Hal ini bertujuan supaya pembaharuan dapat membesar ketika gradien bernilai kecil untuk menghindari gradien yang semakin mengecil, serta pembaharuan dapat mengecil ketika gradien bernilai besar untuk menghindari gradien yang terlalu besar. Parameter $\mathbf{m}^{(\tau+1)}$ dan $\hat{\mathbf{v}}^{(\tau+1)}$ pada Persamaan (2.29) dan (2.32) adalah *correction bias* dari momentum dan laju pembelajaran adaptif. *Correction bias* bertujuan untuk mencegah nilai momentum dan laju pembelajaran adaptif bias ke nilai 0, dikarenakan kedua parameter tersebut memiliki nilai yang cenderung kecil mendekati 0 pada awal iterasi awal.

Setelah diberikan penjelasan mengenai model NN dan fungsi-fungsi yang terkandung dalam mekanisme NN, Subbab 2.3.2 memberikan penjelasan mengenai arsitektur *autoencoder* yang memanfaatkan NN.

2.3.2. Arsitektur *Autoencoder*

Autoencoder adalah model *artificial neural network* (NN) yang memiliki dua bagian, yaitu *encoder* dan *decoder* (Hinton & Salakhutdinov, 2006). *Autoencoder* menerapkan pembelajaran secara *unsupervised*. Misalkan terdapat suatu input berupa vektor $\mathbf{x} = (x_1, x_2, \dots, x_D)^T$ dan juga vektor $\mathbf{z} = (z_1, z_2, \dots, z_d)^T$ dengan $D > d$. *Encoder* secara matematis adalah fungsi pemetaan nonlinear $f_\theta: X \rightarrow Z$ yang memetakan vektor \mathbf{x} dari ruang data X berdimensi D menuju suatu vektor \mathbf{z} di ruang laten Z dengan dimensi yang lebih rendah $d < D$ (Vincent *et al.*, 2010). *Encoder* dapat direpresentasikan sebagai fungsi pada persamaan (2.33):

$$\mathbf{z} = f_\theta(\mathbf{x}). \quad (2.33)$$

Sedangkan *decoder* secara umum merupakan fungsi pemetaan nonlinear $g'_\theta: Z \rightarrow X$ yang merekonstruksi vektor \mathbf{z} dari ruang laten Z menjadi suatu vektor $\mathbf{y} = (y_1, y_2, \dots, y_D)^T$ dari ruang data X (Vincent *et al.*, 2010). *Decoder* dapat direpresentasikan sebagai fungsi pada persamaan (2.34)

$$\mathbf{y} = g'_\theta(\mathbf{z}). \quad (2.34)$$

Parameter θ dan θ' pada (2.33) dan (2.34) secara berturut-turut merupakan bobot model NN pada *encoder* dan *decoder*.

Autoencoder melakukan pembelajaran dengan tujuan meminimumkan perbedaan antara data awal dengan data hasil rekonstruksi. Pembelajaran ini akan menghasilkan rekonstruksi terbaik dari input yang mampu membawa informasi penting yang terkandung di dalamnya. Dengan objektif tersebut, model *autoencoder* diharapkan dapat mempertahankan banyak informasi dari data sebenarnya (Vincent *et al.*, 2010) dan mampu merepresentasikan data yang sesungguhnya dalam dimensi yang lebih rendah (Goodfellow, Bengio, & Courville, 2016).

Selanjutnya, pada Subbab 2.4 diberikan penjelasan mengenai *clustering* dan beberapa algoritma *clustering* yang memanfaatkan konsep dari Subbab 2.2 dan 2.3.

2.4. *Clustering*

Unsupervised learning merupakan cabang *machine learning* yang bertujuan menemukan pola pada suatu data. Pada *unsupervised learning*, data yang digunakan tidak memiliki *label* sehingga model yang dilatih tidak memiliki pembanding yang dapat digunakan dalam pembelajaran. Salah satu contoh teknik penyelesaian secara *unsupervised learning* yang sering digunakan adalah *clustering* (Du & Swamy, 2013)

Metode *clustering* merupakan metode yang mengumpulkan data ke dalam beberapa *cluster* sedemikian sehingga data dalam *cluster* yang sama memiliki tingkat kesamaan tinggi, sementara data dalam *cluster* yang berbeda memiliki tingkat kesamaan minimum (Jain, 2010). Pada penelitian ini, diterapkan salah satu jenis *clustering* yaitu *centroid-based clustering*. *Centroid-based clustering* adalah metode yang menentukan tingkat similaritas dengan cara mengukur jarak antara titik ke pusat *cluster* (*centroid*) $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_D)$ (Yusdiansyah, Murfi, & Wibowo, 2019).

Ukuran jarak yang dapat dipakai untuk mengukur similaritas antar objek adalah jarak Euclidean, jarak Minkowski, atau jarak Manhattan (Uppada, 2014). Ukuran jarak

yang digunakan pada *centroid-based clustering* adalah jarak Euclidean. Misalkan teradapat dua titik pada ruang berdimensi D , yaitu $\mathbf{p} = (p_1, p_2, \dots, p_D)$ dan $\mathbf{q} = (q_1, q_2, \dots, q_D)$. Ukuran jarak Euclidean dapat dihitung dengan menggunakan Persamaan (2.35) (Pandit & Gupta, 2011).

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_D - q_D)^2}. \quad (2.35)$$

Clustering dapat dibagi menjadi dua jenis berdasarkan sifat keanggotannya, yaitu *soft clustering* dan *hard clustering*. Metode *soft clustering* adalah metode dimana suatu titik data dapat menjadi anggota untuk lebih dari satu *cluster* (Bezdek, Hall, & Clarke, 1993). Metode *hard clustering* adalah metode dimana satu titik data hanya bisa berada dalam tepat 1 *cluster* (Maji, Roy, & Biswas, 2002). Salah satu contoh algoritma yang menerapkan metode *soft clustering* adalah *fuzzy c-means*, sementara contoh algoritma yang menerapkan metode *hard clustering* adalah *k-means clustering*. Dalam penelitian ini, masalah yang menjadi perhatian adalah *text clustering*, yaitu proses *clustering* yang menggunakan data teks dengan algoritma yang digunakan adalah *k-means clustering*, *eigenspace-based fuzzy c-means*, *deep embedded clustering*, dan *improved deep embedded clustering*.

Selanjutnya, pada Subbab 2.4.1 sampai 2.4.4 dijelaskan beberapa jenis algoritma *clustering* yang digunakan penelitian ini.

2.4.1. *K-Means Clustering*

K-means clustering adalah algoritma yang mencari dan mendefinisikan *cluster* sebagai partisi dari data (Jain, 2010). Algoritma *K-Means Clustering* bertujuan untuk melakukan partisi terhadap N data berdimensi D menjadi K *cluster* dengan meminimumkan suatu fungsi objektif (Bishop, 2006). Untuk suatu himpunan data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ yang berdimensi D , fungsi objektif yang diminimumkan dapat dilihat pada persamaan (2.36),

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2. \quad (2.36)$$

Pada setiap titik data \mathbf{x}_n dengan $n = 1, \dots, N$, $r_{nk} \in \{0,1\}$ adalah nilai keanggotaan dari data \mathbf{x}_n pada *cluster* K . Jika titik data \mathbf{x}_n adalah anggota *cluster* K maka r_{nk} bernilai 1 dan r_{nj} bernilai 0 untuk $j \neq k$. Fungsi objektif J merupakan penjumlahan dari kuadrat

jarak antara setiap titik data \mathbf{x}_n dengan setiap *centroid* $\boldsymbol{\mu}_k$. Untuk meminimumkan fungsi objektif J , perlu ditentukan nilai r_{nk} dan $\boldsymbol{\mu}_k$ yang sesuai dengan melakukan prosedur iteratif yang melalui dua tahap, yaitu optimisasi r_{nk} dilanjutkan dengan optimisasi $\boldsymbol{\mu}_k$.

Tahap pertama adalah meminimumkan persamaan (2.36) terhadap r_{nk} dengan mengasumsikan nilai $\boldsymbol{\mu}_k$ tetap. Nilai r_{nk} pada fungsi objektif J bersifat linier dan independen terhadap nilai n yang berbeda. Hal ini mengakibatkan optimisasi dapat dilakukan dengan memasukkan data \mathbf{x}_n ke *cluster* dengan titik pusat terdekat. Secara matematis, nilai r_{nk} dapat diekspresikan dalam persamaan (2.37) (Bishop, 2006),

$$r_{nk} = \begin{cases} 1, & k = \arg \min_k \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \\ 0, & \text{yang lain} \end{cases}. \quad (2.37)$$

Tahap kedua adalah meminimumkan persamaan (2.36) terhadap $\boldsymbol{\mu}_k$ dengan mengasumsikan nilai r_{nk} tetap. Optimisasi dapat dilakukan dengan menentukan nilai $\boldsymbol{\mu}_k$ yang membuat nilai fungsi turunan pertama fungsi J terhadap $\boldsymbol{\mu}_k$ menjadi sama dengan nol. Penurunan optimisasi fungsi J terhadap nilai $\boldsymbol{\mu}_k$ untuk $\forall k = 1, 2, \dots, K$ dapat dilakukan sebagai berikut:

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{\mu}_k} &= \frac{\partial}{\partial \boldsymbol{\mu}_k} \left(\sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \right) \\ &= \frac{\partial}{\partial \boldsymbol{\mu}_k} (r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2) \\ &= \sum_{n=1}^N 2r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) (-1) \end{aligned}$$

dengan menyamakan penurunan di atas dengan nilai 0 diperoleh:

$$\begin{aligned} \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) &= \mathbf{0} \\ \sum_{n=1}^N r_{nk} \mathbf{x}_n - \sum_{n=1}^N r_{nk} \boldsymbol{\mu}_k &= \mathbf{0}, \end{aligned}$$

sehingga didapatkan nilai $\boldsymbol{\mu}_k$ yang akan mengoptimalkan nilai fungsi J , ditunjukkan pada persamaan (2.38) (Bishop, 2006):

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}} \quad (2.38)$$

Persamaan (2.38) menunjukkan bahwa nilai μ_k untuk suatu cluster K yang dapat mengoptimalkan fungsi objektif merupakan rata rata dari nilai data x_n yang masuk ke cluster tersebut. Sehingga dapat disimpulkan bahwa fungsi objektif pada persamaan (2.36) akan menjadi minimum apabila nilai $\{r_{nk}\}$ dan $\{\mu_k\}$ secara berturut-turut mengikuti persamaan (2.37) dan (2.38). Algoritma k -means clustering dapat dilihat pada Tabel 2.1.

Tabel 2.1. Algoritma k -means clustering

Algoritma 1 K-Means Clustering
<p>INPUT himpunan data $\{x_1, x_2, \dots, x_N\}$, banyak cluster K, toleransi ε, maksimum iterasi $maxiter$</p> <ol style="list-style-type: none"> 1. Inisialisasi secara acak pusat cluster $\{\mu_1, \mu_2, \dots, \mu_K\}$. 2. Tetapkan $iter = 0$ 3. REPEAT <ol style="list-style-type: none"> 4. Tetapkan $iter = iter + 1$ 5. Cari nilai $r_{nk} = \begin{cases} 1, & k = \arg \min_k \ x_n - \mu_k\ ^2 \\ 0, & \text{yang lain} \end{cases}$ 6. Cari nilai $\mu_k = \frac{\sum_{n=1}^N r_{nk} x_n}{\sum_{n=1}^N r_{nk}}$. 7. Cari nilai $J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \ x_n - \mu_k\ ^2$. 8. UNTIL $J < \varepsilon$ OR $iter > maxiter$ 9. STOP <p>OUTPUT nilai keanggotaan $\{r_{nk}\}$</p>

2.4.2. Eigenspace Based Fuzzy C Means

Pertama, akan dijelaskan terlebih dahulu mengenai algoritma *fuzzy c-means* (FCM). FCM merupakan algoritma pengelompokan data dimana setiap data masuk ke dalam suatu *cluster* berdasarkan derajat keanggotaan (Bezdek, Ehrlich, & Full, 1984). Konsep dari FCM adalah menentukan pusat *cluster*, yang merupakan lokasi rata-rata untuk setiap *cluster*, dan derajat keanggotaan setiap data secara iteratif. Pusat *cluster* akan berpindah ke posisi yang tepat dan nilai derajat keanggotaan akan menyesuaikan dan menuju ke *cluster* yang tepat. Iterasi ini dilakukan hingga nilai fungsi tujuan sudah berada di bawah ambang yang ditentukan atau banyak iterasi maksimum sudah tercapai. *Output*

dari FCM adalah barisan pusat *cluster* yang meminimumkan fungsi objektif serta derajat keanggotaan untuk setiap data pada setiap *cluster* yang ada.

Misalkan terdapat suatu himpunan data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ dan himpunan pusat *cluster* $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$ yang keduanya berdimensi D . Jumlah dari jarak setiap data ke pusat *cluster* dapat diekspresikan dengan fungsi J pada persamaan (2.39) (Bezdek, Ehrlich, & Full, 1984):

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2, \quad (2.39)$$

dengan $m > 1$ adalah derajat *fuzzy* dan r_{nk}^m menunjukkan derajat keanggotaan dari data ke- n pada *cluster* ke- k dengan tambahan diketahui derajat *fuzzy* m . Nilai r_{nk} menunjukkan seberapa besar kemungkinan suatu observasi bisa menjadi bagian dari suatu *cluster*. Untuk $n = 1, 2, \dots, N$ dan $k = 1, 2, \dots, K$, nilai r_{nk} memiliki beberapa batasan seperti yang ditunjukkan pada (2.40) (Bezdek, Ehrlich, & Full, 1984):

$$\begin{aligned} 0 \leq r_{nk} &\leq 1, \\ \sum_{k=1}^K r_{nk} &= 1. \end{aligned} \quad (2.40)$$

Clustering dengan menggunakan metode FCM bertujuan untuk meminimumkan fungsi jarak J pada (2.39) terhadap r_{nk} dan $\boldsymbol{\mu}_k$ seperti yang ditunjukkan pada persamaan (2.41):

$$\min_{r_{nk}, \boldsymbol{\mu}_k} J = \min_{r_{nk}, \boldsymbol{\mu}_k} \sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2, \quad (2.41)$$

dengan kendala (2.42):

$$\sum_{k=1}^K r_{nk} = 1, \quad k = 1, 2, \dots, K, \quad (2.42)$$

atau ekuivalen dengan (2.43):

$$\sum_{k=1}^K r_{nk} - 1 = 0. \quad (2.43)$$

Proses meminimumkan fungsi J menggunakan prosedur iteratif yang meliputi dua tahap, yaitu optimisasi nilai r_{nk} yang dilanjutkan dengan optimisasi nilai $\boldsymbol{\mu}_k$.

Tahap pertama adalah meminimumkan fungsi J terhadap nilai r_{nk} dengan mengasumsikan nilai μ_k . Sebelum menentukan nilai r_{nk} yang dapat meminimumkan fungsi J , akan dikenalkan terlebih dahulu konsep pengali Lagrange. Misalkan diberikan masalah pemrograman linier seperti yang ditunjukkan pada persamaan (2.44):

$$\min z = F(x_1, x_2, \dots, x_n), \quad (2.44)$$

dengan sebanyak I kendala yang dapat diekspresikan seperti pada persamaan (2.45):

$$\phi_i(x_1, x_2, \dots, x_n) = b_i, i = 1, 2, \dots, I. \quad (2.45)$$

Untuk menyelesaikan masalah pemrograman linier pada (2.44) dan (2.45), didefinisikan suatu fungsi tujuan Lagrange, L , yang diekspresikan dalam persamaan (2.46):

$$L = F(x_1, x_2, \dots, x_n) - \sum_{i=1}^I \lambda_i [\phi_i(x_1, x_2, \dots, x_n) - b_i], \quad (2.46)$$

dengan λ_i adalah pengali Langrange untuk kendala ke- i . Masalah optimisasi pada (2.41) dan (2.43) dapat didefinisikan ulang dengan menggunakan (2.44) sampai (2.46) sehingga diperoleh fungsi objektif baru yang ditunjukkan pada persamaan (2.47):

$$L = \left(\sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|x_n - \mu_k\|^2 \right) - \sum_{n=1}^N \lambda_n \left(\sum_{k=1}^K r_{nk} - 1 \right). \quad (2.47)$$

Nilai r_{nk} yang meminimumkan fungsi L pada (2.47) dapat dicari dengan mencari nilai r_{nk} yang membuat nilai fungsi turunan pertama (2.47) terhadap r_{nk} menjadi sama dengan 0. Penentuan nilai r_{nk} dapat diturunkan sebagai berikut

$$\begin{aligned} \frac{\partial L}{\partial r_{nk}} &= \frac{\partial}{\partial r_{nk}} \left(\left(\sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|x_n - \mu_k\|^2 \right) - \left(\sum_{n=1}^N \lambda_n \left(\sum_{k=1}^K r_{nk} - 1 \right) \right) \right) \\ &= \frac{\partial}{\partial r_{nk}} (r_{nk}^m \|x_n - \mu_k\|^2 - \lambda_n (r_{nk} - 1)) \\ &= \frac{\partial}{\partial r_{nk}} (r_{nk}^m \|x_n - \mu_k\|^2) - \frac{\partial}{\partial r_{nk}} (\lambda_n (r_{nk} - 1)) \\ &= m r_{nk}^{m-1} \|x_n - \mu_k\|^2 - \lambda_n \end{aligned}$$

Sehingga dengan menyamakan penurunan di atas dengan nol, diperoleh persamaan (2.48):

$$r_{nk} = \left(\frac{\lambda_n}{m \|x_n - \mu_k\|^2} \right)^{\frac{1}{m-1}}. \quad (2.48)$$

Nilai r_{nk} pada Persamaan (2.48) masih mengandung pengali Lagrange λ_n yang tidak diketahui, dengan mensubsitusikan (2.48) ke dalam (2.42) dapat diperoleh penurunan berikut:

$$\sum_{k=1}^K \left(\frac{\lambda_n}{m\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2} \right)^{\frac{1}{m-1}} = 1$$

$$\left(\frac{\lambda_n}{m} \right)^{\frac{1}{m-1}} \sum_{k=1}^K \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2} \right)^{\frac{1}{m-1}} = 1,$$

sehingga diperoleh persamaan (2.49):

$$\left(\frac{\lambda_n}{m} \right)^{\frac{1}{m-1}} = \frac{1}{\sum_{k=1}^K \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2} \right)^{\frac{1}{m-1}}}. \quad (2.49)$$

Selanjutnya dengan mensubsitusikan persamaan (2.49) ke (2.48) dan mengubah indeks k pada persamaan (2.49) menjadi j untuk menghindari kekeliruan, diperoleh penurunan berikut:

$$r_{nk} = \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2} \right)^{\frac{1}{m-1}} \left(\frac{1}{\sum_{j=1}^K \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2} \right)^{\frac{1}{m-1}}} \right)$$

$$= \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|} \right)^{\frac{2}{m-1}} \left(\frac{1}{\sum_{j=1}^K \left(\frac{1}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|} \right)^{\frac{2}{m-1}}} \right),$$

sehingga nilai r_{nk} yang mengoptimalkan fungsi objektif L dapat diperoleh melalui persamaan (2.50):

$$r_{nk} = \frac{1}{\sum_{j=1}^K \left(\frac{\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|}{\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|} \right)^{\frac{2}{m-1}}}. \quad (2.50)$$

Kemudian untuk tahap berikutnya, optimisasi nilai μ_k didapatkan dengan mencari nilai μ_k yang membuat turunan pertama dari fungsi objektif L terhadap μ_k menjadi sama dengan 0. Sehingga dapat diperoleh melalui penurunan berikut:

$$\begin{aligned}\frac{\partial L}{\partial \mu_k} &= \frac{\partial}{\partial \mu_k} \left(\left(\sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|x_n - \mu_k\|^2 \right) - \left(\sum_{n=1}^N \lambda_n \left(\sum_{k=1}^K r_{nk} - 1 \right) \right) \right) \\ &= \frac{\partial}{\partial \mu_k} \left(\sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \|x_n - \mu_k\|^2 \right) \\ &= \frac{\partial}{\partial \mu_k} (r_{nk}^m \|x_n - \mu_k\|^2) \\ &= \sum_{n=1}^N 2r_{nk}^m (x_n - \mu_k) (-1),\end{aligned}$$

dengan menyamakan hasil penurunan di atas dengan nol diperoleh persamaan berikut:

$$\begin{aligned}\sum_{n=1}^N r_{nk}^m (x_n - \mu_k) &= \mathbf{0} \\ \sum_{n=1}^N r_{nk}^m x_n - \sum_{n=1}^N r_{nk}^m \mu_k &= \mathbf{0},\end{aligned}$$

sehingga nilai μ_k yang meminimumkan fungsi objektif L dapat ditentukan dengan persamaan (2.51)

$$\mu_k = \frac{\sum_{n=1}^N r_{nk}^m x_n}{\sum_{n=1}^N r_{nk}^m} \quad (2.51)$$

Sehingga dapat disimpulkan bahwa kondisi optimal dari algoritma FCM adalah nilai r_{nk} dan μ_k yang secara berturut-turut ditentukan dari persamaan (2.50) dan persamaan (2.51).

FCM memiliki performa yang baik pada data berdimensi rendah namun akan gagal pada data berdimensi tinggi. FCM akan cenderung menghasilkan *centroid* yang sama pada data berdimensi tinggi (Winkler, Klawonn, & Kruse, 2011). Dalam rangka mengatasi masalah tersebut, data berdimensi tinggi perlu ditransformasi terlebih dahulu menjadi data berdimensi rendah sebelum FCM dilakukan (Alatas, Murfi, & Bustamam, 2018).

Seperti yang sudah dijelaskan pada Subbab 2.2, *truncated singular value decomposition* (TSVD) mendekomposisi matriks A berukuran $m \times n$ dan menghasilkan aproksimasi berupa $\tilde{U}\tilde{\Sigma}\tilde{V}^T$. Misalkan terdapat kumpulan vektor data yang disusun

menjadi suatu matriks $X = [\mathbf{x}_1 \ \mathbf{x}_2 \dots \mathbf{x}_n]$, dengan \mathbf{x}_i adalah vektor data ke- i berdimensi m . Matriks data X dapat diaproksimasi menggunakan TSVD sehingga diperoleh dekomposisi $X \approx \tilde{U}\tilde{\Sigma}\tilde{V}^T$. Kemudian matriks X direpresentasikan dengan matriks berdimensi lebih rendah dengan mengambil matriks $\tilde{\Sigma}\tilde{V}^T$ pada dekomposisi TSVD matriks X .

Didefinisikan $\tilde{X} = \tilde{\Sigma}\tilde{V}^T$ sebagai hasil reduksi dimensi dari matriks data X menggunakan metode TSVD. Matriks \tilde{X} menjadi input untuk tahap *clustering* FCM dan metode ini dikenal dengan *eigenspace-based fuzzy c-means*. Algoritma EFCM dapat dilihat pada Tabel 2.2.

Tabel 2.2. Algoritma *eigenspace-based fuzzy c-means*

Algoritma 2 Eigenspace-based Fuzzy C-Means
<p>INPUT himpunan data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, banyak <i>cluster</i> K, toleransi ε, maksimum iterasi <i>maxiter</i></p> <ol style="list-style-type: none"> 1. Susun himpunan data menjadi matriks $X = [\mathbf{x}_1 \ \mathbf{x}_2 \dots \mathbf{x}_N]^T$ 2. Reduksi dimensi matriks X dengan <i>TSVD</i> menjadi $\tilde{X} = \tilde{\Sigma}\tilde{V}^T = [\tilde{\mathbf{x}}_1 \ \tilde{\mathbf{x}}_2 \dots \tilde{\mathbf{x}}_N]^T$ 3. Inisialisasi secara acak pusat <i>cluster</i> $\{\boldsymbol{\mu}_1, \ \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$. 4. Tetapkan $iter = 0$ 5. REPEAT <ol style="list-style-type: none"> 6. Tetapkan $iter = iter + 1$ 7. Cari nilai $r_{nk} = \frac{1}{\sum_{j=1}^K \left(\frac{\ \mathbf{x}_n - \boldsymbol{\mu}_k\ }{\ \mathbf{x}_n - \boldsymbol{\mu}_j\ } \right)^{\frac{2}{m-1}}}$. 8. Cari nilai $\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk}^m \tilde{\mathbf{x}}_n}{\sum_{n=1}^N r_{nk}^m}$. 9. Hitung $J = \sum_{n=1}^N \sum_{k=1}^K r_{nk}^m \ \tilde{\mathbf{x}}_n - \boldsymbol{\mu}_k\ ^2$. 10. UNTIL $J < \varepsilon$ OR $iter > maxiter$ 11. Untuk setiap nilai $n = 1, \dots, N$ dan $k = 1, \dots, K$, tetapkan nilai $u_n = \arg \max_k (q_{nk})$ 12. STOP <p>OUTPUT nilai keanggotaan $\{u_n\}$</p>

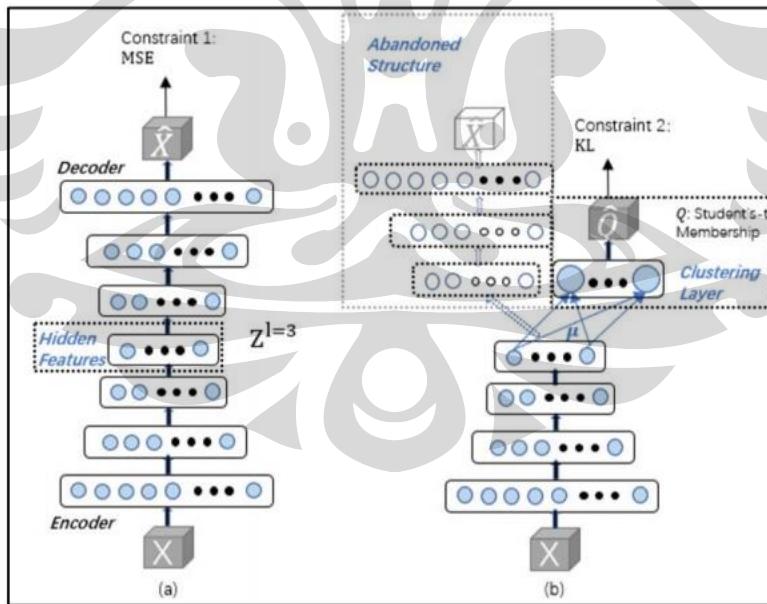
2.4.3. Deep Embedded Clustering

Penjelasan pada bagian ini merujuk pada Penelitian Xie, Girshick, & Farhadi (2016) kecuali disebutkan yang lain. Metode *clustering* diketahui bekerja kurang baik pada ruang

berdimensi tinggi dan menghasilkan *cluster* yang tidak efektif (Steinbach, Ertoz, & Kumar, 2004). Sehingga Xie, Girschik, dan Farhadi (2016) mengajukan suatu metode *clustering* yang dinamai *Deep Embedded Clustering* (DEC).

Metode DEC merupakan penelitian metode *clustering* mengenai ruang fitur. Optimisasi tidak hanya dilakukan pada *cluster* yang dihasilkan, namun juga mengoptimisasasi parameter pemetaan yang memetakan ruang data ke ruang laten secara simultan. Proses optimisasi secara simultan dapat meningkatkan kualitas *cluster* dan ruang fitur serta mengurangi kompleksitas waktu apabila dibandingkan dengan metode yang tidak melakukan optimisasi secara simultan.

Misalkan terdapat himpunan n data $\{\mathbf{x}_i \in X\}_{i=1}^n$ yang akan dikelompokkan ke dalam k *cluster*. Setiap *cluster* direpresentasikan oleh pusat *cluster* $\boldsymbol{\mu}_j$, $j = 1, \dots, k$. Dalam metode DEC, tahapan *clustering* tidak dilakukan dengan menggunakan ruang data X . Namun, data ditransformasikan terlebih dahulu melalui pemetaan non linier $f_\theta: X \rightarrow Z$ dengan θ adalah parameter yang dipelajari dan Z adalah ruang fitur laten. Parameterisasi fungsi f_θ dilakukan dengan menggunakan struktur NN. Arsitektur dari model DEC dapat dilihat pada Gambar 2.8b.



Gambar 2.8. (a) Arsitektur *deep autoencoder* (b) Arsitektur DEC

Sumber: (Feng, Chen, Chen, & Guo, 2020)

Seperti yang dijelaskan pada bagian 2.3, *deep autencoder* adalah salah satu model NN yang dapat digunakan untuk mereduksi dimensi data. Sehingga, reduksi dimensi pada DEC menggunakan *deep autoencoder* seperti yang ditunjukkan pada Gambar 2.8a. Model

DEC seperti yang ditunjukkan pada Gambar 2.8b tersusun atas dua komponen yaitu bagian *encoder* dan bagian *clustering (clustering layer)*. *Clustering layer* adalah lapisan tempat *clustering* dilakukan dengan menggunakan data yang dimensinya sudah tereduksi. Bobot pada *clustering layer* adalah titik pusat dari masing-masing *cluster*.

Metode DEC terdiri atas dua tahapan, tahap pertama adalah inisialisasi parameter θ dan parameter pusat *cluster* μ_j yang dijelaskan pada Subbab 2.4.3.1. Sementara tahap kedua adalah optimisasi parameter θ dan pusat *cluster* μ_j secara simultan yang dijelaskan pada Subbab 2.4.3.2.

2.4.3.1. Inisialisasi Parameter DEC

Tahap pertama dari DEC adalah inisialisasi parameter θ dengan menggunakan *deep autoencoder* (DAE). Setiap lapisan pada jaringan DAE diinisialisasi satu demi satu dengan menggunakan *denoising autoencoder*. *Denoising autoencoder* adalah 2 buah lapisan NN yang didefinisikan pada persamaan (2.52) sampai (2.55):

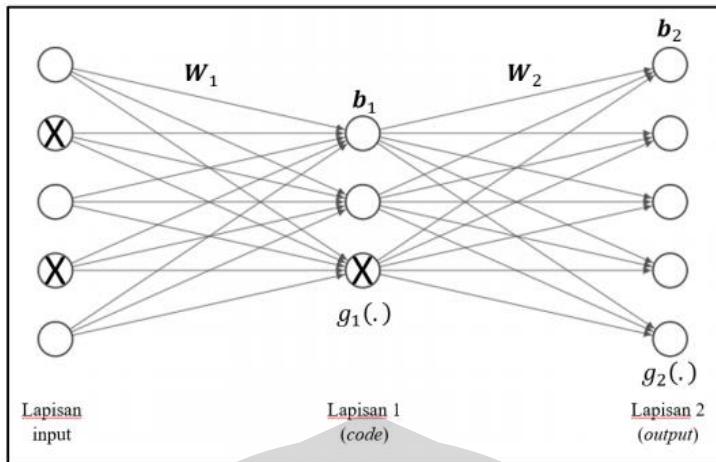
$$\tilde{x} \sim \text{Dropout}(x), \quad (2.52)$$

$$h = g_1(W_1\tilde{x} + b_1), \quad (2.53)$$

$$\tilde{h} \sim \text{Dropout}(h), \quad (2.54)$$

$$y = g_2(W_2\tilde{h} + b_2), \quad (2.55)$$

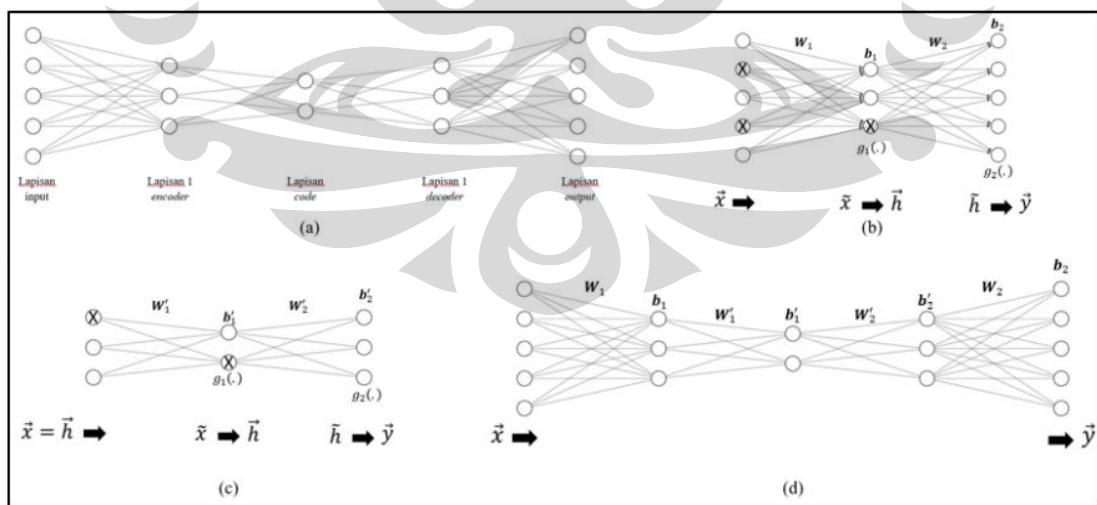
dengan *Dropout*(.) adalah pemetaan yang secara acak mengubah sebagian komponen input menjadi 0, fungsi g_1 dan g_2 secara berturut-turut adalah fungsi aktivasi untuk proses *encoding* dan *decoding*, W_1 dan W_2 adalah parameter bobot dari model *encoder* dan *decoder*, dan b_1 serta b_2 adalah parameter bias dari model *encoder* dan *decoder*. Pembelajaran dilakukan dengan melakukan minimalisasi MSE seperti yang dipaparkan pada bagian 2.3.1.3. *Output h* yang diperoleh dari (2.53) digunakan sebagai input bagi lapisan berikutnya. Struktur *denoising autoencoder* diilustrasikan Gambar 2.9.



Gambar 2.9. Struktur denoising autoencoder

Sumber: (Pradana, 2020)

Kemudian, SAE melakukan pembelajaran dengan menggunakan algoritma *greedy layer-wise training*. Setelah tahapan pembelajaran, semua lapisan *encoder* dari *denoising autoencoder* digabungkan, disusul dengan penggabungan semua lapisan *decoder* yang memiliki urutan lapisan yang terbalik dari urutan *encoder*. Prosedur dilanjutkan dengan proses *fine tuning* untuk meminimumkan *reconstruction loss* yaitu MSE. Tahap akhir dari prosedur ini adalah lapisan *decoder* dihilangkan dan lapisan *encoder* digunakan untuk pemetaan awal antara ruang data dengan ruang laten. Ilustrasi inisialisasi parameter θ dengan menggunakan *deep autoencoder* dapat dilihat pada Gambar 2.10.



Gambar 2.10. Proses pembelajaran autoencoder dengan *greedy layer-wise pretraining* (a) Autoencoder yang dibangun (b) Pembangunan lapisan 1 *encoder* (c) Pembangunan lapisan *code* (d) tahap *finetuning* dari autoencoder

Sumber: (Pradana, 2020)

Proses inisialisasi *centroid* dari *cluster* dilakukan dengan memasukkan data ke dalam struktur *encoder* yang sebelumnya digunakan untuk mendapatkan data dengan dimensi yang tereduksi. Kemudian algoritma *k-means clustering* diimplementasikan dengan menggunakan data pada ruang laten untuk mendapatkan *centroid* awal $\{\boldsymbol{\mu}_j\}_{j=1}^k$.

2.4.3.2. Optimisasi Parameter DEC

Setelah inisialisasi parameter, dilakukan tahap selanjutnya dari metode DEC yaitu optimisasi parameter secara simultan. Pada tahap ini, proses perhitungan *soft assignment* q_{ij} dan penentuan distribusi bantuan p_{ij} diiterasikan secara bergantian. Parameter q_{ij} dan p_{ij} dijelaskan pada bagian selanjutnya.

Perhitungan *soft assignment* dilakukan dengan menggunakan distribusi Student-t. Dengan bantuan distribusi tersebut, similaritas antara data tereduksi \mathbf{z}_i dengan pusat *cluster* $\boldsymbol{\mu}_j$ dapat ditentukan. Fungsi *kernel* berdistribusi Student-t dapat dilihat pada persamaan (2.56):

$$q_{ij} = \frac{\left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_j \left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}, \quad (2.56)$$

dengan $\mathbf{z}_i = f_\theta(\mathbf{x}_i) \in Z$ adalah data ke-*i* \mathbf{x}_i , yang dipetakan ke ruang laten, $\boldsymbol{\mu}_j$ adalah pusat *cluster* ke-*j*, dan α adalah derajat kebebasan dari distribusi Student-t. Fungsi *kernel* berdistribusi Student-t digunakan dalam perhitungan *soft assignment* supaya jarak kedua titik di dimensi tinggi akan memiliki jarak yang jauh lebih besar di ruang laten. Hal ini mempertahankan similaritas antara kedua titik ketika berada di ruang laten.

Nilai q_{ij} dapat diinterpretasikan sebagai probabilitas data ke-*i* berada pada cluster *j*, atau bisa disebut dengan nilai *soft assignment* dari data ke-*i*. Nilai $\sum_j q_{ij}$ adalah 1 karena keseluruhan probabilitas dari data ke-*i* masuk ke seluruh *cluster* adalah 1. Nilai q_{ij} yang tinggi mengindikasikan jarak data ke-*i* yang semakin dekat dengan *cluster* ke-*j*.

Tahapan selanjutnya adalah memilih distribusi bantuan P . Dalam metode DEC, distribusi bantuan P atau nilai p_{ij} dapat dihitung dengan terlebih dahulu mengkuadratkan

nilai q_{ij} kemudian menormalisasikannya menurut frekuensi setiap *cluster*. Tahapan ini diekspresikan dalam persamaan (2.57):

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \quad (2.57)$$

dengan $f_j = \sum_i q_{ij}$ adalah frekuensi dari *cluster* ke- j .

Pembaharuan *cluster* dilakukan secara iteratif dengan melakukan pembelajaran dari *assignment* yang memiliki tingkat kepercayaan tinggi pada distribusi bantuan. Model melakukan pembelajaran dengan mencocokkan *soft assignment* dengan distribusi tujuan. Objektif dari algoritma *cluster* adalah meminimumkan *loss* yaitu meminimumkan divergensi Kullback-Leibler (KL) antara *soft assignments* q_i dengan distribusi bantuan p_i . Fungsi *loss* ini dapat didefinisikan dalam persamaan (2.58):

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right). \quad (2.58)$$

Optimisasi pusat *cluster* $\{\mu_j\}$ dan parameter θ dari model NN dilakukan secara bersamaan dengan menggunakan SGD dengan momentum seperti yang dijelaskan pada Subbab 2.3.1.6. Fungsi *loss* L pada persamaan (2.58) bergantung pada pemetaan data di ruang laten \mathbf{z}_i dan pusat *cluster* μ_i ketika parameter q_{ij} dan p_{ij} disubsitusi dengan persamaan (2.56) dan (2.57). Sehingga, gradien fungsi L yang digunakan adalah gradien L terhadap \mathbf{z}_i dan μ_j . Gradien tersebut diekspresikan dalam persamaan (2.59) dan (2.60):

$$\frac{\partial L}{\partial z_i} = \frac{\alpha + 1}{\alpha} \sum_j \left(1 + \frac{\|\mathbf{z}_i - \mu_j\|^2}{\alpha} \right)^{-1} \times (p_{ij} - q_{ij})(\mathbf{z}_i - \mu_j), \quad (2.59)$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha + 1}{\alpha} \sum_i \left(1 + \frac{\|\mathbf{z}_i - \mu_j\|^2}{\alpha} \right)^{-1} \times (p_{ij} - q_{ij})(\mathbf{z}_i - \mu_j). \quad (2.60)$$

Gradien dari $\partial L/\partial z_i$ kemudian digunakan model NN dalam *backpropagation* standar untuk kemudian menghitung gradien dari parameter NN $\partial L/\partial \theta$ sehingga bisa memperbarui parameter θ . Sementara gradien $\partial L/\partial \mu_j$ digunakan untuk optimisasi parameter μ_j pada *cluster*. Prosedur ini akan dihentikan ketika banyaknya perubahan di antara dua iterasi yang berurutan berada dibawah batas toleransi yang ditentukan. Algoritma DEC dapat dilihat pada Tabel 2.3.

Tabel 2.3. Algoritma *deep embedded clustering*

Algoritma 3 Deep Embedded Clustering
<p>INPUT himpunan data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, banyak <i>cluster</i> K, toleransi ε, banyak lapisan <i>encoder</i> m, dimensi data d, dimensi laten d', dimensi setiap lapisan tersembunyi dari <i>encoder</i> $u_1 - u_2 - \dots - u_{m-1}$.</p> <ol style="list-style-type: none"> 1. Bangun <i>denoising autoencoder</i> dengan dimensi lapisan $d - u_i - d$ dan dipelajari untuk meminimumkan fungsi <i>loss</i> $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\ ^2$, dengan $y(\mathbf{x}_n, \mathbf{w})$ adalah <i>output</i> dari <i>denoising autoencoder</i>. 2. Tetapkan $i = 2$. 3. WHILE $i \leq m - 1$ DO: <ol style="list-style-type: none"> 4. Definisikan $\mathbf{h}_n^{(i-1)}$ sebagai <i>output</i> dari lapisan laten dari <i>denoising autoencoder</i> sebelumnya. 5. Bangun <i>denoising autoencoder</i> dengan dimensi lapisan $u_{i-1} - u_i - u_{i-1}$ dan dipelajari untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{h}_n^{(i-1)}, \mathbf{w}) - \mathbf{h}_n^{(i-1)}\ ^2$, dengan $y(\mathbf{h}_n^{(i-1)}, \mathbf{w})$ adalah <i>output</i> dari <i>denoising autoencoder</i>. 6. Tetapkan $i = i + 1$. 7. END WHILE 8. Bangun <i>denoising autoencoder</i> dengan dimensi setiap lapisan $u_{m-1} - d' - u_{m-1}$ dan dipelajari untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{h}_n^{(m-1)}, \mathbf{w}) - \mathbf{h}_n^{(m-1)}\ ^2$, dengan $y(\mathbf{h}_n^{(m-1)}, \mathbf{w})$ adalah <i>output</i> dari <i>denoising autoencoder</i>. 9. Sambungkan seluruh bagian <i>encoder</i> dari <i>denoising autoencoder</i> yang diperoleh dari langkah 1-5. Kemudian sambungkan struktur tersebut dengan seluruh bagian <i>decoder</i> dari <i>denoising autoencoder</i> yang diperoleh dari langkah 1-5 dengan urutan yang berkebalikan dengan urutan <i>encoder</i>. Diperoleh <i>autoencoder</i> dengan dimensi setiap lapisan $d - u_1 - u_2 - \dots - u_{m-1} - d' - u_{m-1} - \dots - u_2 - u_1 - d.$ <ol style="list-style-type: none"> 10. Pelajari <i>autoencoder</i> untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\ ^2$ dengan $y(\mathbf{x}_n, \mathbf{w})$ adalah <i>output</i> dari <i>autoencoder</i>.

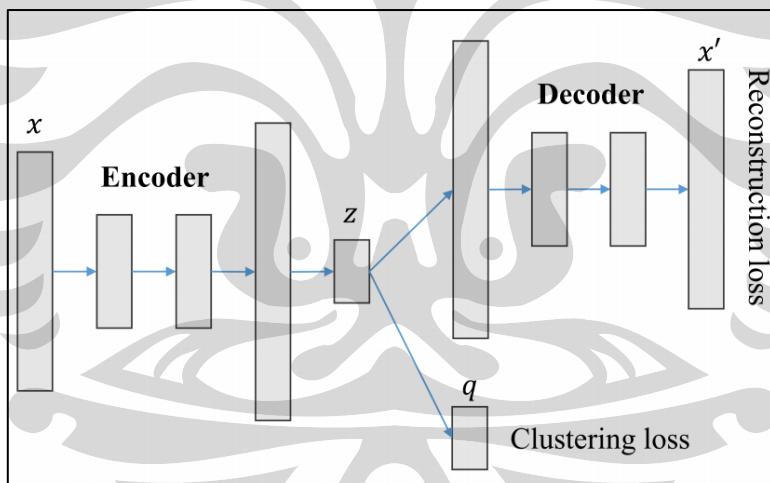
11. Reduksi dimensi data dengan menggunakan bagian *encoder* dari *autoencoder* sehingga diperoleh data berdimensi rendah $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$
 12. Lakukan *k-means clustering* (Algoritma 1) pada $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$ untuk inisialisasi pusat *cluster* $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$ dan keanggotaan $\mathbf{z}_n, \{u_n\}$
 13. Bangun dan sambungkan lapisan *clustering* pada lapisan laten *autoencoder* dan buang bagian *decoder* pada *autoencoder*. Dimensi lapisan *clustering* adalah K dan bobot masing-masing neuron adalah $\boldsymbol{\mu}_j, j = 1, 2, \dots, K$.
 14. REPEAT
 15. Input data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ke *encoder* sehingga diperoleh $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$
 16. Simpan *assignment cluster* terakhir $u' = u$
 17. Hitung nilai q_{ij} pada lapisan *clustering*
$$q_{ij} = \frac{\left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_j \left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}$$
 18. Hitung $p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$
 19. Hitung nilai $loss L = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right)$
 20. Tetapkan $u_i = \arg \max_j (q_{ij})$
 21. Tetapkan $U =$ banyak perubahan antara u' dengan u
 22. Jika $U/N > \varepsilon$, perbarui bobot pada *encoder* dan pada lapisan *clustering* menggunakan *backpropagation*
 23. UNTIL $\delta = U/N < \varepsilon$
 24. STOP
- OUTPUT** nilai keanggotaan $\{u_n\}$

2.4.4. Improved Deep Embedded Clustering

Improved deep embedded clustering (IDEC) adalah pengembangan dari metode DEC (Xie *et al.*, 2016). Model DEC memiliki kelemahan yaitu pada saat mempelajari *encoder* menggunakan *clustering loss*, ada distorsi pada ruang laten akibat bagian

decoder yang dibuang. Metode IDEC memperbaiki kelemahan tersebut dengan tetap menggunakan bagian *decoder* dan menempelkan *clustering loss* pada ruang laten. Metode ini disebut sebagai pelestarian struktur lokal dari *autoencoder*. Ilustrasi dari struktur IDEC dapat dilihat pada Gambar 2.11. *Autoencoder* dipelajari dengan meminimumkan *reconstruction loss*. Proses pembelajaran awal yang dilakukan pada IDEC sama dengan yang dilakukan pada DEC. Penjelasan pada bagian ini merujuk pada penelitian Guo *et al.* (2017) kecuali disebutkan yang lain.

Misalkan terdapat himpunan yang berisikan n data dengan setiap data memiliki dimensi sebesar d , $x_i \in \mathbb{R}^d$. Banyaknya *cluster* K sudah diketahui sebelumnya dan pusat *cluster* direpresentasikan dengan $\mu_j \in \mathbb{R}^d$. Nilai dari $s_i \in \{1, 2, \dots, K\}$ merepresentasikan indeks dari *cluster* yang diberikan pada masing-masing data x_i . Definisikan juga pemetaan non linier berupa $f_W: x_i \rightarrow z_i$ dan $g_w: z_i \rightarrow x'_i$ dengan z_i adalah x_i yang dimensinya tereduksi menjadi suatu dimensi yang lebih rendah di ruang laten dan x'_i adalah hasil rekonstruksi x_i .



Gambar 2.11. Ilustrasi struktur model *improved deep embedded clustering*

Sumber: (Guo et al., 2017)

Metode IDEC bertujuan untuk menemukan pemetaan f_W yang baik sehingga hasil data yang tereduksi $\{\mathbf{z}_i\}_{i=1}^n$ cocok digunakan dalam permasalahan *clustering*. Dua komponen penting yang perlu diperhatikan adalah *autoencoder* dan *clustering loss*. *Autoencoder* digunakan untuk mempelajari representasi dari data secara *unsupervised* sehingga fitur yang diperoleh dapat melestarikan struktur lokal dari data sebenarnya. *Clustering loss* berperan dalam memanipulasi ruang laten sehingga titik yang sudah

dipetakan di ruang laten lebih tersebar. Tujuan dari metode IDEC dapat diekspresikan pada persamaan (2.16):

$$L = L_r + \gamma L_c, \quad (2.61)$$

dengan L_r dan L_c secara berturut-turut adalah *reconstruction loss* dan *clustering loss*, dan $\gamma > 0$ adalah koefisien yang mengendalikan derajat tingkat distorsi dari ruang laten. Ketika $\gamma = 1$ dan $L_r \equiv 0$ maka objektif pada persamaan (2.61) akan menjadi objektif dari model DEC.

Serupa dengan metode DEC, metode IDEC terbagi menjadi dua tahapan, yaitu tahapan inisialisasi yang dijelaskan pada Subbab 2.4.4.1, dan tahap optimisasi dengan memanfaatkan pelestarian struktur lokal yang dijelaskan pada Subbab 2.4.4.2.

2.4.4.1. Inisialisasi dan *Clustering Loss*

Clustering loss L_c yang digunakan pada model IDEC adalah divergensi Kullback-Leibler yang merupakan *loss* yang digunakan model DEC pada persamaan (2.58). Dalam melakukan inisialisasi parameter θ pada *autoencoder*, pendekatan serupa dengan model DEC juga dilakukan. Seperti yang dijelaskan pada Subbab 2.4.3.1, Pembelajaran terhadap *denoising autoencoder* dilakukan dan data yang tereduksi dari *deep autoencoder* digunakan untuk melakukan *clustering*. Pusat *cluster* awal $\{\boldsymbol{\mu}_j\}_{j=1}^K$ diperoleh dengan menggunakan algoritma *k-means clustering* pada $\{\mathbf{z}_i = f_W(\mathbf{x}_i)\}_{i=1}^n$.

2.4.4.2. Pelestarian Struktur Lokal

Titik data yang tereduksi ke ruang laten belum tentu cocok digunakan sebagai input dalam permasalahan *clustering*. Model DEC (Xie *et al.*, 2016) membuang struktur *decoder* dan melakukan *fine tuning* pada *encoder* terhadap L_c . Namun, Teknik ini dinilai dapat melakukan distorsi terhadap ruang laten dan memperburuk hasil representasi yang diperoleh sehingga juga mempengaruhi performa *clustering*. Oleh karena itu, struktur *decoder* pada metode IDEC tetap dipertahankan dan dilakukan penambahan berupa struktur *clustering* pada lapisan ruang laten.

Pada proses pembelajaran model IDEC, *noise* pada *denoising autoencoder* yang digunakan dalam proses pembelajaran awal tidak digunakan lagi untuk menjamin efektivitas *clustering loss*. Hal ini disebabkan karena *clustering* seharusnya dilakukan tidak dengan menggunakan data yang sudah diberikan *noise* secara acak seperti pada

stacked denoising autoencoder, melainkan dengan menggunakan fitur dari data bersih. *Reconstruction loss* L_r yang digunakan adalah MSE yang sudah dijelaskan pada Subbab 2.3.1.3. Dengan menggunakan struktur ini, *autoencoder* dapat tetap melestarikan struktur lokal (Peng *et al.*, 2016; Goodfellow *et al.*, 2016) dan modifikasi ruang laten menggunakan *clustering loss* tidak akan mengakibatkan distorsi.

Optimisasi dilakukan menggunakan *mini-batch stochastic gradient descent* (SGD) dan *backpropagation*. Parameter yang akan dioptimisasi dan diperbaharui adalah bobot dari *autoencoder*, pusat *cluster*, dan distribusi bantuan P . Gradien dari L_c terhadap z_i dan μ_j secara berturut-turut sudah ditunjukkan dalam persamaan (2.59) dan (2.60) pada Subbab 2.4.2. Algoritma IDEC dapat dilihat pada Tabel 2.4.

Tabel 2.4. Algoritma *improved deep embedded clustering*

Algoritma 4 Improved Deep Embedded Clustering
<p>INPUT himpunan data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, banyak <i>cluster</i> K, toleransi ε, banyak lapisan <i>encoder</i> m, dimensi data d, dimensi laten d', dimensi setiap lapisan tersembunyi dari <i>encoder</i> $u_1 - u_2 - \dots - u_{m-1}$</p> <ol style="list-style-type: none"> 1. Bangun <i>denoising autoencoder</i> dengan dimensi lapisan $d - u_i - d$ dan dipelajari untuk meminimumkan fungsi <i>loss</i> $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\ ^2$, dengan $y(\mathbf{x}_n, \mathbf{w})$ adalah <i>output</i> dari <i>denoising autoencoder</i> 2. Tetapkan $i = 2$ 3. WHILE $i \leq m - 1$ DO: <ol style="list-style-type: none"> 4. Definisikan $\mathbf{h}_n^{(i-1)}$ sebagai <i>output</i> dari lapisan laten dari <i>denoising autoencoder</i> sebelumnya 5. Bangun <i>denoising autoencoder</i> dengan dimensi lapisan $u_{i-1} - u_i - u_{i-1}$ dan dipelajari untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ y(\mathbf{h}_n^{(i-1)}, \mathbf{w}) - \mathbf{h}_n^{(i-1)}\ ^2$, dengan $y(\mathbf{h}_n^{(i-1)}, \mathbf{w})$ adalah <i>output</i> dari <i>denoising autoencoder</i> 6. Tetapkan $i = i + 1$ 7. END WHILE

8. Bangun *denoising autoencoder* dengan dimensi setiap lapisan $u_{m-1} - d' - u_{m-1}$ dan dipelajari untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \|y(\mathbf{h}_n^{(m-1)}, \mathbf{w}) - \mathbf{h}_n^{(m-1)}\|^2$, dengan $y(\mathbf{h}_n^{(m-1)}, \mathbf{w})$ adalah *output* dari *denoising autoencoder*
9. Sambungkan seluruh bagian *encoder* dari *denoising autoencoder* yang diperoleh dari langkah 1-5. Kemudian sambungkan struktur tersebut dengan seluruh bagian *decoder* dari *denoising autoencoder* yang diperoleh dari langkah 1-5 dengan urutan yang berkebalikan dengan urutan *encoder*. Diperoleh *autoencoder* dengan dimensi setiap lapisan

$$d - u_1 - u_2 - \cdots - u_{m-1} - d' - u_{m-1} - \cdots - u_2 - u_1 - d$$

10. Pelajari *autoencoder* untuk meminimumkan $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \|y(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2$ dengan $y(\mathbf{x}_n, \mathbf{w})$ adalah *output* dari *autoencoder*.
11. Reduksi dimensi data dengan menggunakan bagian *encoder* dari *autoencoder* sehingga diperoleh data berdimensi rendah $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$
12. Lakukan *k-means clustering* (Algoritma 1) pada $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$ untuk inisialisasi pusat *cluster* $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$ dan keanggotaan data pada *cluster* $\{u_n\}$.
13. Bangun dan sambung lapisan *clustering* pada lapisan laten. Kemudian hapus semua *noise* pada *autoencoder*. *Reconstruction loss* dipelajari dengan menggunakan fungsi $\mathcal{L}_r = \sum_{n=1}^N \|g(\mathbf{z}_n) - \mathbf{x}_n\|^2$, dengan $g(\mathbf{z}_n)$ adalah *output* dari bagian *decoder*. Sementara *clustering loss* dipelajari dengan menggunakan fungsi $\frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$.

14. REPEAT

15. Input data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ke *encoder* sehingga diperoleh $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$
16. Simpan *assignment cluster* terakhir $u' = u$
17. Hitung nilai q_{ij} pada lapisan *clustering*

$$q_{ij} = \frac{\left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_j \left(1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}$$

$$18. \text{ Hitung } p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$$

19. Tetapkan $u_i = \arg \max_j (q_{ij})$
 20. Tetapkan $U =$ banyak perubahan antara u' dengan u
 21. Jika $U/N > \varepsilon$, perbarui bobot pada *encoder* dan pada lapisan *clustering* menggunakan *backpropagation*
 22. UNTIL $\delta = U/N < \varepsilon$
 23. STOP
- OUTPUT** nilai keanggotaan $\{u_n\}$



BAB 3

BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER

Pada bab ini dijelaskan mengenai metode representasi data teks yang diusulkan pada penelitian ini. Pembahasan pertama mengenai model transformers pada Subbab 3.1. Selanjutnya, dibahas mengenai model *bidirectional encoder representation from transformer* (BERT) pada Subbab 3.2. Pembahasan terakhir adalah mengenai implementasi model BERT sebagai representasi data teks yang dibahas pada Subbab 3.3.

3.1. *Transformer*

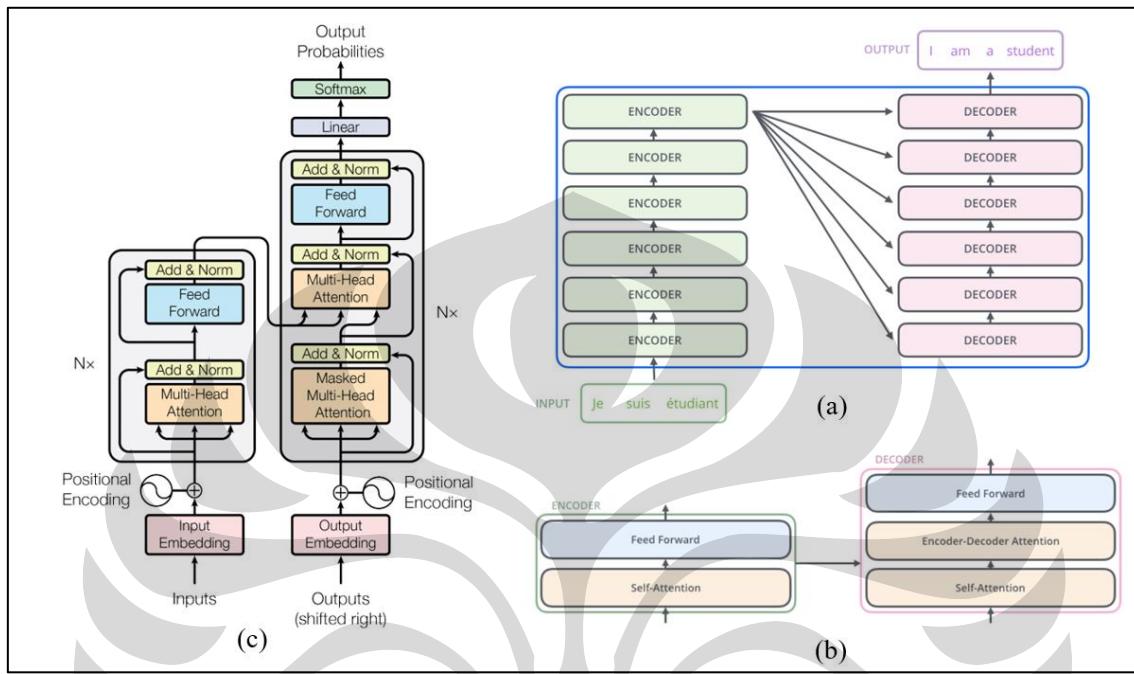
Transformer adalah arsitektur *Artificial Neural Network* (NN) yang dikembangkan Vaswani *et al.* (2017). Struktur yang dimiliki *transformer* berbeda dibandingkan dengan RNN dan CNN karena *transformer* mengandalkan mekanisme *multi-head self-attention* pada setiap lapisan *encoder* maupun *decoder*. Arsitektur jaringan ini terbukti memiliki performa tinggi dalam menyelesaikan masalah yang berkaitan dengan *Natural Language Processing* seperti *syntactic parsing* dan *language translation* melalui *sequence transduction* (Sutskever, Vinyals, & Le, 2014; Bahdanau, Cho, & Bengio, 2014; Cho *et al.*, 2014). Pada subbab 3.1.1 sampai 3.1.5 akan dijelaskan mengenai arsitektur model *transformers* dan berbagai mekanisme yang menyusun model *transformers*.

3.1.1. Arsitektur Model *Transformer*

Arsitektur dari *transformer* mengimplementasikan struktur pasangan *encoder-decoder*. Ilustrasi dari arsitektur *transformer* dapat dilihat pada Gambar 3.1. Lapisan *encoder* berfungsi untuk memahami konteks suatu kata dalam dokumen atau kalimat, sementara lapisan *decoder* digunakan untuk menyelesaikan masalah translasi menuju bahasa berbeda.

Pada *transformer* banyaknya lapisan *encoder* sama dengan banyaknya lapisan *decoder*. Seperti yang ditunjukkan pada Gambar 3.1a, struktur *encoder* tersusun atas 6 lapisan identik yang ditumpuk dan digabungkan. Setiap lapisan *encoder* tersusun atas 2 buah sub-lapisan seperti yang ditunjukkan pada gambar bertepi hijau pada Gambar 3.1b. Sub-lapisan pertama adalah bagian mekanisme *multi-head self-attention* dan sub-lapisan kedua adalah bagian *position-wise fully connected feed-forward network*. Koneksi

residual diterapkan di antara setiap 2 sub-lapisan, diikuti dengan lapisan normalisasi seperti yang terlihat pada kotak kuning *Add & Norm* pada Gambar 3.1c. Penjelasan mengenai *multi-head self attention*, *position-wise fully connected feed-forward neural network*, dan koneksi residual akan dijelaskan pada subbab selanjutnya.



Gambar 3.1. Ilustrasi arsitektur *transformer*

Sumber: (Vaswani et al., 2017; Alammar, 2018), telah diolah kembali

Seperti pada Gambar 3.1a, struktur *decoder* juga tersusun atas tumpukan dari sebanyak 6 lapisan identik. Lapisan *decoder* mengandung 3 sub-lapisan, yaitu 2 sub-lapisan pada struktur *encoder*, dan sub-lapisan yang menjalankan *multi-head attention* pada *output* dari tumpukan *encoder*. Hal ini diilustrasikan pada gambar bertipe merah muda pada Gambar 3.1b. Serupa dengan *encoder*, koneksi residual yang diikuti dengan lapisan normalisasi juga diterapkan di antara setiap sub-lapisan. Sub-lapisan *multi-head self-attention* dimodifikasi pada tumpukan *decoder* menggunakan metode *masking* seperti yang ditunjukkan kotak jingga *Masked Multi-Head Attention* pada Gambar 3.1c.

3.1.2. *Attention*

Fungsi *attention* adalah fungsi yang menghasilkan nilai berupa besaran atensi suatu kata yang sedang ditinjau ke kata-kata lain. Misalkan terdapat himpunan vektor $\{x_1, x_2, \dots, x_n\}$ berdimensi d_{model} yang merupakan input untuk suatu lapisan *encoder*. Himpunan vektor ini merupakan vektor representasi numerik dari himpunan kata dalam

dokumen. Untuk suatu input \mathbf{x}_i , didefinisikan vektor *query* berdimensi d_k $\mathbf{q}_i \in \mathbb{R}^{d_k}$ dan pasangan vektor *key-value*, yang masing-masing berdimensi d_k dan d_v , $\mathbf{k}_i \in \mathbb{R}^{d_k}$ dan $\mathbf{v}_i \in \mathbb{R}^{d_v}$. Nilai vektor \mathbf{q}_i , \mathbf{k}_i , dan \mathbf{v}_i diperoleh melalui persamaan (3.1) sampai (3.3):

$$\mathbf{q}_i = \mathbf{x}_i W^Q, \quad (3.1)$$

$$\mathbf{k}_i = \mathbf{x}_i W^K, \quad (3.2)$$

$$\mathbf{v}_i = \mathbf{x}_i W^V, \quad (3.3)$$

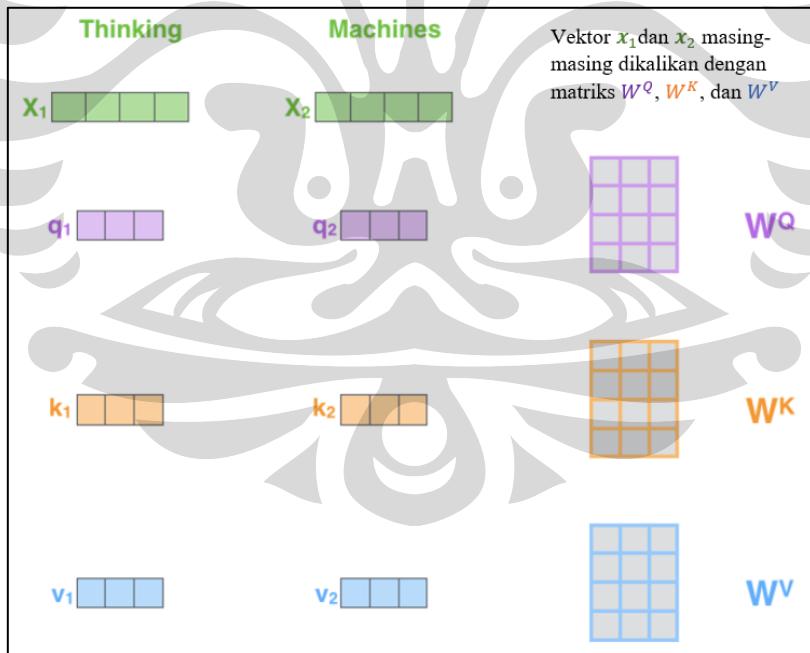
dengan keterangan sebagai berikut:

$W^Q \in M_{d_{model} \times d_v}(\mathbb{R})$ = matriks parameter bobot untuk menentukan nilai vektor \mathbf{q}_i ,

$W^K \in M_{d_{model} \times d_v}(\mathbb{R})$ = matriks parameter bobot untuk menentukan nilai vektor \mathbf{k}_i ,

$W^V \in M_{d_{model} \times d_k}(\mathbb{R})$ = matriks parameter bobot untuk menentukan nilai vektor \mathbf{v}_i .

Sebagai ilustrasi, misalkan terdapat dua kata dalam dokumen, yaitu *thinking* dan *machines*. Kata *thinking* dan *machines* secara berturut-turut memiliki vektor representasi numerik \mathbf{x}_1 dan \mathbf{x}_2 . Gambar 3.2 mengilustrasikan penentuan nilai vektor *query*, *key*, dan *value* untuk kata *thinking* dan *machine*:

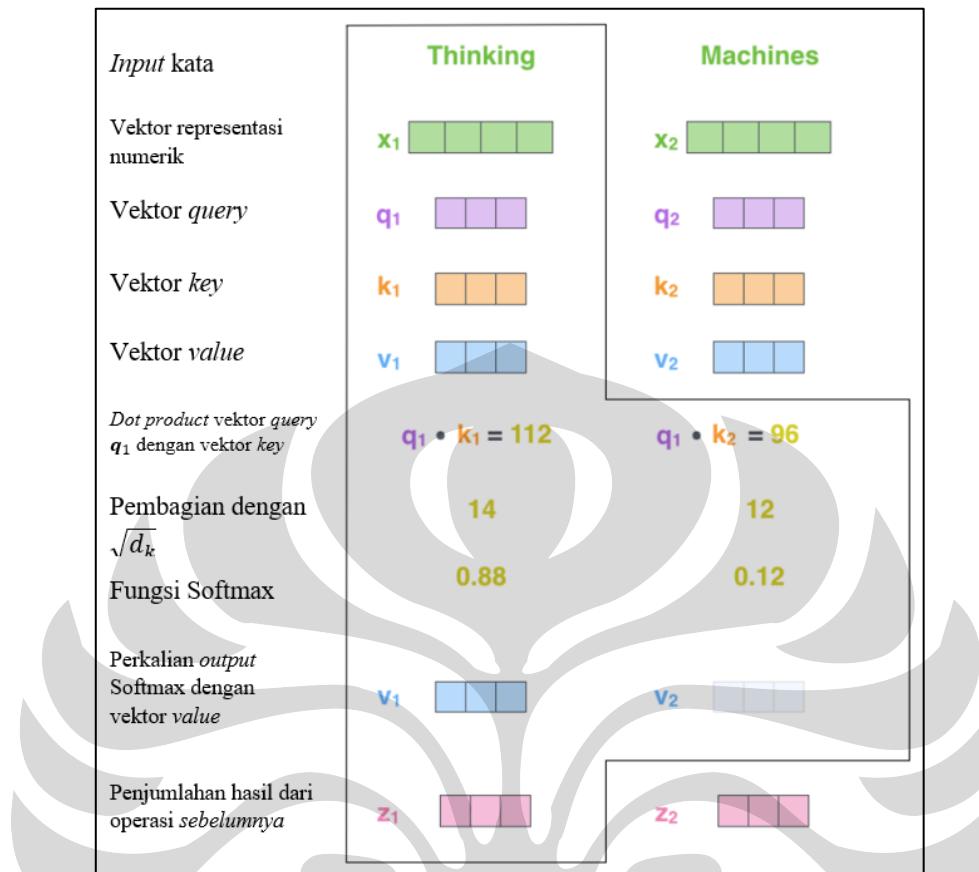


Gambar 3.2. Ilustrasi penentuan nilai vektor *query*, *key*, dan *value*

Sumber: (Alammar, 2018), telah diolah kembali

Vektor \mathbf{x}_1 yang merepresentasikan kata *thinking* dikalikan dengan W^Q , W^K , dan W^V sehingga menghasilkan vektor \mathbf{q}_1 , \mathbf{k}_1 , dan \mathbf{v}_1 . Operasi serupa dilakukan untuk vektor \mathbf{x}_2 menghasilkan vektor \mathbf{q}_2 , \mathbf{k}_2 , dan \mathbf{v}_2 .

Operasi pada fungsi *attention* dapat diilustrasikan pada Gambar 3.3:



Gambar 3.3. Ilustrasi fungsi *attention*

Sumber: (Alammar, 2018), telah diolah kembali

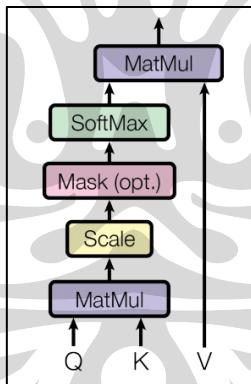
Misalkan terdapat dua kata dalam dokumen, yaitu *thinking* dan *machines*, dengan vektor representasi numerik, *query*, *key*, dan *value* yang sudah dijelaskan pada ilustrasi Gambar 3.2. Misalkan dimensi vektor *query* $d_k = 64$. Ilustrasi ini menentukan besaran atensi kata *thinking* ke kata-kata yang lain, yaitu *thinking* itu sendiri dan *machines*. Pertama, operasi *dot product* antara vektor *key* dengan vektor *value* dilakukan. Vektor *query* yang digunakan dalam *dot product* adalah q_1 karena kata yang sedang ditinjau adalah kata pertama yaitu *thinking*. Dalam ilustrasi ini, misalkan diperoleh hasil *dot product* sebesar 112 dan 96 seperti pada Gambar 3.3. Selanjutnya, hasil *dot product* dibagi dengan $\sqrt{d_k} = \sqrt{64} = 8$. Kemudian, hasil pembagian dimasukkan ke dalam fungsi Softmax menggunakan persamaan (2.12) sehingga diperoleh hasil berikut:

$$f(14) = \frac{e^{14}}{e^{14} + e^{12}} = 0.88,$$

$$f(12) = \frac{e^{12}}{e^{14} + e^{12}} = 0.12.$$

Selanjutnya, nilai yang diperoleh dari fungsi *softmax* dikalikan dengan vektor *value* \mathbf{v}_1 dan \mathbf{v}_2 . Pada tahap ini, vektor \mathbf{v}_2 diilustrasikan berwarna lebih muda dibandingkan dengan \mathbf{v}_1 karena nilai fungsi Softmax yang dikalikan ke \mathbf{v}_1 jauh lebih besar dibandingkan \mathbf{v}_2 . Terakhir, vektor *value* yang sudah dikalikan dengan nilai fungsi Softmax dijumlahkan sehingga diperoleh vektor *output* akhir \mathbf{z}_1 . Penentuan besaran atensi kata *machines* ke kata *thinking* dan *machines* dapat dilakukan dengan tahapan yang sama, namun mengganti vektor *query* pada proses *dot product* menjadi \mathbf{q}_2 . Sehingga untuk kata *machines* diperoleh vektor *output* akhir \mathbf{z}_2 .

Himpunan vektor *queries*, *keys*, dan *values* secara berturut-turut dapat disusun dalam suatu matriks $Q = [\mathbf{q}_1 \mathbf{q}_2 \dots \mathbf{q}_n]^T$, $keys K = [\mathbf{k}_1 \mathbf{k}_2 \dots \mathbf{k}_n]^T$, dan $values V = [\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_n]^T$. Secara umum, fungsi *attention* yang diilustrasikan pada Gambar 3.3 dapat dideskripsikan sebagai pemetaan matriks Q, K , dan V ke suatu matriks *output* dengan memanfaatkan operasi *scaled dot product*. Ilustrasi dari operasi *scaled dot product* dapat dilihat pada Gambar 3.4.



Gambar 3.4. Ilustrasi *scaled dot product*

Sumber: (Vaswani et al., 2017)

Matriks Q yang dikalikan dengan matriks K^T menghasilkan nilai yang menunjukkan besaran atensi suatu kata yang sedang ditinjau ke kata-kata lain. Hasil perkalian tersebut kemudian dikalikan dengan invers dari akar dimensi $keys \frac{1}{\sqrt{d_k}}$.

Pemberian *mask* juga dilakukan pada beberapa kasus tertentu yang akan dijelaskan pada bagian berikutnya. Kemudian hasil tersebut dimasukkan ke dalam fungsi *softmax* sehingga menghasilkan bobot untuk matriks *values* V . Terakhir, bobot tersebut dikalikan

dengan matriks *values* V untuk memperoleh matriks *output*. Dengan demikian, *output* dari fungsi *attention* merupakan *output* dari operasi *scaled dot product* yang diekspresikan pada persamaan (3.4):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (3.4)$$

Alasan perkalian $\frac{1}{\sqrt{d_k}}$ digunakan adalah ukuran dimensi d_k yang relatif besar menyebabkan magnitudo dari perkalian titik membesar. Hal ini mengakibatkan hasil fungsi *softmax* kurang efektif karena lebih cenderung menuju ke daerah dengan gradien yang sangat kecil.

Fungsi *attention* digunakan pada mekanisme *multi-head attention* yang dijelaskan pada Subbab 3.1.2.1.

3.1.2.1. *Multi-head attention*

Misalkan matriks Q , K , dan V adalah matriks *queries*, *keys*, dan *values* yang dijelaskan pada Subbab 3.1.2. *Multi-head attention* adalah arsitektur yang melakukan fungsi *attention* sebanyak h kali secara simultan dengan menggunakan matriks Q , K , dan V yang berbeda. Arsitektur *multi-head attention* bertujuan untuk menghasilkan sebanyak h besaran atensi yang berbeda untuk setiap kata, sehingga menangkap berbagai kemungkinan ketergantungan yang berbeda. Mekanisme *multi-head attention* dengan nilai $h = 8$ dapat diilustrasikan pada Gambar 3.5.

Misalkan terdapat himpunan vektor $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ berdimensi d_{model} . Pada lapisan *encoder* yang pertama, himpunan vektor ini merupakan vektor representasi numerik dari himpunan kata dalam dokumen. Sementara pada lapisan *encoder* lainnya, himpunan vektor $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ merupakan himpunan vektor *output* dari lapisan *encoder* sebelumnya. Jika matriks $X \in M_{n \times d_{model}}(\mathbb{R})$ adalah susunan himpunan vektor input $X = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_n]^T$, maka untuk setiap $i \in \{1, 2, \dots, h\}$, nilai Q_i , K_i , dan V_i secara berturut-turut dapat diperoleh melalui persamaan (3.5) sampai (3.7)

$$Q_i = XW_i^Q, \quad (3.5)$$

$$K_i = XW_i^K, \quad (3.6)$$

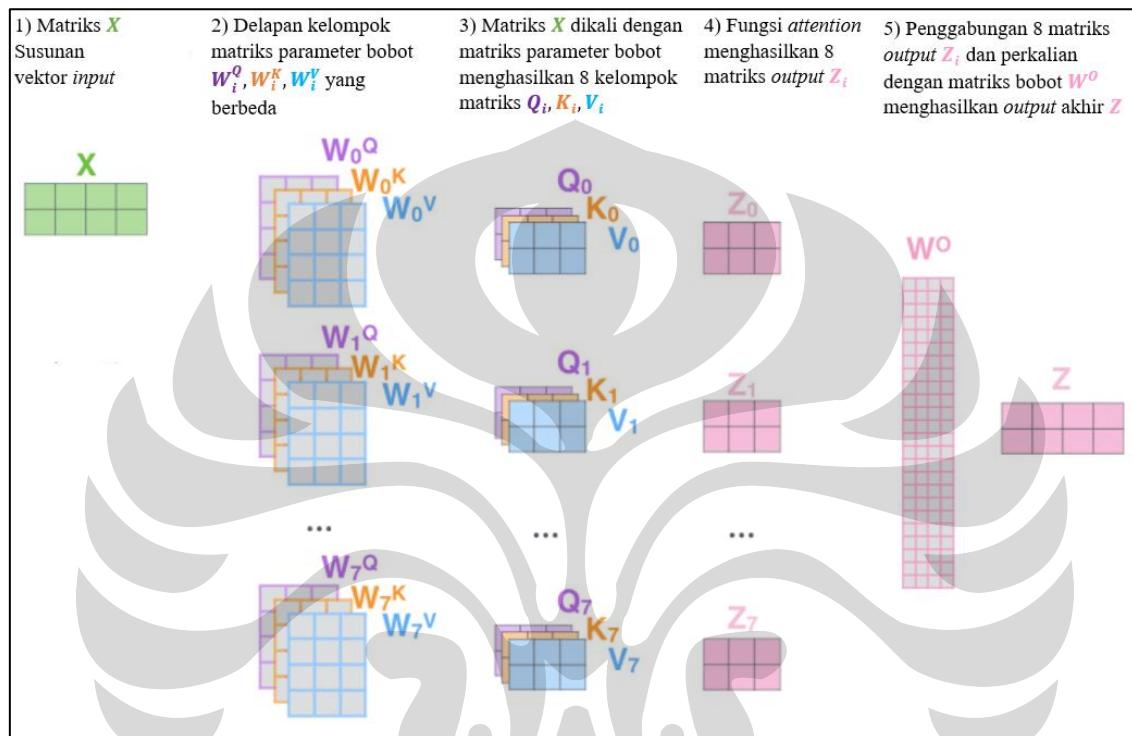
$$V_i = XW_i^V, \quad (3.7)$$

dengan keterangan sebagai berikut:

$W_i^Q \in M_{d_{model} \times d_k}(\mathbb{R})$ = matriks bobot ke- i untuk menentukan nilai matriks Q_i ,

$W_i^K \in M_{d_{model} \times d_k}(\mathbb{R})$ = matriks bobot ke- i untuk menentukan nilai matriks K_i ,

$W_i^V \in M_{d_{model} \times d_v}(\mathbb{R})$ = matriks bobot ke- i untuk menentukan nilai matriks V_i .



Gambar 3.5. Ilustrasi *multi-head attention*

Sumber: (Alammar, 2018), telah diolah kembali

Untuk setiap $i \in \{1, 2, \dots, h\}$, dilakukan operasi *scaled dot product* (3.4) secara simultan menggunakan Q_i , K_i , dan V_i sehingga menghasilkan sebanyak h matriks *output* berukuran $n \times d_v$. Hasil *output* ini kemudian digabungkan dan dipetakan kembali menggunakan matriks parameter bobot $W_O \in M_{hd_v \times d_{model}}(\mathbb{R})$. Akibatnya, *output* dari proses tersebut berdimensi sama dengan *output* dari fungsi *attention* (3.4). *Multi-head attention* dapat diekspresikan dengan persamaan (3.8):

$$\text{MultiHead}(Q, K, V) = \text{Concat}(Z_1, Z_2, \dots, Z_h)W^O, \quad (3.8)$$

dengan *Concat()* adalah operasi penggabungan dan nilai Z_i diperoleh dari persamaan (3.9):

$$Z_i = \text{Attention}(Q_i, K_i, V_i). \quad (3.9)$$

Pada penelitian ini menggunakan nilai $h = 12$.

3.1.2.2. Penerapan *Attention* dalam Model *Transformer*

Transformer menggunakan *multi-head attention* pada tiga kasus berbeda yang dapat dilihat pada Gambar 3.1c. Kasus pertama adalah pada sub-lapisan *multi-head attention* yang menghubungkan antara lapisan *encoder* dengan *decoder*. Pada kasus ini, *multi-head attention* menggunakan *queries* yang diperoleh dari pemetaan input lapisan *decoder* dan menggunakan *keys* serta *values* yang diperoleh dari pemetaan *output* lapisan *encoder* terakhir. Pengaturan ini mengakibatkan setiap posisi pada *decoder* memberikan atensi ke seluruh posisi pada *encoder*.

Kasus kedua adalah pada struktur *encoder* yang mengandung lapisan *self-attention*. Berbeda dengan penjelasan sebelumnya dimana *queries* dan *keys* serta *values* diperoleh dari pemetaan *sequence* yang berbeda, *self-attention* adalah mekanisme yang menggunakan *keys*, *queries*, dan *values* dari pemetaan *sequence* yang sama. Pada struktur *encoder*, *sequence* yang digunakan berasal dari *embedding* pada awal proses atau dari *encoder* pada tumpukan sebelumnya. Setiap posisi pada *encoder* dapat memberikan atensi semua posisi pada lapisan *encoder* sebelumnya.

Kasus ketiga adalah lapisan *self-attention* pada *decoder* yang membatasi setiap posisi pada *decoder* untuk tidak memperhatikan posisi setelah posisi tersebut. Aliran informasi dari posisi yang lebih tinggi (lebih ke kanan dalam susunan kalimat) perlu dihindari untuk menghindari kebocoran informasi. Implementasi ini dilakukan dengan memberikan *mask*, yaitu dengan memberikan nilai $-\infty$ pada setiap entri *values* yang berkaitan dengan aliran informasi yang tidak diinginkan.

Pada Subbab 3.1.3 dijelaskan mengenai arsitektur *transformer* berikutnya yaitu *position-wise feed forward network*.

3.1.3. *Position-wise Feed Forward Network*

Position-wise feed forward network adalah arsitektur *artificial neural network*. Arsitektur ini terdiri atas dua kali transformasi linier dengan fungsi aktivasi ReLU di antara kedua transformasi linier. Misalkan z_{ki} adalah entri baris ke- k dan kolom ke- i dari matriks Z yang diperoleh pada Subbab 3.1.2.1. Parameter $w_{ji}^{(1)}$ adalah bobot dari sambungan neuron ke- i pada lapisan pertama dengan neuron ke- j pada lapisan kedua dan $w_{j0}^{(1)}$ adalah parameter bias antara lapisan pertama dengan kedua. Untuk suatu kata pada

posisi ke- k dan neuron ke- j , transformasi linier pertama yang disertai dengan fungsi aktivasi ReLU diekspresikan pada persamaan (3.10):

$$\max\left(0, \sum_i w_{ji}^{(1)} z_{ki} + w_{j0}^{(1)}\right). \quad (3.10)$$

Output yang diperoleh dari fungsi pada (3.10) menjadi input untuk transformasi linier kedua sehingga menghasilkan persamaan (3.11):

$$FFN(m) = \sum_j w_{mj}^{(2)} \max\left(0, \sum_i w_{ji}^{(1)} z_{ki} + w_{j0}^{(1)}\right) + w_{l0}^{(2)}, \quad (3.11)$$

dengan $FFN(m)$ adalah *output* pada neuron ke- m dari lapisan *output*. Dimensi dari input dan *output* jaringan yang digunakan pada penelitian ini adalah $d_{model} = 768$, dengan lapisan dalam memiliki dimensi $d_{ff} = 3072$.

Setelah dijelaskan mengenai *attention* pada 3.1.2 dan arsitektur *position-wise feed forward network* pada 3.1.3, pada Subbab 3.1.4 dijelaskan mengenai sub-lapisan koneksi residu dan normalisasi yang pada Subbab 3.1.1 sudah dijelaskan akan berada di antara setiap 2 sub-lapisan.

3.1.4. Koneksi Residu dan Normalisasi

Konsep koneksi residual diperkenalkan pada model ResNet, dan merupakan pemetaan yang menghubungkan input pada suatu lapisan dengan *output* pada lapisan tersebut (He, Zhang, Ren, & Sun, 2016). Misalkan Z adalah *output* yang dihasilkan pada sub-lapisan tersebut dan X adalah input dari sub-lapisan tersebut, persamaan (3.12) menunjukkan hasil yang diperoleh dari koneksi residu Z' :

$$Z' = Z + X. \quad (3.12)$$

Selanjutnya diterapkan *layer normalization* yang bertujuan untuk menghindari *covariate shift* (Ioffe & Szegedy, 2015). Proses ini merupakan standarisasi terhadap neuron pada lapisan tersembunyi h_i , sehingga hasil yang diperoleh diaproksimasikan berdistribusi normal dengan rata-rata 0 dan standar deviasi 1. Proses normalisasi dapat diekspresikan dengan persamaan (3.13):

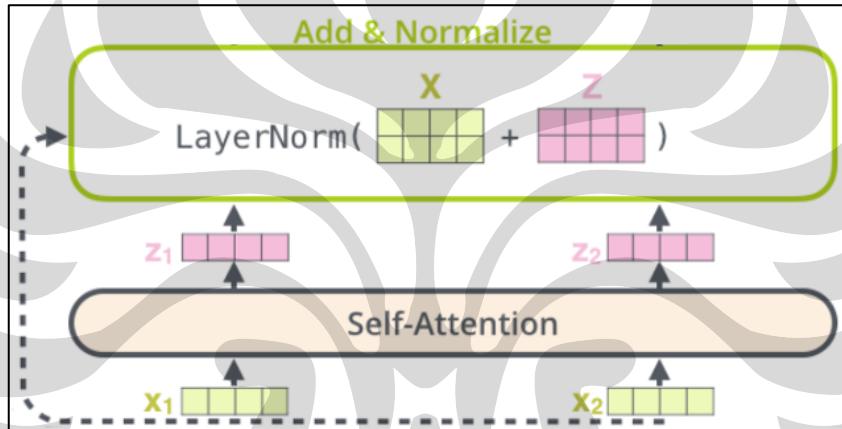
$$h_i = \frac{g}{\sigma}(h_i - \mu), \quad (3.13)$$

dengan parameter μ dan σ secara berturut-turut adalah rata-rata dan standar deviasi dari sebanyak H neuron tersembunyi yang dapat diperoleh menggunakan persamaan (3.14) dan (3.15):

$$\mu = \frac{1}{H} \sum_{i=1}^H h_i, \quad (3.14)$$

$$\sigma = \sqrt{\sum_{i=1}^H (h_i - \mu)^2}, \quad (3.15)$$

Sebagai ilustrasi, Gambar 3.6 menunjukkan salah satu penerapan koneksi residu dan normalisasi setelah sub-lapisan *self-attention* pada *transformer encoder*:



Gambar 3.6. Ilustrasi koneksi residu dan normalisasi pada *transformer*

Sumber: (Alammar, 2018), telah diolah kembali

Output sub-lapisan *self-attention* pada lapisan *encoder* melalui proses koneksi residu dan normalisasi terlebih dahulu sebelum menjadi input bagi sub-lapisan *position-wise feed forward network*. Pada Gambar 3.6, terdapat 2 vektor x_1 dan x_2 yang menjadi input sub-lapisan *self-attention*. Vektor x_1 dan x_2 masing-masing menghasilkan *output* berupa vektor z_1 dan z_2 . Vektor x_1 dan x_2 dapat disusun menjadi matriks $X = [x_1 \ x_2]^T$ dan vektor z_1 dan z_2 dapat disusun menjadi matriks $Z = [z_1 \ z_2]^T$. Pertama, koneksi residu dengan menggunakan persamaan (3.12) menghasilkan matriks $X + Z$. Kemudian, nilai $X + Z$ yang diperoleh dimasukkan ke dalam fungsi *LayerNorm* atau persamaan (3.13) sehingga diperoleh *output* akhir yang akan menjadi input untuk sub-lapisan *position-wise feed forward network*.

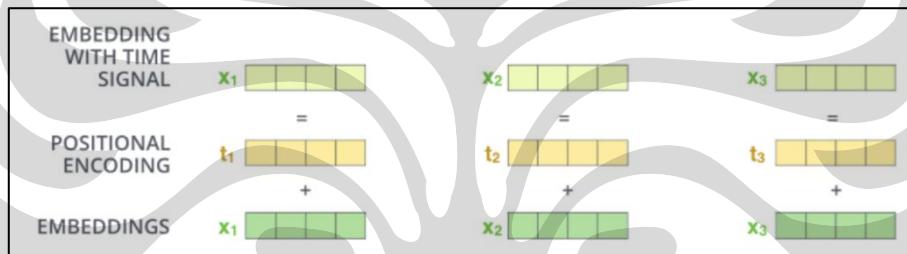
3.1.5. Positional Encoding

Positional Encoding menambahkan informasi mengenai posisi kata pada suatu kalimat, pada bagian ini akan disebut sebagai *sequence*. Hal ini dibutuhkan karena model *Transformer* tidak memiliki sifat rekursi maupun konvolusi yang memberikan informasi mengenai posisi. Pada model *transformers*, *positional encoding* menggunakan fungsi sinusoidal yang diekspresikan pada persamaan (3.16) dan (3.17)

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (3.16)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d_{model}}}}\right), \quad (3.17)$$

dengan *pos* menyatakan posisi dan *i* menyatakan dimensi.



Gambar 3.7. Ilustrasi penjumlahan antara *input embeddings* dan *positional encodings*

Sumber: (Lionbridge, 2019)

Persamaan (3.16) dan (3.17) menunjukkan bahwa setiap dimensi untuk suatu posisi kata akan berkorespondensi dengan suatu sinusoida. Penggunaan fungsi sinusoidal ini mengakibatkan model dapat melakukan ekstrapolasi. Ekstrapolasi berguna ketika *sequence* yang diuji lebih panjang dari *sequence* pada saat proses pembelajaran. Ilustrasi dari penjumlahan ini dapat dilihat pada Gambar 3.7.

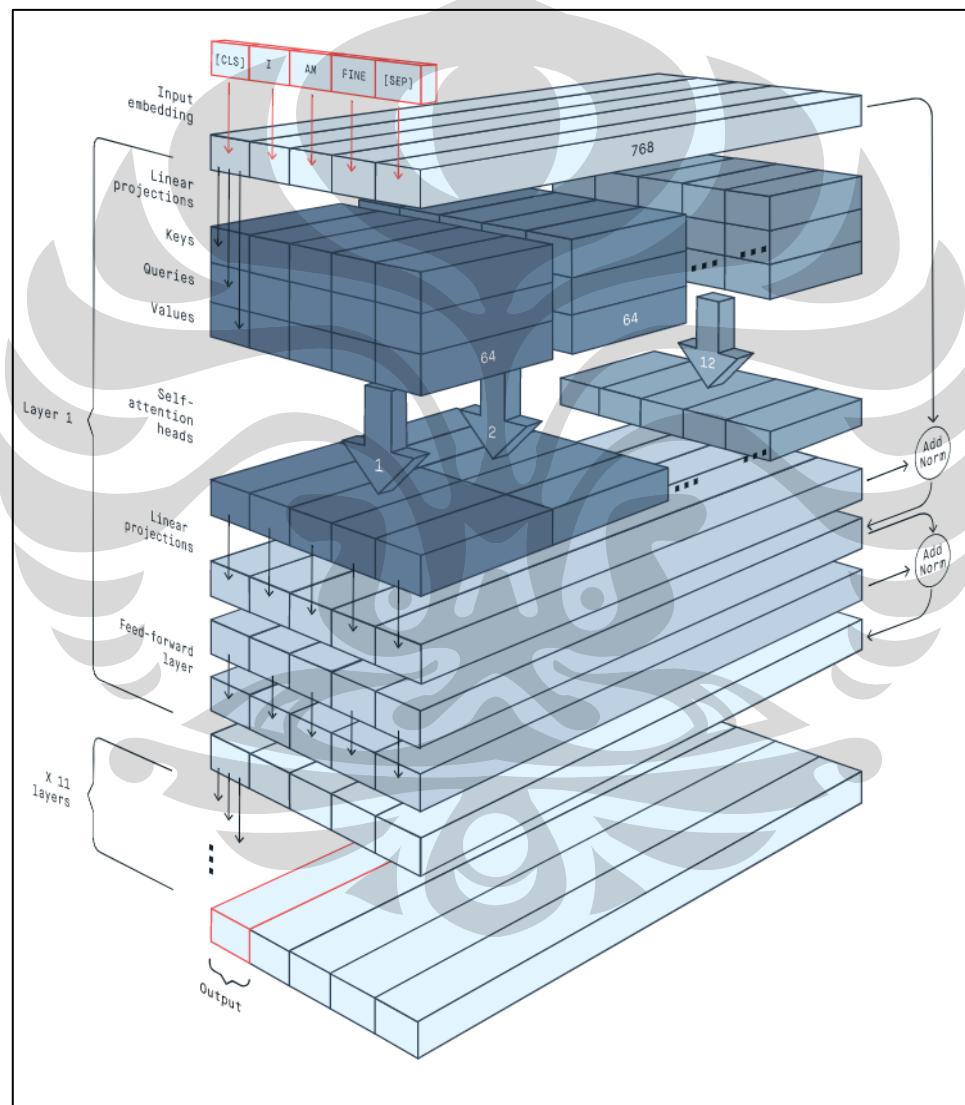
Struktur *encoder* dari arsitektur *transformer* akan menjadi penyusun utama dari model *bidirectional encoder representation from transformer* yang dijelaskan pada Subbab 3.2.

3.2. Bidirectional Encoder Representation from Transformer

Model *Bidirectional Encoder Representation from Transformers* (BERT) adalah *pre-trained language model* yang dikembangkan oleh Devlin *et al.* (2018) sebagai salah satu upaya untuk meningkatkan kualitas dan efisiensi solusi masalah *Natural Language*

Processing (NLP). Semua penjelasan dari subbab ini merujuk pada penelitian Devlin *et al.* (2018) kecuali disebutkan yang lain. Ilustrasi arsitektur model BERT dapat dilihat pada Gambar 3.8.

Arsitektur utama dari model BERT adalah lapisan-lapisan *transformer encoder* yang dijelaskan pada Subbab 3.1. Model BERT tersusun atas 12 lapisan *transformer encoder* seperti yang ditunjukkan pada Gambar 3.5. Setiap lapisan *transformer encoder* pada model BERT memiliki *hidden size* sebesar 768 dan nilai h pada lapisan *multi-head self-attention* sebesar 12.



Gambar 3.8. Ilustrasi model BERT

Sumber: (Peltarion, 2021) telah diolah kembali

Pada bagian ini diberikan penjelasan mengenai struktur input dari model BERT, dilanjutkan dengan penjelasan *pre-training* yang dilakukan pada model BERT,

dilanjutkan dengan penjelasan mengenai pendekatan berbasis fitur dari model BERT, dan diakhiri dengan penjelasan beberapa metode ekstraksi dan normalisasi yang digunakan pada representasi teks yang dihasilkan model BERT.

3.2.2. Representasi Input

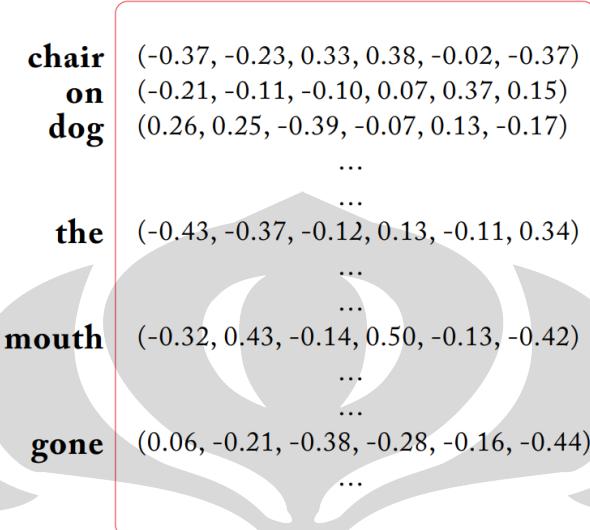
Sebelum menjelaskan representasi input yang digunakan model BERT, akan dijelaskan terlebih dahulu mengenai model WordPiece. Model WordPiece adalah model yang berfungsi untuk melakukan tokenisasi terhadap suatu dokumen (Wu *et al.*, 2016). Model WordPiece mampu menangani permasalahan *out-of-vocabulary*, yaitu permasalahan yang muncul ketika suatu kata tidak ada di dalam korpus yang sudah dilatih. Sehingga, *embedding* tetap dapat dilakukan untuk kata-kata langka yang tidak ada di dalam *vocabulary* (d'Sa, Illina, & Fohr., 2020).

Dalam melakukan tokenisasi, kata-kata ditokenisasi berdasarkan model WordPiece. Apabila ditemukan suatu kata yang tidak ada di dalam korpus, kata tersebut dipecah lagi menjadi subkata yang terkandung dalam korpus (Wu *et al.*, 2016). Sebagai ilustrasi, misalkan kata *embeddings* adalah kata yang tidak terkandung dalam *vocabulary* model WordPiece. Model WordPiece melakukan tokenasi dengan cara menyegmentasi kata *embeddings* menjadi subkata, sehingga diperoleh 4 buah *token* yaitu [em, ##bed, ##ding, ##s]. Penambahan simbol khusus ## pada bagian depan suatu *token* bertujuan untuk menginformasikan model bahwa tersebut adalah hasil segmentasi dari suatu kata.

Untuk suatu dokumen teks pada data, tokenisasi dilakukan terlebih dahulu dengan menggunakan model WordPiece sehingga diperoleh suatu *token sequence*. Sebagai ilustrasi, misalkan terdapat dokumen teks yaitu “*my dog is cute he likes playing.*” Proses tokenisasi dengan model WordPiece menghasilkan suatu *token sequence* yaitu [*my, dog, is, cute, he, likes, play, ##ing*].

Dua *token* khusus, [CLS] dan [SEP], diberikan pada dua posisi berbeda. *Token* [CLS] diberikan pada awal *token sequence*, sementara *token* [SEP] diberikan pada akhir atau pertengahan *token sequence*. Sebagai ilustrasi, misalkan terdapat *token sequence* [*my, dog, is, cute, he, likes, play, ##ing*]. *Token* khusus [CLS] ditambahkan di awal *token sequence* dan [SEP] ditambahkan di pertengahan dan akhir *token sequence* sehingga diperoleh [[CLS], *my, dog, is, cute, [SEP], he, likes, play, ##ing, [SEP]*].

Selanjutnya, tahapan *embedding* dilakukan pada setiap *token sequence*. Tahap *embedding* bertujuan untuk memetakan setiap *token* pada *token sequence* menuju vektor numerik dengan dimensi tertentu. Sebagai ilustrasi, Gambar 3.9 menunjukkan pemetaan *token* menuju vektor numerik berdimensi 6:



Gambar 3.9. Pemetaan *word embedding*

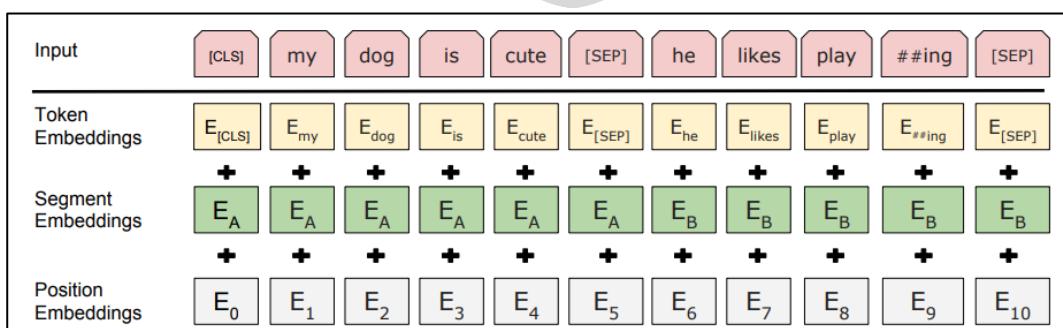
Sumber: (Goldberg, 2017)

Misalkan terdapat *token sequence* *[dog, on, chair]*. Setiap *token* dipetakan sesuai dengan Gambar 3.9 sehingga diperoleh vektor numerik untuk *token sequence* *[dog, on, chair]* yaitu:

$$[[-0.37, -0.23, 0.33, 0.38, -0.02, -0.37], [-0.21, -0.11, -0.10, 0.07, 0.37, 0.15], [-0.37, -0.23, 0.33, 0.38, -0.02, -0.37]]$$

Dimensi vektor numerik pada penelitian ini adalah 768.

Setiap *token sequence* memiliki 3 *embedding*, yaitu *token embedding*, *segment embedding*, dan *position embedding*. Perhitungan input dari model BERT dengan menggunakan ketiga *embedding* dapat diilustrasikan pada Gambar 3.10:



Gambar 3.10. Representasi input model BERT

Sumber: (Devlin *et al.*, 2018)

Parameter E_i adalah suatu vektor numerik i , dengan $E_i \neq E_j$ untuk $i \neq j$.

Token embedding bertujuan untuk mentransformasikan *token* menjadi vektor representasi sedemikian sehingga jarak antara setiap vektor representasi dapat menunjukkan similaritas antara setiap *token*. Misalkan terdapat dua buah *token* i dan j . Jika *token embedding* dari *token* i dan j berjarak relatif dekat, maka *token* i dan j memiliki tingkat similaritas yang relatif tinggi, dan begitu juga sebaliknya. Sebagai ilustrasi, misalkan terdapat *token sequence* sebagai berikut:

$$[[CLS], my, dog, is, cute, [SEP], he, likes, play, ##ing, [SEP]]$$

Seperti yang ditunjukkan pada barisan kotak berwarna kuning di Gambar 3.10, dua *token* $[SEP]$ pada *token sequence* dipetakan menuju vektor numerik yang sama yaitu $E_{[SEP]}$, dan *token* lainnya dipetakan menuju vektor numerik yang berbeda-beda.

Segment embedding bertujuan untuk membedakan pasangan kalimat dalam dokumen. Dalam permasalahan seperti *question answering*, suatu *token sequence* perlu dipisahkan menjadi dua bagian, yaitu *sequence A* dan *B*. Pemberian *segment embedding* memberikan karakteristik tambahan apakah suatu *token* berasal dari *sequence A* atau *B*. *Token* khusus $[SEP]$ yang dijelaskan sebelumnya berguna sebagai pemisah antara *sequence A* dengan *B*. Apabila pemisahan *token sequence* tidak dibutuhkan, *token* khusus $[SEP]$ diletakkan di akhir *token sequence*. Setiap *token* pada *sequence A* memiliki *segment embedding* yang sama, demikian juga untuk setiap *token* pada *sequence B*. Sebagai ilustrasi, Gambar 3.11 menunjukkan pemetaan *token* pada *sequence A* dan *B* menuju vektor numerik berdimensi 6:

Sequence A	(0.16, 0.03, -0.17, -0.13, 0.64, -0.12)
Sequence B	(0.41, 0.08, 0.44, 0.02, 0.46, -0.17)

Gambar 3.11. Pemetaan *segment embedding*

Misalkan terdapat *token sequence* $[[CLS], my, dog, is, cute, [SEP], he, likes, play, ##ing, [SEP]]$. *Token* khusus $[SEP]$ setelah *token* *cute* berperan sebagai pemisah. Sehingga diperoleh *sequence A* $[[CLS], my, dog, is, cute, [SEP]]$ dan *sequence B* $[he, likes, play, ##ing, [SEP]]$. Kemudian, *token* $[CLS]$, *my*, *dog*, *is*, *cute*, dan $[SEP]$ pertama memiliki nilai vektor numerik $[0.16, 0.03, -0.17, -0.13, 0.64, -0.12]$. Di sisi lain, *token* *he*, *likes*, *play*, $##ing$, dan $[SEP]$ kedua memiliki nilai vektor numerik $[0.41, 0.08, 0.44, 0.02, 0.46, -0.17]$.

Position embedding bertujuan memberikan *embedding* yang berisikan informasi mengenai posisi *token* dalam dokumen. Hal ini dikarenakan model BERT tidak memiliki bentuk konvolusi atau rekursi yang menyimpan informasi mengenai posisi *token*. *Token* dengan posisi yang sama pada dokumen berbeda memiliki *embedding* yang sama. Sebagai ilustrasi, Gambar 3.12 menunjukkan pemetaan *token* pada setiap posisi menuju vektor numerik berdimensi 6:

Posisi 1	(-0.04, 0.50, 0.04, 0.44, -0.19, 0.58)
Posisi 2	(-0.01, -0.35, -0.27, 0.20, 0.79, 0.02)
Posisi 3	(0.26, 0.28, -0.34, -0.02, -0.53, 0.46)
Posisi 4	(0.02, -0.47, 0.32, -0.49, 0.01, 0.12)
Posisi 5	(0.76, 0.21, -0.99, -0.15, 0.06, -0.84)
:	:

Gambar 3.12. Pemetaan *position embedding*

Misalkan terdapat dua buah *token sequence*, yaitu $[[CLS], my, dog, is, cute]$ dan $[[CLS], he, likes, play, \#\#ing]$. *Token* khusus [CLS] pada kedua *token sequence* memiliki vektor numerik yang sama dari pemetaan *position embedding*, yaitu $[-0.04, 0.50, 0.04, 0.44, -0.19, 0.58]$. Hal ini dikarenakan keduanya berada pada posisi pertama. Demikian pula untuk *token* *my* dan *he* yang berada pada posisi kedua dari masing-masing *token sequence*, vektor numerik dari *position embedding* untuk *token* *my* dan *he* adalah $[-0.01, -0.35, -0.27, 0.20, 0.79, 0.02]$, dan begitu seterusnya untuk *token* pada posisi lain.

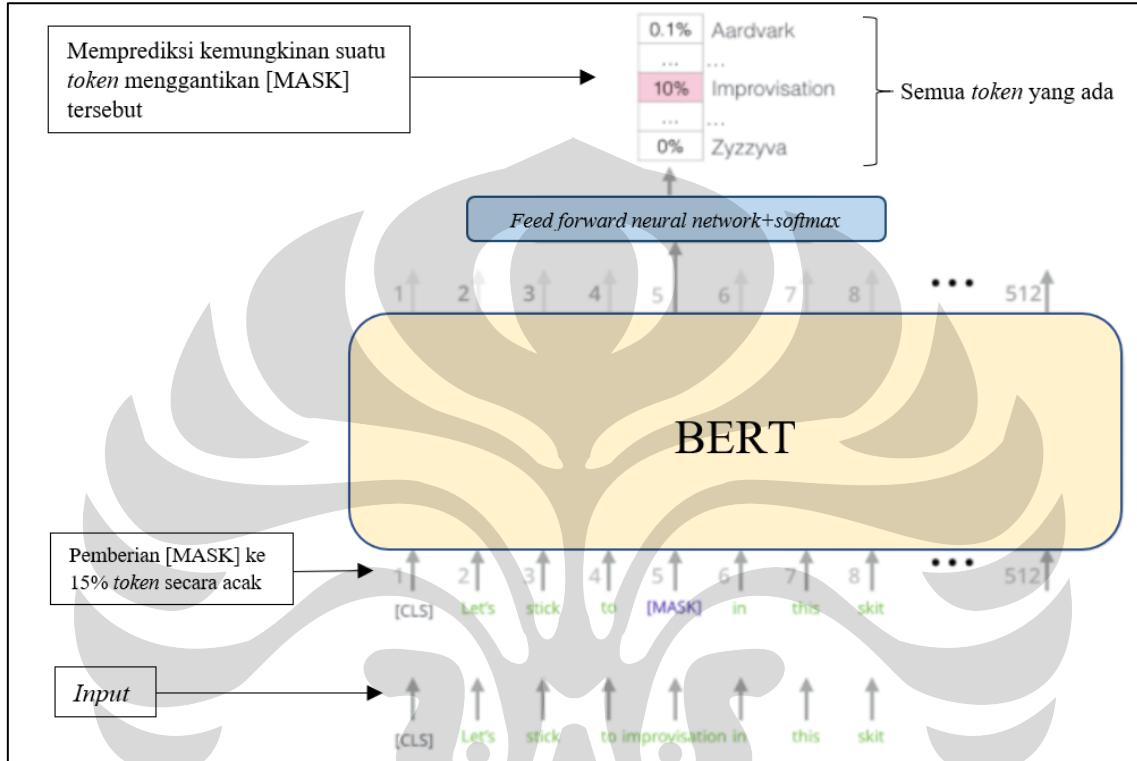
Terakhir, seperti yang diilustrasikan pada Gambar 3.10, setelah diperoleh vektor numerik dari *token embedding*, *segment embedding*, dan *position embedding*, ketiga vektor tersebut dijumlahkan sehingga didapatkan input bagi model BERT.

3.2.3. Pre-training BERT

Dalam penelitian ini, model BERT yang digunakan adalah model yang sudah melalui proses *pre-training* pada penelitian Devlin *et al.* (2018). Tahapan *pre-training* menggunakan 2 metode *unsupervised learning*, yaitu *masked language modelling* (MLM) yang dijelaskan pada Subbab 3.2.3.1 dan *next sentence prediction* (NSP) yang dijelaskan pada Subbab 3.2.3.2.

3.2.3.1. Masked Language Modelling

Model *deep bidirectional* dapat memahami pola dan konteks *sequence* dari kedua arah secara simultan. Dalam rangka melatih suatu model supaya memiliki *deep bidirectional*, dilakukan pembelajaran *masked language modelling* (MLM). Alur proses pembelajaran MLM diilustrasikan pada Gambar 3.13:



Gambar 3.13. Alur pembelajaran *masked language modelling*

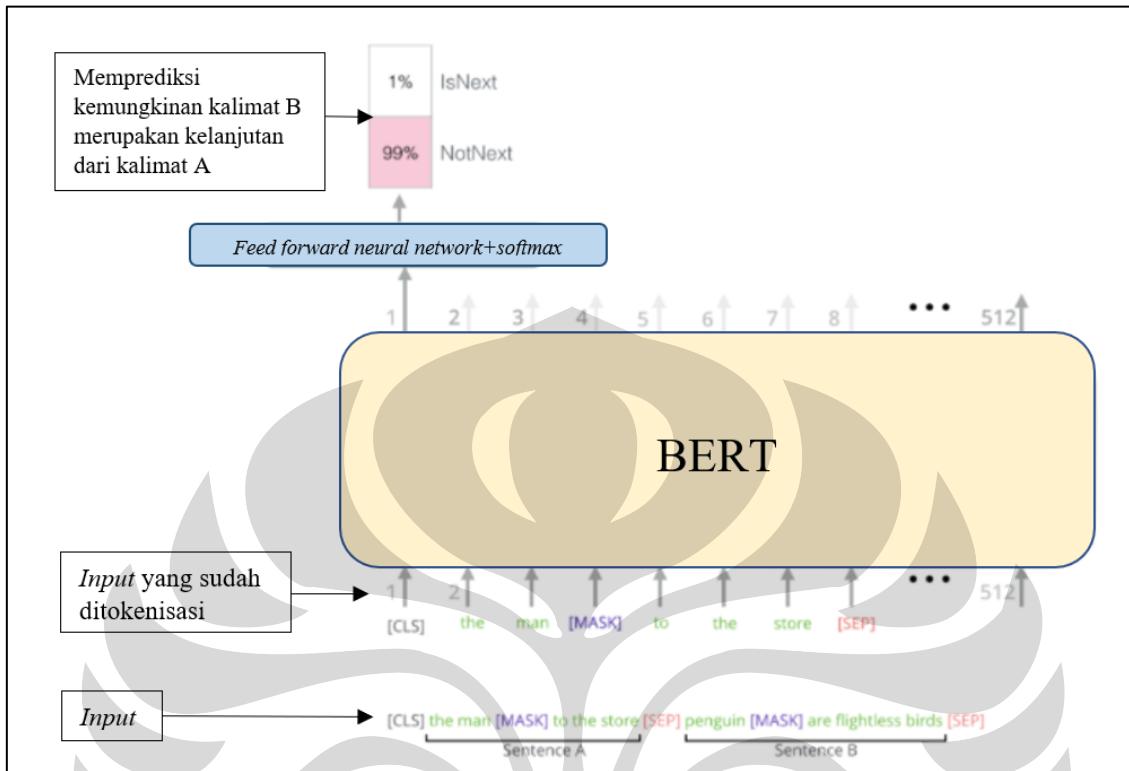
Sumber: (Alammar, 2018) telah diolah kembali

Sebelum proses pembelajaran MLM dipilih sebanyak 15% dari semua *token* untuk diprediksi. Sebanyak 12% dari semua *token* pada masing-masing *sequence* diberikan *mask* secara acak menggunakan *token* khusus [MASK], sementara 1.5% lainnya diganti dengan *token* lain secara acak, dan 1.5% yang tersisa tidak diganti. Pembelajaran MLM dilakukan dengan memprediksi *token* yang diberikan *mask* dan *token* yang diubah menjadi *token* lain.

3.2.3.2. Next Sentence Prediction

Permasalahan NLP seperti *Question Answering* (QA) dan *Natural Language Inference* (NLI) bergantung pada pembelajaran hubungan dua buah kalimat. Sehingga untuk melatih suatu model yang dapat memahami hubungan dua buah kalimat, dilakukan

pre-training terhadap permasalahan *next sentence prediction* (NSP). Alur proses pembelajaran NSP diilustrasikan pada Gambar 3.14:



Gambar 3.14. Alur pembelajaran *next sentence prediction*

Sumber: (Alammar, 2018) telah diolah kembali

Misalkan terdapat suatu kalimat berpasangan, A dan B. Permasalahan NSP memiliki tujuan untuk mengklasifikasi apakah kalimat B merupakan kelanjutan dari kalimat A. Masalah NSP memiliki relasi erat dengan *representation learning* (Jernite, Bowman, & Sontag, 2017; Logeswaran & Lee, 2018). Namun, hasil akhir dari penelitian sebelumnya berupa suatu *sentence embedding* bernilai tetap. Di sisi lain, hasil akhir pada model BERT adalah suatu model dengan parameter yang sudah dipelajari yang kemudian dapat dilakukan *fine tuning* sesuai dengan permasalahan NLP yang ingin diselesaikan.

3.2.3.3. Data *Pre-training*

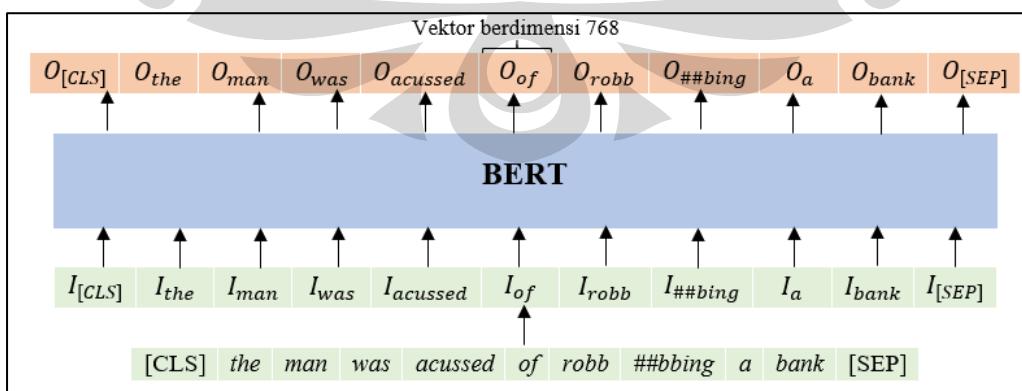
Untuk tahapan ini, korpus yang digunakan adalah BookCorpus (800 juta kata) (Zhu *et al.* 2015) dan Wikipedia Bahasa Inggris (2,5 miliar kata). Ekstraksi yang diterapkan pada Wikipedia mengambil paragraf teks dan mengabaikan teks dalam bentuk daftar, tabel, dan *header*. Korpus tingkat dokumen seperti Wikipedia Bahasa Inggris dan BookCorpus memiliki kelebihan yaitu dapat mengekstraksi *sequence* yang panjang.

3.3. Bidirectional Encoder Representation from Transformer Sebagai Metode Representasi Teks

Pada Subbab 3.2 telah dijelaskan mengenai model BERT secara umum. Pada bagian ini diberikan penjelasan mengenai pendekatan berbasis fitur dari model BERT, yang merupakan implementasi model BERT untuk memperoleh representasi data teks. Selain itu dijelaskan juga metode ekstraksi fitur serta normalisasi fitur yang diterapkan pada *output* dari pendekatan berbasis fitur dengan model BERT.

3.3.1. Pendekatan Berbasis Fitur dengan BERT

Pendekatan berbasis fitur dengan BERT melakukan ekstraksi terhadap fitur tetap dari model yang sudah dilakukan *pre-training*. Pendekatan ini juga dikenal dengan metode *contextualized word embedding*. Pada *contextualized word embedding*, setiap kata dipetakan menuju ruang vektor dan kata yang memiliki makna yang relatif similar berjarak relatif dekat pada ruang vektor tersebut (Kaliyar, 2020). Pendekatan ini memiliki 2 keunggulan apabila dibandingkan dengan melakukan *fine tuning* langsung pada model BERT. Keunggulan pertama adalah dapat menambahkan arsitektur berbeda yang spesifik terhadap suatu permasalahan. Hal ini dikarenakan tidak semua permasalahan NLP dapat diselesaikan dengan menggunakan arsitektur *transformer encoder*. Kedua, efisiensi komputasi meningkat. Hal ini dikarenakan *pre-computation* representasi yang mahal secara komputasi cukup dilakukan sekali kemudian dapat digunakan pada berbagai eksperimen.



Gambar 3.15. Ilustrasi metode representasi teks dengan model BERT

Pendekatan berbasis fitur dengan BERT sebagai metode representasi teks diilustrasikan pada Gambar 3.15. Parameter I_x adalah vektor representasi input

berdimensi 768 dari *token* x yang dijelaskan pada Subbab 3.2.2. Parameter \mathbf{O}_x adalah vektor *output* berdimensi 768 yang berkorespondensi dengan token x . BERT pada Gambar 3.15 adalah model BERT dengan arsitektur *transformer encoder* seperti yang dijelaskan pada Subbab 3.2 dan juga merupakan model BERT yang sudah melalui proses *pre-training* sesuai dengan Subbab 3.2.3. Proses pengambilan representasi teks dengan model BERT dilakukan dengan cara melakukan *feed forward* ke dalam model BERT. Sehingga untuk suatu dokumen yang mengandung n *token*, representasi teks yang diperoleh adalah n vektor numerik berdimensi 768. vektor *output* dari seluruh kata pada dokumen dapat disusun menjadi suatu matriks $O = [\mathbf{O}_1 \mathbf{O}_2 \dots \mathbf{O}_n]^T$ berukuran $n \times 768$. Dikarenakan representasi yang dihasilkan masih berupa matriks berukuran $n \times 768$, dilakukan ekstraksi fitur yang mereduksi representasi menjadi sebuah vektor berdimensi 768. Subbab 3.2.2 memberikan penjelasan mengenai beberapa jenis metode ekstraksi fitur yang digunakan pada penelitian ini.

3.3.2. Ekstraksi Fitur

Strategi ekstraksi fitur dibutuhkan untuk mengubah representasi data teks model BERT yang berdimensi tinggi menjadi vektor fitur berukuran tetap dengan dimensi lebih rendah. Penelitian ini mengimplementasikan dua strategi ekstraksi fitur yaitu *max pooling* dan *mean pooling*. *Max pooling* mengasumsikan bahwa nilai tertinggi mengandung fitur yang paling penting. Misalkan dalam suatu dokumen terdapat n *token* dan untuk *token* ke- i memiliki representasi berupa vektor \mathbf{h}_i berdimensi d $[h_{i1}, h_{i2}, \dots, h_{id}]$, strategi *max pooling* dapat direpresentasikan dalam fungsi pada persamaan (3.18) (Guan *et al.*, 2020):

$$\mathbf{h}[k] = \max_{i=1 \dots n} h_{ik}, \quad (3.18)$$

dengan h_{ik} adalah elemen ke- k dari vektor representasi dari *token* ke- i dan $\mathbf{h}[k]$ adalah entri ke- k dari vektor representasi hasil ekstraksi fitur, \mathbf{h} . Sebagai ilustrasi, misalkan terdapat suatu dokumen dengan 4 *token* dan token ke- i memiliki vektor representasi numerik \mathbf{h}_i berdimensi 6 seperti yang ditunjukkan pada (3.19):

$$\begin{aligned} \mathbf{h}_1 &= [0.28, 0.94, 0.03, 0.54, 0.08, 0.39] \\ \mathbf{h}_2 &= [0.68, 0.26, 0.80, 0.12, 0.62, 0.82] \\ \mathbf{h}_3 &= [0.92, 0.07, 0.04, 0.68, 0.87, 0.61] \\ \mathbf{h}_4 &= [0.06, 0.73, 0.47, 0.18, 0.59, 0.97] \end{aligned} \quad (3.19)$$

Penentuan nilai $\mathbf{h}[k]$ untuk setiap $k = 1, 2, \dots, 6$ mengikuti persamaan (3.18) sehingga diperoleh:

$$\mathbf{h}[1] = \max(0.28, 0.68, 0.92, 0.06) = 0.92$$

$$\mathbf{h}[2] = \max(0.94, 0.26, 0.07, 0.73) = 0.94$$

$$\mathbf{h}[3] = \max(0.03, 0.80, 0.04, 0.47) = 0.80$$

$$\mathbf{h}[4] = \max(0.54, 0.12, 0.68, 0.18) = 0.68$$

$$\mathbf{h}[5] = \max(0.08, 0.63, 0.87, 0.59) = 0.87$$

$$\mathbf{h}[6] = \max(0.39, 0.82, 0.61, 0.97) = 0.97$$

Sehingga diperoleh vektor representasi akhir yang sudah direduksi adalah $\mathbf{h} = [0.92, 0.94, 0.80, 0.68, 0.87, 0.97]$.

Mean pooling memiliki landasan bahwa semua vektor fitur kontekstual dapat merepresentasikan teks keseluruhan, dan dengan mengambil nilai rata-rata dari vektor ini maka *noise* dapat lebih tereduksi. Misalkan terdapat n *token* dan untuk *token* ke- i memiliki representasi berupa vektor \mathbf{h}_i berdimensi d $[h_{i1}, h_{i2}, \dots, h_{id}]$ dan $\mathbf{h}[k]$ adalah entri ke- k dari vektor representasi hasil ekstraksi fitur, \mathbf{h} . strategi *Mean pooling* dapat diekspresikan dalam fungsi pada persamaan (3.20) (Guan *et al.*, 2020)

$$\mathbf{h}[k] = \frac{\sum_{i=1}^n h_{ik}}{n}. \quad (3.20)$$

Sebagai ilustrasi, misalkan terdapat suatu dokumen dengan 4 *token* dan token ke- i memiliki vektor representasi numerik \mathbf{h}_i berdimensi 6 seperti yang ditunjukkan pada (3.19). Penentuan nilai $\mathbf{h}[k]$ untuk setiap $k = 1, 2, \dots, 6$ mengikuti persamaan (3.20) sehingga diperoleh:

$$\mathbf{h}[1] = \frac{0.28 + 0.68 + 0.92 + 0.06}{4} = 0.485$$

$$\mathbf{h}[2] = \frac{0.94 + 0.26 + 0.07 + 0.73}{4} = 0.5$$

$$\mathbf{h}[3] = \frac{0.03, 0.80, 0.04 + 0.47}{4} = 0.335$$

$$\mathbf{h}[4] = \frac{0.54 + 0.12 + 0.68 + 0.18}{4} = 0.38$$

$$\mathbf{h}[5] = \frac{0.08 + 0.63 + 0.87 + 0.59}{4} = 0.5425$$

$$\mathbf{h}[6] = \frac{0.39 + 0.82 + 0.61 + 0.97}{4} = 0.6975$$

Sehingga diperoleh vektor representasi akhir yang sudah direduksi adalah $\mathbf{h} = [0.485, 0.5, 0.335, 0.38, 0.5425, 0.6975]$.

Output berupa vektor representasi yang diperoleh dari metode ekstraksi fitur akan menjadi input untuk tahapan normalisasi fitur. Subbab 3.3.3 memberikan penjelasan mengenai berbagai jenis metode normalisasi fitur yang digunakan pada penelitian ini.

3.3.3. Normalisasi Fitur

Normalisasi atau transformasi fitur perlu dilakukan untuk memastikan bahwa vektor representasi berukuran tetap pada setiap dokumen memiliki karakteristik normalitas atau stabilitas. Dalam penelitian ini, diimplementasikan 4 strategi normalisasi yang berbeda, yaitu *identity normalization*, *standard normalization*, *layer normalization*, dan *min-max normalization*

Identity normalization adalah fungsi identitas $f(\mathbf{h}) = \mathbf{h}$ dan digunakan sebagai *baseline* untuk metode normalisasi yang lain. Untuk suatu vektor fitur \mathbf{h}_i , *standard normalization* menerapkan fungsi pada persamaan (3.21) (Guan *et al.*, 2020)

$$\bar{\mathbf{h}}_i = \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|}, \quad (3.21)$$

dengan $\bar{\mathbf{h}}_i$ adalah hasil vektor yang sudah dinormalisasi, strategi ini melakukan transformasi menjadi vektor dengan norm sebesar 1. Jarak Euclidean antar dua vektor fitur akan sama dengan jarak *cosine* di antara dua vektor tersebut. Sebagai ilustrasi, misalkan terdapat suatu vektor fitur \mathbf{h} seperti yang ditunjukkan pada (3.22):

$$\mathbf{h} = [0.92, 0.94, 0.80, 0.68, 0.87, 0.97]. \quad (3.22)$$

Proses *standard normalization* pada vektor \mathbf{h} dengan menggunakan persamaan (3.21) diawali dengan melakukan perhitungan norm $\|\mathbf{h}\|$ sebagai berikut:

$$\|\mathbf{h}\| = \sqrt{0.92^2 + 0.94^2 + 0.80^2 + 0.68^2 + 0.87^2 + 0.97^2} = 2.1284.$$

Kemudian dengan menggunakan persamaan (3.21), diperoleh vektor yang dinormalisasi dengan *standard normalization* sebagai berikut:

$$\begin{aligned} \bar{\mathbf{h}} &= \frac{[0.92, 0.94, 0.80, 0.68, 0.87, 0.97]}{2.1284} \\ &= [0.432, 0.442, 0.376, 0.319, 0.409, 0.456] \end{aligned}$$

Strategi *layer normalization* diimplementasikan untuk menghindari adanya masalah pergeseran kovariat dalam proses pelatihan *neural network* (Ba, Kiros, & Hinton,

2016). Untuk suatu vektor fitur \mathbf{h}_i , *layer normalization* menerapkan fungsi pada persamaan (3.23):

$$\bar{\mathbf{h}}_i = \frac{\mathbf{h}_i - \phi_i}{\sigma_i}, \quad (3.23)$$

dengan ϕ_i dan σ_i berturut-turut adalah rata-rata dan standar deviasi dari vektor fitur \mathbf{h}_i . Sebagai ilustrasi, misalkan terdapat suatu vektor fitur \mathbf{h} seperti yang ditunjukkan pada (3.22). Proses *layer normalization* pada vektor \mathbf{h} diawali dengan menentukan nilai rata-rata dari vektor fitur \mathbf{h} :

$$\phi = \frac{0.92 + 0.94 + 0.80 + 0.68 + 0.87 + 0.97}{6} = 0.8633,$$

dan nilai standar deviasi dari vektor fitur \mathbf{h} :

$$\sigma = \sqrt{\frac{(0.92 - 0.8633)^2 + (0.94 - 0.8633)^2 + (0.8 - 0.8633)^2 + (0.68 - 0.8633)^2 + (0.87 - 0.8633)^2 + (0.97 - 0.8633)^2}{6}} = 0.09843.$$

Sehingga dengan menggunakan persamaan (3.23), diperoleh vektor yang dinormalisasi dengan *layer normalization* sebagai berikut:

$$\begin{aligned} \bar{\mathbf{h}} &= \frac{[0.92, 0.94, 0.80, 0.68, 0.87, 0.97] - [0.8633, 0.8633, 0.8633, 0.8633, 0.8633, 0.8633]}{0.09843} \\ &= \frac{[0.0567, 0.0767, -0.0633, -0.1833, 0.0067, 0.1067]}{0.09843} \\ &= [0.5760, 0.7792, -0.6431, -1.8622, 0.0681, 1.084] \end{aligned}$$

Strategi *min-max normalization* adalah strategi normalisasi yang tetap melestarikan distribusi awal dari vektor fitur. Untuk suatu vektor fitur \mathbf{h}_i berdimensi d $\{h_{i1}, h_{i2}, \dots, h_{id}\}$, *min-max normalization* menerapkan fungsi pada persamaan (3.24):

$$\bar{\mathbf{h}}_i = \frac{\mathbf{h}_i - \min_d(h_{id})}{\max_d(h_{id}) - \min_d(h_{id})}. \quad (3.24)$$

Min-max normalization melakukan transformasi dengan cara melakukan penskalaan terhadap vektor fitur \mathbf{h}_i yang awalnya memiliki interval nilai $[\min_d(h_{id}), \max_d(h_{id})]$ menjadi $[0,1]$ (Han, Pei, & Kamber, 2011). Sebagai ilustrasi, misalkan terdapat suatu vektor fitur \mathbf{h} seperti yang ditunjukkan pada (3.22). Proses *standard normalization* pada vektor \mathbf{h} diawali dengan menentukan nilai entri maksimum dan minimum dari vektor fitur \mathbf{h} :

$$\min_{d=1,\dots,6} (h_d) = \min(0.92, 0.94, 0.80, 0.68, 0.87, 0.97) = 0.68,$$

$$\max_{d=1,\dots,6} (h_d) = \max(0.92, 0.94, 0.80, 0.68, 0.87, 0.97) = 0.97,$$

kemudian dilanjutkan dengan penentuan selisih nilai entri maksimum dan minimum dari vektor fitur \mathbf{h} :

$$\max_{d=1,\dots,6} (h_d) - \min_{d=1,\dots,6} (h_d) = 0.97 - 0.68 = 0.29.$$

Sehingga dengan menggunakan persamaan (3.24), diperoleh vektor yang dinormalisasi dengan *min-max normalization* sebagai berikut:

$$\begin{aligned}\bar{\mathbf{h}} &= \frac{[0.92, 0.94, 0.80, 0.68, 0.87, 0.97] - [0.68, 0.68, 0.68, 0.68, 0.68, 0.68]}{0.29} \\ &= \frac{[0.24, 0.26, 0.12, 0, 0.19, 0.29]}{0.29} \\ &= [0.8276, 0.8966, 0.4138, 0, 0.6552, 1]\end{aligned}$$

Sehingga setelah melalui proses *min-max normalization* diperoleh vektor numerik berdimensi 6.

BAB 4

HASIL SIMULASI DAN PEMBAHASAN

Bab ini membahas hasil simulasi dari *Bidirectional Encoder Representation from Transformers* (BERT) sebagai metode representasi teks dalam *text clustering*. Algoritma *text clustering* yang digunakan dalam simulasi ini adalah *k-means clustering* (KM), *eigenspace-based fuzzy c-means* (EFCM), *deep embedded clustering* (DEC), dan *improved deep embedded clustering* (IDEC).

4.1. Sepsifikasi Mesin dan Perangkat Lunak

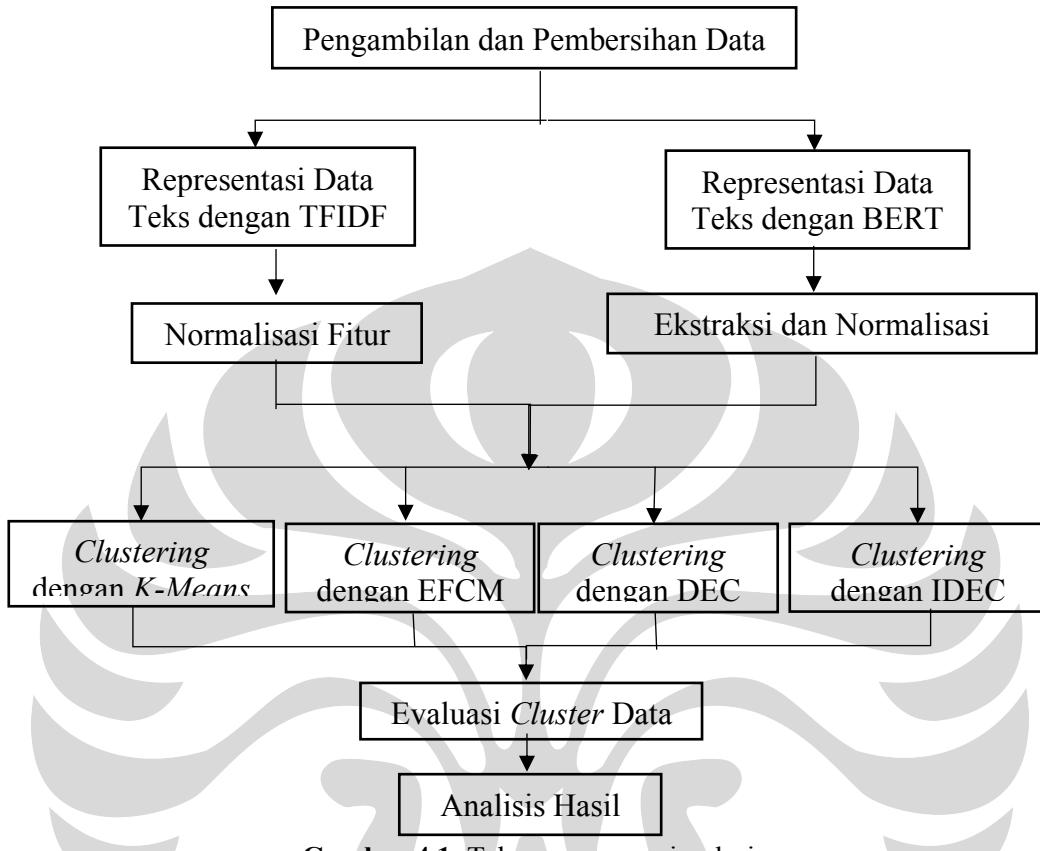
Spesifikasi mesin dan perangkat lunak yang digunakan pada tahapan simulasi dapat dilihat secara lengkap pada Tabel 4.1

Tabel 4.1. Spesifikasi mesin dan perangkat lunak

Mesin	
<i>Central Processing Unit</i> (CPU)	Intel® CoreTM i7-8850H @2.60GHz
<i>Memory</i> (RAM)	16GB
<i>Graphic Processing Unit</i> (GPU)	Nvidia® Quadro® P600
Sistem Operasi	Windows 10, Pro 64-bit
Perangkat Lunak	
Perangkat Lunak Pemrograman	Jupyter Notebook 6.0.2 Google Colaboratory
Bahasa Pemrograman	Python 3.7.5
Pustaka yang Digunakan	1. pandas 0.25.3 2. numpy 1.17.3 3. sklearn 0.21.3 4. coclust 0.2.1 5. skfuzzy 0.4.2 6. nltk 3.2.5 7. tensorflow 1.15.0 & 2.0.0 8. keras 2.3.1 9. bert-serving-client 1.10.0 10. bert-serving-server 1.10.0

4.2. Tahapan Umum Simulasi

Secara umum, tahapan simulasi pada penelitian ini diilustrasikan pada Gambar 4.1.



Gambar 4.1. Tahapan umum simulasi

Simulasi diawali dengan pengambilan data. Data yang digunakan adalah data pada penelitian Guan *et al.* (2020) yang kemudian dibersihkan dengan merujuk pada penelitian yang sama. Data kemudian diproses lebih lanjut dengan mengekstrasi representasi data teks. Dua metode pengambilan representasi teks yang digunakan adalah *term frequency-inverse document frequency* (TFIDF) dan model BERT. Tokenisasi yang sesuai dengan masing-masing metode representasi data teks dilakukan terlebih dahulu. Setelah dilakukan tokenisasi, data ditransformasi menggunakan TFIDF dan BERT. Representasi dari TFIDF dinormalisasi untuk menjadi input pada beberapa metode *text clustering*, yaitu DEC dan IDEC. Hasil representasi data teks yang diperoleh dari BERT juga dilakukan beberapa metode ekstraksi dan normalisasi yang berbeda.

Setelah diperoleh representasi data teks yang konstan dari kedua metode, representasi tersebut dijadikan input pada 4 model *text clustering*, yaitu KM, EFCM, DEC, dan IDEC. Evaluasi *cluster* yang dihasilkan dilakukan dengan membandingkan kelompok

yang dihasilkan dengan *ground-truth label* data. Performa model dievaluasi menggunakan *clustering accuracy* (ACC), *normalized mutual information* (NMI), dan *adjusted rand index* (ARI). Kinerja model BERT sebagai metode representasi data teks dianalisis dengan memperhatikan metrik yang digunakan. Selanjutnya, kinerja antara representasi dari TFIDF dengan representasi dari model BERT dalam *text clustering* juga dibandingkan. Terakhir, kinerja antara representasi data teks dari model BERT dengan metode ekstraksi dan normalisasi fitur yang berbeda dibandingkan.

4.3. Data

Pada bagian dilakukan pembahasan tahapan awal dari simulasi yang dijelaskan pada bagian 4.2, yaitu pengambilan datayang dilanjutkan dengan pra-pengolahan data.

4.3.1. Pengambilan Data

Tiga data yang digunakan adalah *AG news*, *Yahoo! Answers*, dan *R2*. Penjelasan mengenai ketiga data tersebut dikutip dari penelitian Guan *et al.* (2020) kecuali disebutkan yang lain. Masing-masing kelas pada data *AG news* dan *Yahoo! Answers* terdiri atas 1000 sampel. Hal ini dikarenakan data yang berjumlah sedikit namun seimbang dapat menghasilkan model yang memiliki performa serupa dengan apabila menggunakan data asli (Guan *et al.*, 2020). Pengambilan 1000 sampel untuk setiap kelas mengurangi beban komputasi secara signifikan tanpa terlalu mempengaruhi performa model. Deskripsi secara singkat dari data yang digunakan pada penelitian ini dapat dilihat pada Tabel 4.2, sementara distribusi kelas dari data *R2* dapat dilihat pada Tabel 4.3:

Tabel 4.2. Deskripsi singkat data

Data	Deskripsi	Banyak Kelas	Jumlah Keseluruhan Data
<i>AG news</i>	Memuat judul dan isi berita dari media AG News yang dikategorisasikan berdasarkan topik berita	4	4000
<i>Yahoo! Answers</i>	Memuat pertanyaan yang ditanyakan pada <i>Yahoo! Answers</i> berserta dengan jawabannya yang dikategorisasikan berdasarkan topik pertanyaan	10	10000
<i>R2</i>	Memuat dokumen yang diekstraksi dari Reuters-21578, yang merupakan	2	5859

	data berisikan dokumen berita dari media massa Reuters pada 1987		
--	--	--	--

Tabel 4.3. Distribusi kelas pada data *R2*

Kelas	Banyak Data
0	3734
1	2125

Pertama, data *AG news* adalah korpus berisikan teks berita yang sudah dikategorisasikan. Teks pada data *AG news* berisikan judul dan isi dari berita yang diambil. Data *AG News* terbagi 4 kategori, yaitu dunia (*world*), olahraga (*sports*), bisnis (*business*), dan sains dan teknologi (*sci/tech*). Selanjutnya, data *Yahoo! Answers* adalah korpus untuk permasalahan klasifikasi topik yang diekstraksi dari *Yahoo! Answers Comprehensive Questions and Answers version 1.0*. Data ini berisikan pertanyaan yang didampingi dengan jawaban yang bersesuaian. Data *Yahoo! Answers* terbagi atas 10 kategori, yaitu masyarakat dan budaya (*social&culture*), sains dan matematika (*science&mathematics*), kesehatan (*health*), pendidikan dan referensi (*education&reference*), olahraga (*sports*), bisnis dan finansial (*business&finance*), hiburan dan musik (*entertainment&music*), keluarga dan relasi (*family&relationships*), komputer dan internet (*computer&internet*), serta politik dan pemerintah (*politics&government*). Terakhir, data *R2* adalah data yang diekstraksi dari *Reuters-21578*. Data *R2* berisikan teks dokumen berita dari 2 kategori teratas pada *Reuters-21578*, yaitu *earn* dan *acq*. Tabel 4.4 sampai 4.6 secara berurutan menunjukkan beberapa contoh data teks yang terdapat dalam data *AG News*, *Yahoo! Answers*, dan *R2*.

Tabel 4.4. Contoh data teks *AG news*

Teks	Kelas
<i>CAIRO , EGYPT -- A man identified as Osama bin Laden , speaking on an audiotape posted on an Islamic website yesterday , praised the men who attacked a US consulate in Jidda , Saudi Arabia , earlier this month and called on militants to stop the flow of oil</i>	<i>world</i>
<i>China's fastest driver , Rubens Barrichello , would rather rent a bed on the main straight of the Formula One track than commute to a Shanghai hotel every day .</i>	<i>sports</i>
<i>The European Commission plans to announce a formal proposal settling the long-running Coca-Cola antitrust case later on Tuesday , a source close to the situation said .</i>	<i>business</i>

<i>Dr Jane Goodall of England , world-acclaimed conservation biologist and pioneer for her work on chimpanzees in Africa , delivers her keynote speech entitled quot ; Reason for Hope quot ; Tuesday , Dec. 7 , 2004 in Singapore</i>	<i>sci/tech</i>
--	-----------------

Tabel 4.5. Contoh data teks R2

Teks	Kelas
<i>Champion Products Inc said its board of directors approved a two-for-one stock split of its common shares for shareholders of record as of April 1 , 1987 .</i>	<i>earn</i>
<i>Computer Terminal Systems Inc said it has completed the sale of 200,000 shares of its common stock , and warrants to acquire an additional one mln shares , to < Sedio N.V. > of Lugano , Switzerland for 50,000 dlers .</i>	<i>acq</i>
<i>CoFAB Inc said it acquired < Gulfex Inc > , a Houston-based fabricator of custom high-pressure process vessels for the energy and petrochemical industries .</i>	<i>acq</i>
<i>Avery said its board authorized a two for one stock split , an increased in the quarterly dividend and plans to offer four mln shares of common stock .</i>	<i>earn</i>
<i>Sweden 's Wallenberg group fought back a bid by the London-based Swedish financier Erik Penser to secure a large stake in Swedish Match < SMBS ST > , one of the companies at the core of their business empire .</i>	<i>acq</i>

Tabel 4.6. Contoh data teks Yahoo! Answers

Teks	Kelas
<i>Why we Worry and always in Tension (Specially We Indians) if leave worrying then how will be life be . ? worries and tensions are part of life and do come at every stage , but to look up at life by a +ve way is quite lovely .</i>	<i>Social & culture</i>
<i>How does peripheral resistance affect cardiac output ? This is a review question for a pathophysiology class . If the peripheral resistance is decreased , as in case of vasodilation , heart tries to pump more blood to fulfill the requirement , thus increasing cardiac output .</i>	<i>Science & mathematics</i>
<i>girls , why breast hurt and getting bigger before period ? ! ? Increases in estrogen , the female sex hormone , cause it .</i>	<i>Health</i>
<i>why do bubbles appear when water boils ? The water from the bottom is being transformed from liquid to gas and the gas bubbles to the top . Kinda like when you 're in the bathtub and fart .</i>	<i>Education & reference</i>
<i>What is the point of Yahoo ! Answers ? It 's pretty ridiculous . All you have to do is Google , right ? This is what yahoo says about it . Yahoo ! Answers is an online community in which participants can ask and answer questions on a wide range of topics , from the serious to the irresistibly trivial .</i>	<i>Computer & internet</i>
<i>what are all the names of the coaches in the nfl ? www.nfl.com</i>	<i>Sports</i>
<i>In the US is it allowed to change your bank account number without closing the account and opening a new one ? If there is a way to do it ,</i>	<i>Business & finance</i>

<i>I'm sure it is easier to just open a new one , transfer all your money , close your old one . If your bank account number were to change you would need new cards , new checks , new routing numbers , etc... so I don't know what you would be trying to accomplish .</i>	
<i>what techno house songs do you like right now ? anything by DJ Venom</i>	<i>Entertainment & music</i>
<i>Whats good Music for someone who past away ? im making a montage for my dgo who passed away.I want music for it .what would you recommend ? Wind Beneath my Wings is played a lot . Also a lot of hymns or classical music without lyrics . Greensleeves and stuff like that .</i>	<i>Family & relationships</i>
<i>what is the total annual cost of living increase from 2004 to 2005 in the US ? That all depends on what index you use . From December 2004 to December 2005 , there was a 3.4 % increase in the consumer price index . Whether or not that accurately reflects cost of living , I leave to you to decide .</i>	<i>Politics & government</i>

4.3.2. Pra-Pengolahan Data

Pada subbab ini dijelaskan mengenai berbagai tahapan pra-pengolahan yang dilakukan pada data yang digunakan. Teks yang diambil sudah dilakukan proses pembersihan data. Proses ini bertujuan untuk memudahkan model dalam memahami representasi data teks yang dihasilkan dan meningkatkan performa model. Proses pembersihan yang dilakukan dapat dilihat sebagai berikut:

1. Setiap baris dalam satu dokumen digabungkan menjadi 1 baris
2. Tanda pagar yang diikuti angka dihapus
3. Teks dan kode yang merujuk pada bahasa pemrograman HTML atau XML dihapus
4. Spasi yang berlebihan dihapus
5. Tanda baca yang repetitif diganti menjadi satu tanda baca saja

Tabel 4.7 menunjukkan beberapa contoh perbedaan data sebelum dibersihkan dan sesudah dibersihkan.

Tabel 4.7. Contoh pembersihan data

Sebelum	Sesudah
<i>Iraq #39;s government continued efforts to end the chaos prevailing in the country.</i>	<i>Iraq s government continued efforts to end the chaos prevailing in the country.</i>
<i>500 grams = how many ounces???</i> <i>km = how many mi.?</i> <i>24 ounces = how many grams?</i>	<i>500 grams = how many ounces ? ? ?</i> <i>314 km = how many mi . ? 24</i> <i>ounces = how many grams ?</i>

Data yang telah melalui tahapan pra-pengolahan selanjutnya dilakukan pengambilan representasi data teks yang dijelaskan pada Subbab 4.4.

4.4. Representasi Data Teks

Pada subbab ini dijelaskan mengenai simulasi pengambilan representasi data teks yang diterapkan. Tahapan ini menggunakan dua metode, yaitu metode TFIDF dan metode BERT. Untuk masing-masing metode, diberikan penjelasan mengenai tokenisasi serta tahapan lainnya seperti *padding* dan *encoding* pada metode BERT, serta normalisasi pada kedua metode representasi.

4.4.1. Representasi Data Teks dengan Metode TFIDF

Sebelum melakukan proses pemgambilan representasi data teks dengan menggunakan metode TFIDF, tokenisasi data dilakukan terlebih dahulu. Tokenisasi untuk metode TFIDF dilakukan dengan bantuan *natural language toolkit* (NLTK) dimana setiap kata pada suatu kalimat harus dipisahkan terlebih dahulu. Tabel 4.8 menunjukkan beberapa contoh perbandingan teks data sebelum dan sesudah dilakukan tokenisasi

Tabel 4.8. Contoh tokenisasi data pada metode TFIDF

Sebelum	Sesudah
<i>Champion Products Inc said its board of directors approved a two-for-one stock split of its common shares for shareholders of record as of April 1 , 1987.</i>	<i>Champion, Products, Inc, said, its, board, of, directors, approved, two, for, one, stock, split, of, its, common, shares, for, shareholders, of, record, as, of, April, 1987</i>
<i>how do u lose 2 inches of fat on ur stomach? Sit ups</i>	<i>how, do, lose, inches, of, fat, on, ur, stomach, Sit, ups</i>

Selanjutnya, diambil representasi data teks yang sudah ditokenisasi menggunakan metode TFIDF. Ilustrasi lebih lanjut dari representasi teks yang dihasilkan dengan metode TFIDF akan diberikan pada Tabel 4.9.

Tabel 4.9. Dokumen untuk ilustrasi TF-IDF

Dokumen 1 (d_1)	<i>board directors approved stock split its common shares</i>
Dokumen 2 (d_2)	<i>stock split common shares shareholders approved</i>
Dokumen 3 (d_3)	<i>shareholders approved reappointment the board directors</i>

Pada Tabel 4.8 terdapat 3 dokumen berbeda dan terdapat beberapa kata yang muncul pada lebih dari 1 dokumen. Perhitungan TFIDF dari setiap kata pada ketiga dokumen yang diberikan pada Tabel 4.9 ditunjukkan pada Tabel 4.10.

Tabel 4.10. Ilustrasi perhitungan TFIDF

t	d	$tf_{t,d}$	df_t	$\log\left(\frac{N}{df_t}\right)$	$w_{t,d}$
<i>board</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	0			0
	3	1			0.176
<i>directors</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	0			0
	3	1			0.176
<i>approved</i>	1	1	3	$\log\left(\frac{3}{3}\right) = 0$	0
	2	1			0
	3	1			0
<i>stock</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	1			0.176
	3	0			0
<i>split</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	1			0.176
	3	0			0
<i>its</i>	1	1	1	$\log\left(\frac{3}{1}\right) = 0.477$	0.477
	2	0			0
	3	0			0
<i>common</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	1			0.176
	3	0			0
<i>shares</i>	1	1	2	$\log\left(\frac{3}{2}\right) = 0.176$	0.176
	2	1			0.176
	3	0			0
<i>shareholders</i>	1	0	2	$\log\left(\frac{3}{2}\right) = 0.176$	0
	2	1			0.176

	3	1			0.176
<i>reappointment</i>	1	0	1	$\log\left(\frac{3}{1}\right) = 0.477$	0
	2	0			0
	3	1			0.477

Berdasarkan perhitungan pada Tabel 4.10, dapat dilihat bahwa setiap dokumen dapat dinyatakan ke dalam bentuk vektor dengan dimensi vektor sebanyak kata yang ada pada semua dokumen. Nilai setiap elemen pada vektor adalah bobot $w_{t,d}$. Dari ilustrasi pada Tabel 4.10, terdapat sebanyak 3 dokumen dan 10 kata. Sehingga hasil representasi kata dengan metode TFIDF yang diperoleh dapat dilihat pada Tabel 4.11.

Tabel 4.11. Hasil akhir metode TFIDF

d	1	2	3
<i>board</i>	0.176	0	0.176
<i>directors</i>	0.176	0	0.176
<i>approved</i>	0	0	0
<i>stock</i>	0.176	0.176	0
<i>split</i>	0.176	0.176	0
<i>its</i>	0.477	0	0
<i>common</i>	0.176	0.176	0
<i>shares</i>	0.176	0.176	0
<i>shareholders</i>	0	0.176	0.176
<i>reappointment</i>	0	0	0.477

Hasil representasi kata dari metode TFIDF berupa matriks berukuran 3×10 dengan entri setiap barisnya adalah nilai pada setiap kolom Tabel 4.11.

Kemudian untuk digunakan dalam model DEC dan IDEC, dilakukan dinormalisasi pada representasi data teks yang dihasilkan metode TFIDF. Dalam tahapan ini, representasi dikalikan dengan akar dari dimensi fitur sehingga untuk suatu vektor representasi data teks ke- i , x_i berdimensi D , diperoleh $\frac{1}{D} \|x_i\|_2^2 = 1$. Sehingga dari metode TFIDF, diperoleh 2 jenis representasi data teks untuk masing-masing data.

4.4.2. Representasi Data Teks dengan Metode BERT

Pada subbab ini dibahas mengenai tahapan pengambilan representasi data teks dengan metode BERT. Sebelum masuk dalam tahapan pengambilan representasi tersebut, beberapa tahapan pra-pengolahan perlu dilakukan terlebih dahulu. Tahapan tersebut tersusun atas tokenisasi, *padding*, dan *encoding*.

Tokenisasi untuk metode BERT dilakukan dengan menggunakan model WordPiece yang dijelaskan pada Subbab 3.2.1. Penambahan token khusus [CLS] dan [SEP] dilakukan di awal dan akhir dokumen. Tabel 4.12 menunjukkan beberapa contoh perbandingan teks data sebelum dan sesudah dilakukan tokenisasi menggunakan model WordPiece

Tabel 4.12. Contoh tokenisasi data pada metode BERT

Sebelum	Sesudah
<i>Reuters - Without fanfare , President Bush signed into law on Friday a nearly 140 billion corporate tax cut bill derided by both Democratic presidential rival John Kerry and Republican Sen. John McCain as a giveaway to special interests .</i>	<i>[CLS], reuters, -, without, fan, ##fare, ,, president, bush, signed, into, law, on, friday, a, nearly, 140, billion, corporate, tax, cut, bill, der, ##ided, by, both, democratic, presidential, rival, john, kerry, and, republican, sen, ., john, mccain, as, a, give, ##away, to, special, interests, ., [SEP]</i>
<i>Reuters - Rescue workers scrabbled through the wreckage of an Egyptian resort hotel on Saturday in a hunt for survivors from bomb blasts targeting Israelis that killed at least 30 people and buried dozens more .</i>	<i>[CLS], reuters, -, rescue, workers, sc, ##ra, ##bbled, through, the, wreckage, of, an, egyptian, resort, hotel, on, saturday, in, a, hunt, for, survivors, from, bomb, blasts, targeting, israelis, that, killed, at, least, 30, people, and, buried, dozens, more, ., [SEP]</i>
<i>When Egypt , Israel and the United States signed a trade agreement this week , it represented the most tangible step in a monthlong thaw in chilly relations between the Egyptians and Israelis .</i>	<i>[CLS], when, egypt, ., israel, and, the, united, states, signed, a, trade, agreement, this, week, ., it, represented, the, most, tangible, step, in, a, month, ##long, tha, ##w, in, chilly, relations, between, the, egyptians, and, israelis, ., [SEP]</i>

Selanjutnya, dilakukan *padding* dan *truncating* terhadap data yang telah ditokenisasi. Banyak maksimal *token* untuk suatu dokumen pada penelitian ini adalah 25 *token*. *Padding* dan *truncating* dilakukan supaya setiap dokumen pada data memiliki panjang yang sama yaitu 25 *token*. Dalam proses *padding*, setiap dokumen yang memiliki banyak *token* kurang dari 25 ditambahkan *token* khusus [PAD] sampai panjang dokumen

mencapai 25 *token*. Sementara dalam proses *truncating*, dokumen yang memiliki lebih dari 25 *token* akan dipotong hanya sampai sebanyak 25 *token* namun dengan token terakhir tetap berupa token khusus [SEP]. Tabel 4.13 menunjukkan beberapa contoh perbandingan *token* dari data teks sebelum dan sesudah dilakukan *padding*.

Tabel 4.13. Contoh *padding* dan *truncating* data pada metode BERT

Sebelum	Sesudah
<i>Reuters - Without fanfare , President Bush signed into law on Friday a nearly 140 billion corporate tax cut bill derided by both Democratic presidential rival John Kerry and Republican Sen. John McCain as a giveaway to special interests .</i>	<i>[CLS], reuters, -, without, fan, ##fare, ,, president, bush, signed, into, law, on, friday, a, nearly, 140, billion, corporate, tax, cut, bill, der, ##ided, [SEP]</i>
<i>Reuters - Rescue workers scrabbled through the wreckage of an Egyptian resort hotel on Saturday in a hunt for survivors from bomb blasts targeting Israelis that killed at least 30 people and buried dozens more .</i>	<i>[CLS], reuters, -, rescue, workers, sc, ##ra, ##bbled, through, the, wreckage, of, an, egyptian, resort, hotel, on, saturday, in, a, hunt, for, survivors, from, [SEP]</i>
<i>When Egypt , Israel and the United States signed a trade agreement this week , it represented the most tangible step in a monthlong thaw in chilly relations between the Egyptians and Israelis .</i>	<i>[CLS], when, egypt, ,, israel, and, the, united, states, signed, a, trade, agreement, this, week, ,, it, represented, the, most, tangible, step, in, a, [SEP]</i>

Tahap berikutnya adalah *encoding*. *Encoding* bertujuan untuk mentransformasi *token* menjadi bilangan bulat sehingga dokumen dapat dibaca oleh model BERT. *Encoding* dilakukan dengan memanfaatkan pemetaan dari model WordPiece yang mengandung 30000 *token* sebagai *key* dan bilangan bulat unik yang bersesuaian sebagai *value*. *Token* pada setiap dokumen akan dipetakan menuju bilangan bulat yang bersesuaian sehingga sekarang setiap *token* yang berbeda direpresentasikan dengan bilangan yang berbeda juga. Tabel 4.14 menunjukkan beberapa contoh perbandingan *token* dari data teks sebelum dan sesudah dilakukan *encoding*.

Tabel 4.14. Contoh *encoding* data pada metode BERT

Sebelum	Sesudah
[CLS], reuters, -, without, fan, ##fare, ,, president, bush, signed, into, law, on, friday, a, nearly, 140, billion, corporate, tax, cut, bill, der, ##ided, [SEP]	101, 26665, 1011, 2302, 5470, 17883, 1010, 2343, 5747, 2772, 2046, 2375, 2006, 5958, 1037, 3053, 8574, 4551, 5971, 4171, 3013, 3021, 4315, 14097, 102
[CLS], reuters, -, rescue, workers, sc, ##ra, ##bled, through, the, wreckage, of, an, egyptian, resort, hotel, on, saturday, in, a, hunt, for, survivors, from, [SEP]	101, 26665, 1011, 5343, 3667, 8040, 2527, 12820, 2083, 1996, 21056, 1997, 2019, 6811, 7001, 3309, 2006, 5095, 1999, 1037, 5690, 2005, 8643, 2013, 102
[CLS], when, egypt, ,, israel, and, the, united, states, signed, a, trade, agreement, this, week, ,, it, represented, the, most, tangible, step, in, a, [SEP]	101, 2043, 5279, 1010, 3956, 1998, 1996, 2142, 2163, 2772, 1037, 3119, 3820, 2023, 2733, 1010, 2009, 3421, 1996, 2087, 24600, 3357, 1999, 1037, 102

Model BERT yang digunakan penelitian ini adalah BERT-base *uncased*. BERT-base *uncased* adalah model BERT yang menggunakan data yang tidak memiliki huruf kapital ketika *pre-training*. Model ini memiliki 12 lapisan *transformer encoder*, *hidden size* sebanyak 768, dan banyak *head* pada sub-lapisan *attention* sebanyak 12. Representasi data teks diperoleh dengan melakukan *feed-forward* pada data yang sudah diolah ke dalam model seperti yang dijelaskan pada Subbab 3.3.1 dan diilustrasikan pada Gambar 3.15. Pada penelitian ini, representasi data teks diambil dari *output* yang diberikan lapisan *transformer encoder* kedua sebelum terakhir (lapisan ke-11). *Output* yang diperoleh adalah tensor berukuran ($n, 25,768$), dengan n adalah banyak dokumen di dalam *dataset*, 25 adalah banyak *token* dalam dokumen, dan 768 adalah *hidden size*.

Selanjutnya diterapkan ekstraksi fitur berupa *mean pooling* atau *max pooling* sehingga *output* tensor berukuran ($n, 25,768$) ditransformasi menjadi matriks berukuran ($n, 768$). Terakhir, dua jenis matriks yang diperoleh dilakukan proses normalisasi fitur menggunakan *identity normalization*, *standard normalization*, *layer normalization*, atau *min-max normalization*. Sehingga dari model BERT, diperoleh total sebanyak 8 jenis matriks representasi data teks untuk satu data.

Selanjutnya akan dibahas mengenai tahapan simulasi berbagai model *text clustering* yang digunakan pada penelitian ini. Simulasi *text clustering* dilakukan dengan menggunakan representasi data teks dari semua *dataset* yang sudah diperoleh sebelumnya.

4.5. Simulasi *Text Clustering*

Simulasi *text clustering* pada penelitian ini dilakukan dengan repetisi sebanyak 50 kali tanpa melakukan perubahan untuk setiap model dan pada setiap *dataset*. Hal ini bertujuan untuk memeriksa kestabilan dari representasi data teks yang dihasilkan. Pada bagian ini dijelaskan mengenai pengaturan *hyperparameter* yang digunakan pada model *k-means clustering*, *eigenspace-based fuzzy c-means*, *deep embedded clustering*, dan *improved deep embedded clustering*.

4.5.1. Model *K-Means Clustering*

Representasi data teks yang diperoleh melalui metode TFIDF dan metode BERT selanjutnya menjadi input bagi *k-means clustering*. Banyak *cluster* yang digunakan sesuai dengan banyak kelas pada *ground-truth label* masing-masing data. *Hyperparameter* yang digunakan algoritma *k-means clustering* dapat dilihat pada Tabel 4.15 (Pedregosa *et al.*, 2011):

Tabel 4.15. Hyperparameter *k-means clustering*

Hyperparameter	Argumen
Metode inisialisasi (<i>init</i>)	<i>k-means++</i>
Banyak repetisi dengan <i>seed</i> yang berbeda (<i>n_init</i>)	10
Iterasi maksimal (<i>max_iter</i>)	300
Ambang batas toleransi (<i>tol</i>)	10^{-4}
Banyak penggunaan <i>thread</i> untuk komputasi (<i>n_jobs</i>)	10
Algoritma <i>k-means</i> yang digunakan (<i>algorithm</i>)	<i>auto</i>

4.5.2. Model *Eigenspace-based Fuzzy C-Means*

Proses *text clustering* menggunakan model EFCM diawali dengan mentransformasikan data menjadi dimensi yang lebih rendah dengan menggunakan dekomposisi TSVD. Selanjutnya dilakukan proses *clustering* menggunakan algoritma FCM dengan data yang sudah direduksi tersebut. Setelah algoritma EFCM konvergen, *cluster* yang memiliki nilai *membership* tertinggi yang akan dipilih sebagai *cluster* prediksi dari masing-masing data. *Hyperparameter* yang digunakan dalam algoritma EFCM dapat dilihat pada Tabel 4.16 (Warner *et al.*, 2019).

Tabel 4.16. Hyperparameter eigen-space based fuzzy c-means

Hyperparameter	Argumen
Banyak komponen utama TSVD (<i>n_components</i>)	5
Derajat fuzzy (<i>m</i>)	1.1
Ambang batas toleransi (<i>error</i>)	10^{-4}
Iterasi maksimal (<i>maxiter</i>)	200
Inisialisasi matriks fuzzy <i>c-partitioned</i> (<i>init</i>)	None

4.5.3. Model Deep Embedded Clustering

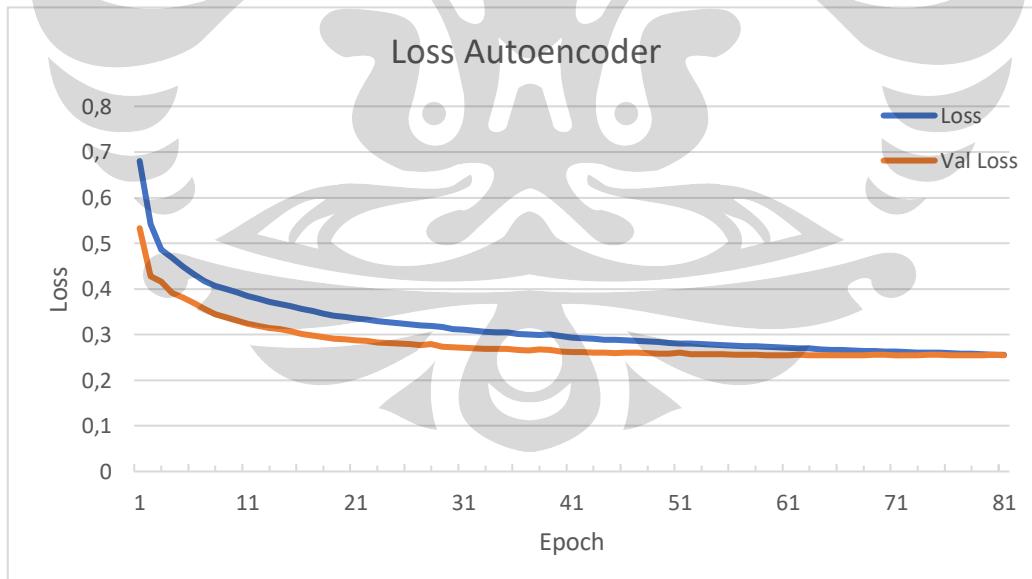
Pada bagian ini dijelaskan implementasi model DEC dalam melakukan *text clustering*. Implementasi dari model DEC untuk *text clustering* merujuk pada penelitian Guo, Gao, Liu, dan Yin (2017). Misalkan d adalah dimensi fitur dari data yang digunakan, d akan bernilai 2000 ketika menggunakan representasi data dengan metode TFIDF dan bernilai 768 ketika menggunakan representasi data dengan metode BERT. Tabel 4.17 menunjukkan *hyperparameter* yang digunakan dalam model DEC dalam penelitian ini (Guo, Gao, Liu, dan Yin, 2017):

Tabel 4.17. Hyperparameter deep embedded clustering

Tahap	Hyperparameter	Argumen
Pre-training	Jumlah neuron autoencoder	<ul style="list-style-type: none"> • $d - 500 - 500 - 2000 - 5$ • $- 2000 - 500$ • $- 500 - d$
	Fungsi Aktivasi <ul style="list-style-type: none"> • Lapisan <i>encoder</i> dan <i>decoder</i> • Lapisan <i>code</i> dan <i>output</i> 	<ul style="list-style-type: none"> • <i>Rectified Linear Unit</i> • Linier
	Banyak <i>epoch</i> (<i>epochs</i>)	500
	<i>earlystopping</i>	<ul style="list-style-type: none"> • monitor <i>validation loss</i> • patience 10 • restore_best_weights <i>True</i>
	<i>validation_split</i>	0.1
	<i>batch_size</i>	256

	<i>pretrain_optimizer</i>	<i>adam</i>
Optimisasi	<i>alpha</i>	1
	<i>init</i>	VarianceScaling(scale=1/3, mode='fan_in', distribution='uniform')
	<i>optimizer</i>	<i>adam</i>
	<i>Update_interval</i>	30
	<i>tol</i>	10^{-4}
	<i>batch_size</i>	256

Pada tahapan *pre-training*, tujuan dari optimisasi yang dilakukan adalah meminimumkan fungsi *loss* yaitu MSE. Tahapan *pre-training autoencoder* dilakukan secara sekaligus menggunakan *earlystopping* dengan *patience* 10, *validation loss* sebagai nilai yang diawasi, dan maksimal iterasi 500. Sehingga, iterasi tahap *pre-training* akan terus berlangsung sampai nilai *val_loss* tidak mengalami penurunan selama 10 iterasi atau sampai iterasi mencapai 500. Pembaharuan bobot pada tahapan *pretraining* menggunakan Adam.



Gambar 4.2. *Loss pre-training autoencoder* pada model DEC

Gambar 4.2 menunjukkan salah satu contoh nilai *loss* dan *validation loss* dari tahapan *pretraining* pada setiap iterasi. Contoh ini diambil pada tahapan *pretraining autoencoder* ketika menggunakan representasi data teks R2 dari metode BERT dengan

max pooling dan *layer normalization*. Gambar 4.2 menunjukkan bahwa *pretraining* baru berhenti ketika sudah memenuhi persyaratan yang disebutkan sebelumnya, dalam kasus ini *pretraining* berhenti pada iterasi yang ke-82.

Selanjutnya pada tahapan optimisasi, data direduksi terlebih dahulu menjadi dimensi 5 dengan menggunakan *autoencoder* yang sudah melalui proses *pre-training*. Kemudian inisialisasi *cluster* dilakukan dengan menggunakan *k-means clustering* dengan banyak *cluster* sesuai dengan banyak kelas *ground-truth label* yang dimiliki masing-masing *dataset*. Kemudian pembelajaran model dilakukan dengan meminimumkan divergensi KL dan pembaharuan bobot dilakukan dengan menggunakan metode Adam.

4.5.4. Model *Improved Deep Embedded Clustering*

Pada bagian ini akan dijelaskan implementasi model IDEC dalam melakukan *text clustering*. Implementasi dari model IDEC untuk *text clustering* merujuk pada penelitian Guo, Gao, Liu, dan Yin (2017) Misalkan d adalah dimensi fitur dari data yang digunakan, d akan sebesar 2000 ketika menggunakan representasi data dengan metode TFIDF dan sebesar 768 ketika menggunakan representasi data dengan metode BERT. Tabel 4.18 menunjukkan *hyperparameter* yang digunakan dalam model IDEC dalam penelitian ini (Guo, Gao, Liu, dan Yin, 2017):

Tabel 4.18. *Hyperparameter improved deep embedded clustering*

Tahap	Hyperparameter	Argumen
Pre-training	Jumlah neuron <i>autoencoder</i>	$d - 500 - 500 - 2000 - 5$ $- 2000 - 500$ $- 500 - d$
	Fungsi Aktivasi <ul style="list-style-type: none"> Lapisan <i>encoder</i> dan <i>decoder</i> Lapisan <i>code</i> dan <i>output</i> 	<ul style="list-style-type: none"> <i>Rectified Linear Unit</i> Linier
	Banyak <i>epoch</i> (<i>epochs</i>)	500
	<i>earlystopping</i>	<ul style="list-style-type: none"> monitor <i>val_loss</i> patience 10 restore_best_weights <i>True</i>

	<i>validation_split</i>	0.1
	<i>batch_size</i>	256
	<i>pretrain_optimizer</i>	<i>adam</i>
	<i>gamma</i>	0.1
Optimisasi	<i>alpha</i>	1
	<i>optimizer</i>	<i>adam</i>
	<i>Update_interval</i>	30
	<i>tol</i>	10^{-6}
	<i>batch_size</i>	256

Tahapan *pre-training* yang digunakan pada model IDEC sama dengan tahapan *pretraining* pada model DEC yang dijelaskan pada Subbab 4.5.4. Pada tahapan optimisasi, data direduksi terlebih dahulu menjadi dimensi 5 dengan menggunakan *autoencoder* yang sudah melalui proses *pre-training*. Kemudian inisialisasi *cluster* dilakukan dengan menggunakan *k-means clustering* dengan banyak *cluster* sesuai dengan banyak kelas *ground-truth label* yang dimiliki masing-masing *dataset*. Kemudian pembelajaran model dilakukan dengan meminimumkan divergensi KL dan pembaharuan bobot dilakukan dengan menggunakan metode Adam.

4.6. Metrik Evaluasi

Pada subbab ini dibahas mengenai berbagai metrik yang digunakan untuk mengevaluasi cluster yang dihasilkan dari keempat model text clustering yang dijelaskan sebelumnya. Metrik evaluasi yang digunakan adalah metrik yang dapat mengukur performa dari model *text clustering* dengan menggunakan data yang sudah memiliki *ground-truth label*. Sebelumnya penjelasan mengenai metrik evaluasi, akan dijelaskan terlebih dahulu mengenai tabel kontingensi. Misalkan terdapat sebuah himpunan S dengan n elemen. Misalkan terdapat dua partisi untuk himpunan S , yaitu $\mathbf{u} = \{u_1, u_2, \dots, u_R\}$ dan $\mathbf{v} = \{v_1, v_2, \dots, v_C\}$, dengan $\bigcup_{i=1}^R u_i = \bigcup_{j=1}^C v_j = S$ dan $u_i \cap u_{i'} = \emptyset$ dan $v_j \cap v_{j'} = \emptyset$ untuk $i \neq i'$ dan $j \neq j'$. Tabel kontingensi yang membandingkan dua buah partisi dapat diilustrasikan pada Gambar 4.3:

Class	v_1	v_2	...	v_C	Sums
u_1	n_{11}	n_{12}	...	n_{1C}	$n_{1\cdot}$
u_2	n_{21}	n_{22}		n_{2C}	$n_{2\cdot}$
.
.
u_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R\cdot}$
Sums	$n_{\cdot 1}$	$n_{\cdot 2}$...	$n_{\cdot C}$	$n_{..} = n$

Gambar 4.3. Tabel kontingensi antara dua partisi

Sumber: (Hubert & Arabie, 1985), telah diolah kembali

dengan n_{ij} adalah banyak elemen yang berada di kedua partisi u_i dan v_j , $n_{i\cdot}$ dan $n_{\cdot j}$ berturut-turut merepresentasikan banyak elemen di dalam partisi u_i dan v_j .

Selanjutnya akan dijelaskan 3 buah metrik yang digunakan, yaitu *clustering accuracy*, *normalized mutual information*, dan *adjusted rand index*.

4.6.1. Clustering Accuracy

Clustering accuracy (ACC) adalah algoritma yang mencari pemetaan terbaik antara *cluster* yang diperoleh dari algoritma *unsupervised* yang digunakan dengan *ground-truth label*. Nilai dari ACC dapat ditentukan melalui persamaan (4.1) (Xu, Liu, Gong, 2003):

$$ACC = \frac{\sum_{i=1}^n \delta(\alpha_i, map(l_i))}{n}, \quad (4.1)$$

dengan α_i adalah *ground-truth label* dari teks ke- i dan l_i adalah *label* dari *cluster* yang diperoleh dari algoritma *unsupervised* yang digunakan. Fungsi $\delta(x, y)$ adalah fungsi yang akan memetakan ke nilai 1 jika $x = y$ dan akan memetakan ke nilai 0 jika sebaliknya. Fungsi *map(.)* akan memetakan *label* dari *cluster* yang dimiliki menuju *ground-truth label* sedemikian sehingga memberikan nilai ACC yang terbaik. (Xie, Girshick, & Farhadi, 2016).

Sebagai ilustrasi didefinisikan p adalah himpunan yang beranggotakan *ground truth label* observasi ke- i $\{p_1, p_2, \dots, p_n\}$, dan q adalah himpunan yang beranggotakan hasil *cluster* observasi ke- i $\{q_1, q_2, \dots, q_n\}$. Misalkan himpunan p dan q memiliki nilai sebagai berikut:

$$\begin{aligned} p &= \{0,0,0,0,0,1,1,1,1,1\}, \\ q &= \{0,0,0,1,1,0,1,1,0,0,1,1\}. \end{aligned} \quad (4.2)$$

Nilai ACC yang diperoleh menggunakan persamaan (4.1) adalah sebagai berikut:

$$\begin{aligned}
 ACC &= \frac{1}{9} (\delta(0,0) + \delta(0,0) + \delta(0,0) + \delta(0,1) + \delta(0,1) + \delta(0,0) + \delta(1,1) + \delta(1,1) \\
 &\quad + \delta(1,0) + \delta(1,0) + \delta(1,1) + \delta(1,1)) \\
 &= \frac{1+1+1+0+0+1+1+1+0+0+1+1}{12} \\
 &= \frac{8}{12} = 0.75
 \end{aligned}$$

Pada Subbab 4.6.2 dibahas mengenai metrik evaluasi kedua yang digunakan pada penelitian ini, yaitu *normalized mutual information*.

4.6.2. Normalized Mutual Information

Misalkan terdapat dua variabel acak X dan Y . *Mutual information* adalah pengukuran yang menentukan besaran informasi yang diperoleh X apabila diketahui Y . *Normalized Mutual Information* (NMI) adalah nilai *mutual information* yang dinormalisasi sehingga nilai *mutual information* yang awalnya tidak terbatas menjadi berada dalam rentang nilai $[0,1]$. Misalkan U adalah *ground-truth label* dan V adalah *label cluster* yang diprediksi menggunakan algoritma *unsupervised*. Nilai dari NMI dapat ditentukan dalam persamaan (4.3) (Xu, Liu, Gong, 2003):

$$NMI(U, V) = \frac{MI(U, V)}{\sqrt{H(U)H(V)}}, \quad (4.3)$$

dengan parameter $H(U)$, $H(V)$, dan $MI(U, V)$ secara berturut-turut diperoleh dari persamaan (4.4) sampai (4.6):

$$H(U) = -\sum_i \frac{n_{i.}}{N} \ln \left(\frac{n_{i.}}{N} \right), \quad (4.4)$$

$$H(V) = -\sum_i \frac{n_{.j}}{N} \ln \left(\frac{n_{.j}}{N} \right), \quad (4.5)$$

$$MI(U, V) = \sum_{i,j} \frac{n_{ij}}{N} \ln \left(N \frac{n_{ij}}{n_{i.} n_{.j}} \right), \quad (4.6)$$

dengan N adalah banyak data, n_{ij} , $n_{i.}$, dan $n_{.j}$ adalah nilai yang dijelaskan pada tabel kontingensi Gambar 4.3. $MI(U, V)$ adalah fungsi *mutual information* antara *cluster* yang diprediksi V dengan *ground-truth label* U . Normalisasi dilakukan dengan membagi nilai fungsi *mutual information* dengan $\sqrt{H(L)H(C)}$.

Sebagai ilustrasi didefinisikan p adalah himpunan yang beranggotakan *ground truth label* untuk observasi ke- i $\{p_1, p_2, \dots, p_n\}$, dan q adalah himpunan yang beranggotakan hasil *cluster* dari observasi ke- i $\{q_1, q_2, \dots, q_n\}$. Misalkan himpunan p dan q memiliki nilai pada (4.2). Nilai $H(p)$, $H(q)$, dan $MI(p, q)$ ditentukan terlebih dahulu sebelum mencari nilai $NMI(p, q)$. Nilai $H(p)$ dan $H(q)$ diperoleh dengan menggunakan persamaan (4.4) dan (4.5) sebagai berikut:

$$\begin{aligned} H(p) &= -\left(\frac{6}{12} \ln\left(\frac{6}{12}\right) + \frac{6}{12} \ln\left(\frac{6}{12}\right)\right) \\ &= -(-0.34657 - 0.34657) = 0.69314 \\ H(q) &= -\left(\frac{4}{12} \ln\left(\frac{4}{12}\right) + \frac{2}{12} \ln\left(\frac{2}{12}\right) + \frac{2}{12} \ln\left(\frac{2}{12}\right)\right) \\ &= -(-0.34657 - 0.34657) = 0.69314 \end{aligned}$$

Nilai $MI(p, q)$ diperoleh dengan menggunakan persamaan (4.6) sebagai berikut:

$$\begin{aligned} MI(p, q) &= \frac{4}{12} \ln\left(12 \frac{4}{6 \times 6}\right) + \frac{2}{12} \ln\left(12 \frac{2}{6 \times 6}\right) + \frac{2}{12} \ln\left(12 \frac{2}{6 \times 6}\right) \\ &\quad + \frac{4}{12} \ln\left(12 \frac{4}{6 \times 6}\right) \\ &= 0.095894 - 0.067578 - 0.067578 + 0.095894 \\ &= 0.056632 \end{aligned}$$

Sehingga dapat diperoleh nilai $NMI(p, q)$ dengan menggunakan persamaan (4.3) sebagai berikut:

$$NMI(p, q) = \frac{0.056632}{\sqrt{0.69314 \times 0.69314}} = 0.081704.$$

4.6.3. Adjusted Rand Index

Berdasarkan banyaknya pasangan elemen yang berada pada subhimpunan yang sama dan juga banyaknya pasangan elemen yang berada pada subhimpunan yang berbeda. Metrik ARI berada dalam rentang nilai $[0,1]$. Pencarian nilai ARI dapat mengikuti persamaan (4.7) (Yeung & Ruzzo, 2001):

$$ARI = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \frac{[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}]}{\binom{n}{2}}}{\frac{1}{2} [\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] - \frac{[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}]}{\binom{n}{2}}}, \quad (4.7)$$

dengan n adalah banyak data, n_{ij} , $n_{i\cdot}$, dan $n_{\cdot j}$ adalah nilai yang dijelaskan pada tabel kontingensi Gambar 4.3. Sebagai ilustrasi didefinisikan p adalah himpunan yang beranggotakan *ground truth label* untuk observasi ke- i $\{p_1, p_2, \dots, p_n\}$, dan q adalah himpunan yang beranggotakan hasil *cluster* dari observasi ke- i $\{q_1, q_2, \dots, q_n\}$. Misalkan himpunan p dan q memiliki nilai pada (4.2). Nilai-nilai $\sum_{i,j} \binom{n_{ij}}{2}$, $\sum_i \binom{n_{i\cdot}}{2}$, $\sum_j \binom{n_{\cdot j}}{2}$, dan $\binom{n}{2}$ ditentukan terlebih dahulu sebagai berikut:

$$\sum_{i,j} \binom{n_{ij}}{2} = \binom{4}{2} + \binom{2}{2} + \binom{2}{2} + \binom{4}{2} = 6 + 1 + 1 + 6 = 14$$

$$\sum_i \binom{n_{i\cdot}}{2} = \binom{6}{2} + \binom{6}{2} = 15 + 15 = 30$$

$$\sum_j \binom{n_{\cdot j}}{2} = \binom{6}{2} + \binom{6}{2} = 15 + 15 = 30$$

$$\binom{n}{2} = \binom{12}{2} = 66$$

Dengan menggunakan informasi nilai yang sudah diperoleh di atas, maka nilai ARI untuk himpunan p dan q dapat diperoleh menggunakan persamaan (4.7) sebagai berikut:

$$\begin{aligned} ARI &= \frac{14 - \frac{30 \times 30}{66}}{\frac{1}{2}(30 + 30) - \frac{30 \times 30}{66}} \\ &= \frac{14 - \frac{150}{11}}{30 - \frac{150}{11}} \\ &= \frac{\frac{4}{11}}{\frac{180}{11}} = \frac{1}{45} = 0.02222 \end{aligned}$$

Pada Subbab 4.7 dibahas mengenai hasil simulasi *text clustering* yang telah dilakukan serta analisis dan pembahasan dari hasil tersebut.

4.7. Hasil dan Analisis Simulasi

Pada subbab ini dipaparkan dan dianalisis hasil dari simulasi keempat model yang dijelaskan sebelumnya. Hasil *clustering* pada setiap *dataset* dievaluasi menggunakan 3 metrik yang berbeda sehingga secara keseluruhan akan terdapat 36 metrik yang diperhatikan dari 3 *dataset* dan 4 model *clustering*. Metrik yang diukur adalah kualitas

cluster yang dihasilkan representasi data teks dengan TFIDF dan BERT pada model *k-means clustering*, dilanjutkan dengan model *eigenspace-based fuzzy c-means*, dilanjutkan dengan *deep embedded clustering*, dan pada *improved deep embedded clustering*. Tabel 4.19 memberikan deskripsi mengenai beberapa penyederhanaan yang digunakan pada bagian berikutnya dari subbab ini.

Tabel 4.19. Deskripsi penyederhanaan

Singkatan	Keterangan
TFIDF	Representasi data teks dengan <i>term frequency inversed document frequency</i>
BERT	Representasi data teks dengan <i>Bidirectional encoder representation from transformer</i>
Max	Ekstraksi fitur dengan menggunakan <i>max pooling</i>
Mean	Ekstraksi fitur dengan menggunakan <i>mean pooling</i>
I	Normalisasi fitur dengan menggunakan <i>identity normalization</i>
LN	Normalisasi fitur secara menggunakan <i>layer normalization</i>
N	Normalisasi fitur secara menggunakan <i>standard normalization</i>
MM	Normalisasi fitur secara menggunakan <i>min-max normalization</i>

4.7.1. Performa pada Model *K-Means Clustering*

Tabel 4.20 menunjukkan hasil simulasi *text clustering* menggunakan model *k-means clustering* pada data *AG news*. Pada Tabel 4.20 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Peningkatan signifikan dapat terlihat pada metrik ACC yang semula rata-rata sebesar 0.5019 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 57.66% menjadi 0.7913. Kemudian peningkatan signifikan juga ditunjukkan pada metrik NMI yang awalnya sebesar 0.2559 pada metode TFIDF mengalami peningkatan sebesar 103.15% menjadi 0.5199. Hal serupa juga ditunjukkan dari metrik lain yaitu ARI yang awalnya sebesar 0.2552 mengalami peningkatan sebesar 103.57% menjadi 0.5195.

Tabel 4.20. Hasil evaluasi *k-means clustering* pada data *AG news*

Dataset	<i>AG news</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.5019±0.0718	0.2559±0.0802	0.2552±0.0803
BERT + Max + I	0.7674±0.0018	0.4872±0.0021	0.4868±0.0021
BERT + Max + LN	0.7913±0.0040	0.5199±0.0050	0.5195±0.0050
BERT + Max + N	0.7858±0.0017	0.5136±0.0025	0.5132±0.0025
BERT + Max + MM	0.4408±0.0012	0.1986±0.0014	0.1979±0.0014
BERT + Mean + I	0.6491±0.0016	0.4196±0.0010	0.4191±0.0010
BERT + Mean + LN	0.6468±0.0036	0.4152±0.0018	0.4148±0.0018
BERT + Mean + N	0.6467±0.0033	0.4151±0.0017	0.4146±0.0017
BERT + Mean + MM	0.3208±0.0051	0.0441±0.0008	0.0432±0.0008

Tabel 4.21 menunjukkan hasil simulasi *text clustering* menggunakan model *k-means clustering* pada data *Yahoo! Answers*. Pada Tabel 4.21, evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.3568 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 4.85% menjadi 0.3741. Metrik NMI yang awalnya sebesar 0.2135 mengalami peningkatan sebesar 7.82% menjadi 0.2302. Terakhir, metrik ARI yang awalnya sebesar 0.2121 pada metode TFIDF mengalami peningkatan sebesar 7.87% menjadi 0.2288.

Tabel 4.21. Hasil evaluasi *k-means clustering* pada data *Yahoo! Answers*

Dataset	<i>Yahoo! Answers</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.3568±0.0059	0.2135±0.0067	0.2121±0.0068
BERT + Max + I	0.3018±0.0131	0.1495±0.0095	0.1479±0.0095
BERT + Max + LN	0.3285±0.0140	0.1797±0.0132	0.1782±0.0132
BERT + Max + N	0.3229±0.0145	0.1739±0.0137	0.1724±0.0137

BERT + Max + MM	0.226±0.0058	0.088±0.0057	0.0864±0.0057
BERT + Mean + I	0.357±0.0079	0.2134±0.0077	0.212±0.0077
BERT + Mean + LN	0.3741±0.0057	0.2302±0.0055	0.2288±0.0055
BERT + Mean + N	0.373±0.0071	0.2286±0.0066	0.2273±0.0066
BERT + Mean + MM	0.1718±0.0066	0.0511±0.0050	0.0493±0.0050

Tabel 4.22 menunjukkan hasil simulasi *text clustering* menggunakan model *k-means clustering* pada data *Yahoo! Answers*. Pada Tabel 4.22 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT. Metrik evaluasi ACC tertinggi ditunjukkan pada saat menggunakan metode ekstraksi *mean pooling* dan metode normalisasi *layer normalization* atau *standard normalization*. Sementara metrik NMI dan ARI terbaik ditunjukkan ketika menggunakan metode *max pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang rata-rata sebesar 0.8471 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sekitar 0.425% menjadi 0.8507. Metrik NMI yang sebesar 0.5034 pada metode TFIDF mengalami peningkatan sekitar 0.358% menjadi 0.5052. Terakhir, metrik ARI yang awalnya sebesar 0.5033 pada metode TFIDF mengalami peningkatan sekitar 0.378% menjadi 0.5052.

Tabel 4.22. Hasil evaluasi *k-means clustering* pada data *R2*

Dataset	<i>Reuters (R2)</i>		
	ACC	NMI	ARI
TFIDF	0.8471±0	0.5034±0	0.5033±0
BERT + Max + I	0.8457±0	0.5025±0	0.5024±0
BERT + Max + LN	0.8472±0	0.5052±0	0.5052±0
BERT + Max + N	0.8469±0	0.4985±0.0015	0.4984±0.0015
BERT + Max + MM	0.8495±0	0.4942±0	0.4941±0
BERT + Mean + I	0.8471±0	0.5034±0	0.5033±0
BERT + Mean + LN	0.8507±0	0.5036±0	0.5035±0
BERT + Mean + N	0.8507±0	0.5036±0	0.5035±0
BERT + Mean + MM	0.6624±0.0002	0.0822±0.0003	0.0821±0.0003

4.7.2. Performa pada Model *Eigenbased-space Fuzzy C-Means*

Tabel 4.23 menunjukkan hasil simulasi *text clustering* menggunakan model *eigenbased-space fuzzy c-means* pada data *AG news*. Pada Tabel 4.23 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan *max pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.5788 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan signifikan sebesar 34.42% menjadi 0.778. Metrik NMI yang awalnya sebesar 0.2979 mengalami peningkatan signifikan sebesar 67.04% menjadi 0.4976. Terakhir, metrik ARI yang awalnya sebesar 0.2973 pada metode TFIDF mengalami peningkatan signifikan sebesar 67.24% menjadi 0.4972.

Tabel 4.23. Hasil evaluasi *eigenspace-based fuzzy c-means* pada data *AG news*

Dataset	<i>AG news</i>		
	ACC	NMI	ARI
TFIDF	0.5788±0.03197	0.2979±0.0309	0.2973±0.0309
BERT + Max + I	0.7561±0.0004	0.4731±0.0006	0.4726±0.0006
BERT + Max + LN	0.778±0.0002	0.4976±0.0004	0.4972±0.0004
BERT + Max + N	0.7642±0.0003	0.4841±0.0004	0.4837±0.0004
BERT + Max + MM	0.4439±0.0085	0.1997±0.0100	0.1991±0.0100
BERT + Mean + I	0.6449±0.0003	0.4086±0.0002	0.4081±0.0002
BERT + Mean + LN	0.6423±0.0003	0.4088±0.0003	0.4083±0.0003
BERT + Mean + N	0.6425±0.0003	0.4089±0.0003	0.4084±0.0003
BERT + Mean + MM	0.3067±0.0037	0.0429±0.0003	0.0421±0.0003

Tabel 4.24 menunjukkan hasil simulasi *text clustering* menggunakan model *eigenspace-based fuzzy c-means* pada data *Yahoo! Answers*. Pada Tabel 4.24 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT. Metrik ACC yang terbaik ditunjukkan ketika menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *standard normalization* untuk normalisasi fitur. Sementara metrik NMI dan ARI terbaik

keduanya ditunjukkan ketika menggunakan metode *max pooling* untuk ekstraksi fitur dan *layer normalization* untuk normalisasi fitur. Performa ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.2482 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 1.65% menjadi 0.2523. Metrik NMI yang awalnya sebesar 0.1177 mengalami peningkatan sebesar 10.62% menjadi 0.1302. Terakhir, metrik ARI yang awalnya sebesar 0.1161 pada metode TFIDF mengalami peningkatan sebesar 10.85% menjadi 0.1287.

Tabel 4.24. Hasil evaluasi *eigenspace-based fuzzy c-means* pada data *Yahoo! Answers*

Dataset	<i>Yahoo! Answers</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.2482±0.0081	0.1177±0.0018	0.1161±0.0018
BERT + Max + I	0.2484±0.0070	0.122±0.0029	0.1204±0.0029
BERT + Max + LN	0.2454±0.0069	0.1302±0.0015	0.1287±0.0015
BERT + Max + N	0.2374±0.0051	0.1255±0.0014	0.1239±0.0014
BERT + Max + MM	0.2043±0.0098	0.0706±0.0044	0.0689±0.0044
BERT + Mean + I	0.2486±0.0077	0.1174±0.0016	0.1158±0.0016
BERT + Mean + LN	0.2522±0.0037	0.1253±0.0012	0.1237±0.0012
BERT + Mean + N	0.2523±0.0032	0.1252±0.0010	0.1236±0.0010
BERT + Mean + MM	0.1632±0.0066	0.0415±0.0019	0.0397±0.0019

Tabel 4.25 menunjukkan hasil simulasi *text clustering* menggunakan model *eigenspace-based fuzzy c-means* pada data R2. Pada Tabel 4.25 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *layer normalization* atau *standard normalization* untuk normalisasi fitur. Metrik ACC yang semula rata-rata sebesar 0.8476 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sekitar 0.342% menjadi 0.8505. Sementara metrik terbaik untuk NMI dan ARI secara berturut-turut adalah sebesar 0.5043 dan 0.5042. Nilai NMI dan ARI yang terbaik ditunjukkan pada kedua metode representasi teks, yaitu pada metode TFIDF dan metode BERT dengan *mean pooling* dan *identity normalization*.

Tabel 4.25. Hasil evaluasi *eigenspace-based fuzzy c-means* pada data *R2*

Dataset	Reuters (R2)		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.8476±0	0.5043±0	0.5042±0
BERT + Max + I	0.8462±0	0.5034±0	0.5033±0
BERT + Max + LN	0.8474±0	0.504±0	0.5039±0
BERT + Max + N	0.8479±0	0.4964±0	0.4964±0
BERT + Max + MM	0.8498±0	0.4957±0	0.4957±0
BERT + Mean + I	0.8476±0	0.5043±0	0.5042±0
BERT + Mean + LN	0.8505±0	0.5±0	0.4999±0
BERT + Mean + N	0.8505±0	0.5±0	0.4999±0
BERT + Mean + MM	0.6636±0	0.0827±0	0.0826±0

4.7.3. Performa pada Model *Deep Embedded Clustering*

Tabel 4.26 menunjukkan hasil simulasi *text clustering* menggunakan model *deep embedded clustering* pada data *AG news*. Pada Tabel 4.26 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *standard normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.7211 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 11.47% menjadi 0.8038. Metrik NMI yang awalnya rata-rata sebesar 0.3861 pada metode TFIDF mengalami peningkatan signifikan sebesar 39.34% menjadi 0.538. Terakhir, metrik ARI yang awalnya rata-rata bernilai 0.4139 pada metode TFIDF mengalami peningkatan signifikan sekitar 37.88% menjadi 0.5707.

Tabel 4.26. Hasil evaluasi *deep embedded clustering* pada data *AG news*

Dataset	<i>AG news</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.7211±0.0250	0.3861±0.0265	0.4139±0.0338
BERT + Max + I	0.2539±0.0274	0.0037±0.0259	0.003±0.0210
BERT + Max + LN	0.7677±0.0436	0.4878±0.0344	0.513±0.0483

BERT + Max + N	0.2585±0.0326	0.004±0.0179	0.0033±0.0162
BERT + Max + MM	0.3529±0.1505	0.0817±0.1476	0.0798±0.1461
BERT + Mean + I	0.7719±0.0506	0.5055±0.0363	0.5304±0.0518
BERT + Mean + LN	0.7653±0.0550	0.4987±0.0426	0.5206±0.0579
BERT + Mean + N	0.8038±0.0325	0.538±0.0210	0.5707±0.0296
BERT + Mean + MM	0.25±0	0.0004±0.0018	0±0

Tabel 4.27 menunjukkan hasil simulasi *text clustering* menggunakan model *deep embedded clustering* pada data *Yahoo! Answers*. Pada Tabel 4.27 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.4024 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 18.14% menjadi 0.4754. Metrik NMI yang awalnya rata-rata sebesar 0.2176 pada metode TFIDF mengalami peningkatan signifikan sebesar 33.59% menjadi 0.2907. Terakhir, metrik ARI yang awalnya rata-rata bernilai 0.1621 pada metode TFIDF mengalami peningkatan signifikan sebesar 44.29% menjadi 0.2339.

Tabel 4.27. Hasil evaluasi *deep embedded clustering* pada data *Yahoo! Answers*

Dataset	<i>Yahoo! Answers</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.4024±0.0282	0.2176±0.0154	0.1621±0.0202
BERT + Max + I	0.1061±0.0143	0.003±0.0074	0.0015±0.0038
BERT + Max + LN	0.3969±0.0186	0.2301±0.0143	0.1761±0.0133
BERT + Max + N	0.1±0	0±0	0±0
BERT + Max + MM	0.1713±0.0708	0.0539±0.0638	0.0312±0.0397
BERT + Mean + I	0.4661±0.0282	0.286±0.0121	0.2317±0.0193
BERT + Mean + LN	0.4754±0.0266	0.2907±0.0119	0.2339±0.0172
BERT + Mean + N	0.427±0.0292	0.2613±0.013	0.1992±0.0172
BERT + Mean + MM	0.1±0	0.0001±0	0±0

Tabel 4.28 menunjukkan hasil simulasi *text clustering* menggunakan model *deep embedded clustering* pada data *R2*. Pada Tabel 4.28 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode TFIDF dengan normalisasi. Metrik ACC, NMI, dan ARI dari representasi data teks metode BERT dengan performa terbaik secara berturut-turut lebih rendah 0.664%, 0.118%, dan 3.19% apabila dibandingkan dengan metrik representasi data metode TFIDF. Hasil ini menunjukkan bahwa representasi data teks dengan menggunakan metode BERT belum mengungguli TFIDF dengan normalisasi.

Tabel 4.28. Hasil evaluasi *deep embedded clustering* pada data *R2*

Dataset	Reuters (<i>R2</i>)		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.859±0.0100	0.5064±0.0205	0.5158±0.0288
BERT + Max + I	0.793±0.0794	0.386±0.1525	0.3545±0.1835
BERT + Max + LN	0.8409±0.0188	0.4827±0.0308	0.466±0.0480
BERT + Max + N	0.8474±0.0033	0.4996±0.0078	0.4825±0.0092
BERT + Max + MM	0.7816±0.0590	0.3727±0.1332	0.3269±0.1348
BERT + Mean + I	0.8497±0.0025	0.504±0.0068	0.4891±0.0070
BERT + Mean + LN	0.8494±0.0017	0.5035±0.0059	0.4882±0.0047
BERT + Mean + N	0.8533±0.0045	0.4996±0.0059	0.4993±0.0128
BERT + Mean + MM	0.6373±0	0.00002±0.0001	0.00001±0.00009

4.7.4. Performa pada Model *Improved Deep Embedded Clustering*

Tabel 4.29 menunjukkan hasil simulasi *text clustering* menggunakan model *improved deep embedded clustering* pada data *AG news*. Pada Tabel 4.29 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *standard normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.7453 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan sebesar 7.59% menjadi 0.8019. Metrik NMI yang awalnya rata-rata sebesar 0.4251 pada metode TFIDF mengalami peningkatan signifikan

sebesar 26.62% menjadi 0.5383. Terakhir, metrik ARI yang awalnya rata-rata bernilai 0.4571 pada metode TFIDF mengalami peningkatan signifikan sebesar 24.44% menjadi 0.5688.

Tabel 4.29. Hasil evaluasi *improved deep embedded clustering* pada data *AG news*

Dataset	<i>AG news</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.7453±0.0243	0.4251±0.0244	0.4571±0.0315
BERT + Max + I	0.376±0.1413	0.1467±0.1565	0.1253±0.1457
BERT + Max + LN	0.7819±0.0411	0.5131±0.0294	0.5394±0.0428
BERT + Max + N	0.3618±0.1478	0.1163±0.1511	0.1072±0.1408
BERT + Max + MM	0.4077±0.111	0.1157±0.1269	0.1093±0.1222
BERT + Mean + I	0.7836±0.0509	0.5296±0.0353	0.5544±0.0511
BERT + Mean + LN	0.782±0.0541	0.5297±0.0398	0.5524±0.0548
BERT + Mean + N	0.8019±0.0330	0.5383±0.0217	0.5688±0.0312
BERT + Mean + MM	0.2616±0.0208	0.0165±0.0184	0.0026±0.0063

Tabel 4.30 menunjukkan hasil simulasi *text clustering* menggunakan model *improved deep embedded clustering* pada data *Yahoo! Answers*. Pada Tabel 4.30 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode BERT dengan menggunakan metode *mean pooling* untuk ekstraksi fitur serta metode *layer normalization* untuk normalisasi fitur. Performa terbaik ini meningkat apabila dibandingkan dengan metode TFIDF yang menjadi *baseline* pada penelitian ini. Metrik ACC yang semula rata-rata sebesar 0.3975 ketika menggunakan representasi data teks dengan menggunakan metode TFIDF mengalami peningkatan signifikan sebesar 22.52% menjadi 0.487. Metrik NMI yang awalnya rata-rata sebesar 0.2243 pada metode TFIDF mengalami peningkatan signifikan sebesar 34.59% menjadi 0.3019. Terakhir, metrik ARI yang awalnya rata-rata bernilai 0.1474 pada metode TFIDF mengalami peningkatan signifikan sebesar 67.57% menjadi 0.247.

Tabel 4.30. Hasil evaluasi *improved deep embedded clustering* pada data *Yahoo!**Answers*

Dataset	<i>Yahoo! Answers</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.3975±0.0235	0.2243±0.0109	0.1474±0.0111
BERT + Max + I	0.1326±0.0354	0.0225±0.0241	0.0135±0.0158
BERT + Max + LN	0.4058±0.0182	0.2394±0.0129	0.1881±0.0131
BERT + Max + N	0.1242±0.0342	0.0193±0.0275	0.0097±0.0144
BERT + Max + MM	0.1694±0.0511	0.0504±0.0497	0.0278±0.0301
BERT + Mean + I	0.477±0.0294	0.2988±0.0126	0.2445±0.0199
BERT + Mean + LN	0.487±0.0258	0.3019±0.0118	0.247±0.0167
BERT + Mean + N	0.4308±0.0303	0.2687±0.0134	0.2078±0.0170
BERT + Mean + MM	0.1015±0.0029	0.0081±0.005	7E-05±0.0004

Tabel 4.31 menunjukkan hasil simulasi *text clustering* menggunakan model *improved deep embedded clustering* pada data R2. Pada Tabel 4.31 dapat dilihat bahwa evaluasi metrik terbaik untuk ACC, NMI, maupun ARI ditunjukkan ketika model menggunakan representasi data teks dari metode TFIDF dengan normalisasi. Metrik ACC, NMI, dan ARI terbaik dari representasi data teks metode BERT secara berturut-turut lebih rendah 1.57%, 2.74%, dan 7.37% apabila dibandingkan dengan metrik dari representasi data teks metode TFIDF. Hasil ini menunjukkan bahwa representasi data teks dengan menggunakan metode BERT belum mengungguli TFIDF dengan normalisasi.

Tabel 4.31. Hasil evaluasi *improved deep embedded clustering* pada data R2

Dataset	<i>Reuters (R2)</i>		
Metode/Metrik	ACC	NMI	ARI
TFIDF	0.8654±0.0116	0.5213±0.0252	0.5345±0.0342
BERT + Max + I	0.8095±0.0616	0.4303±0.1373	0.3917±0.1442
BERT + Max + LN	0.8401±0.0228	0.485±0.0349	0.4643±0.0572
BERT + Max + N	0.8428±0.0297	0.4889±0.0704	0.4718±0.0686
BERT + Max + MM	0.7815±0.0588	0.3623±0.1399	0.3255±0.1366
BERT + Mean + I	0.8494±0.0011	0.507±0.0049	0.4881±0.0032
BERT + Mean + LN	0.8494±0.0007	0.5045±0.0048	0.4884±0.0021

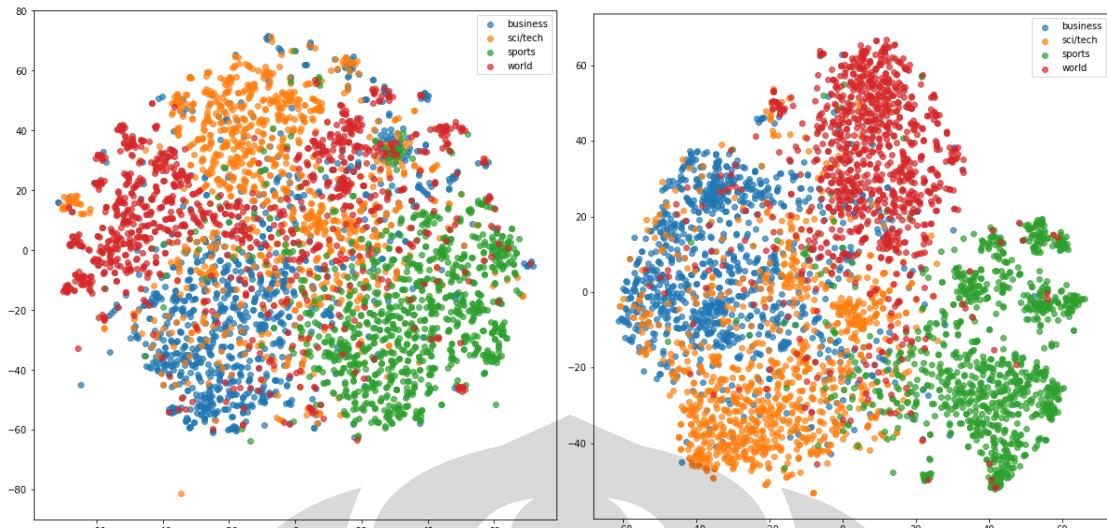
BERT + Mean + N	0.8518±0.0038	0.4952±0.0068	0.4951±0.0108
BERT + Mean + MM	0.6374±0.0002	0.0007±0.0014	0.0004±0.0007

Pada Subbab 4.7.5 dilakukan pembahasan lebih lanjut mengenai hasil simulasi yang diperoleh pada Subbab 4.7.1 sampai 4.7.4.

4.7.5. Analisis Perbandingan Representasi Data Teks Antara Metode TFIDF Dengan Metode BERT

Hasil yang diperoleh pada Subbab 4.7.1 sampai 4.7.4 menunjukkan bahwa secara keseluruhan, metode representasi data teks dengan metode BERT mengungguli performa representasi data teks dengan metode TFIDF pada 28 dari 36 metrik. Selain itu pada 8 dari 36 metrik lainnya, performa metode representasi data teks dari metode BERT juga masih mampu bersaing dengan representasi data teks dari metode TFIDF. Representasi data teks dari metode BERT secara rata-rata mengalami penurunan sebesar 1.117% pada metrik ACC, 0.953% pada metrik NMI, dan 3.52% pada metrics ARI. Penurunan ini tidak sebanding dengan peningkatan yang dialami pada 28 metrik lainnya. Salah satu alasan dari unggulnya performa representasi data teks dengan model BERT adalah karena model BERT dapat menghasilkan representasi yang menempatkan teks-teks yang similar pada posisi yang berdekatan dan jarak Euclid antara dua buah fitur teks dapat merepresentasikan relasi semantik antara keduanya. Untuk menunjukkan hal ini, dilakukan visualisasi representasi data teks dengan metode TFIDF dan metode BERT menggunakan t-SNE. Pada subbab ini akan ditunjukkan visualisasi antara 2 representasi data teks pada dua situasi, yaitu ketika metode BERT mengungguli metode TFIDF dan ketika sebaliknya. Untuk selanjutnya, representasi data teks yang digunakan pada visualisasi t-SNE adalah TFIDF dengan normalisasi dan model BERT dengan *mean pooling* dan *standard normalization*.

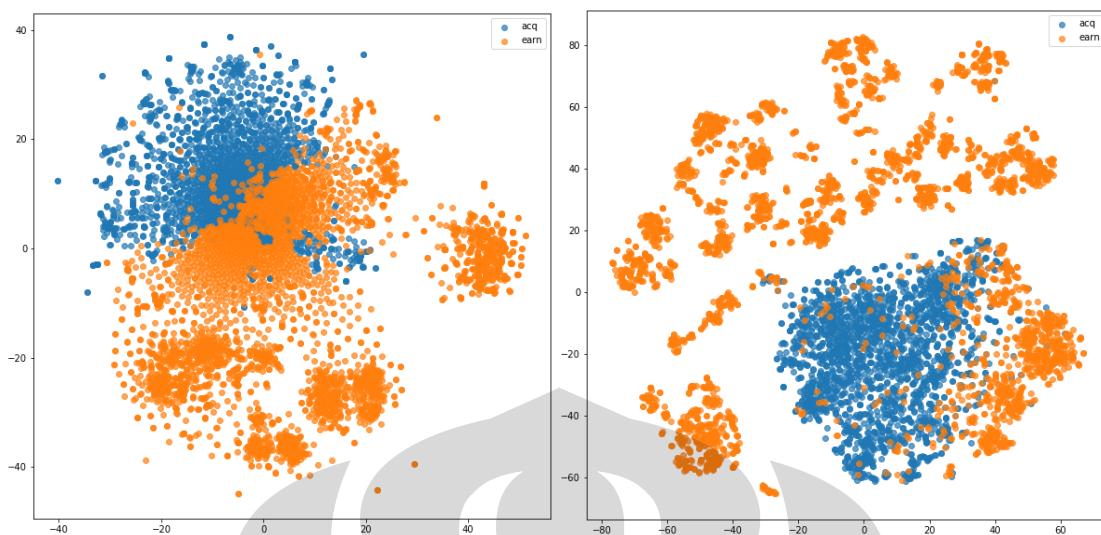
Pertama ditunjukkan visualisasi dari situasi ketika metode BERT mengungguli metode TFIDF. Visualisasi *ground-truth label* dari *AG news* menggunakan representasi data teks dapat dilihat pada Gambar 4.4.



Gambar 4.4. Visualisasi t-SNE *ground-truth label AG news* dengan representasi data teks (a) TFIDF (b) BERT

Dapat dilihat bahwa pada Gambar 4.4a kelompok antara kelas yang berbeda kurang dapat dibedakan dengan baik. Kelas *world* yang ditandai dengan warna merah, kelas *sci/tech* yang ditandai dengan warna jingga, dan kelas *business* yang ditandai dengan warna biru saling bercampur, sementara kelas *sports* yang ditandai dengan warna hijau cukup dapat dipisahkan dibandingkan kelas yang lain. Hal ini berbeda dengan Gambar 4.4b dimana kelas *world* dan *sports* secara garis besar dapat dibedakan dan secara garis besar *cluster* antara keempat kelas juga dapat dibedakan. Hal ini sejalan dengan performa model *text clustering* pada *AG news* yang ditunjukkan pada Tabel 4.26 dimana metode BERT dengan *mean normalization* dan *standard normalization* lebih baik dibandingkan metode TFIDF.

Selanjutnya ditunjukkan visualisasi dari situasi ketika metode TFIDF mengungguli metode BERT. Visualisasi *ground truth label* pada data *R2* dengan t-SNE diberikan pada Gambar 4.5. Pada Gambar 4.5a dapat dilihat bahwa di tengah terdapat daerah dimana anggota kelas *acq* dan anggota kelas *earn* saling bercampur. Sementara pada Gambar 4.5b dapat dilihat bahwa elemen kelas *earn*, ditandai dengan warna jingga, yang berada di sisi kanan bawah terpisah dari elemen kelas *earn* lainnya. Elemen kelas *earn* yang terpisah juga memiliki posisi yang lebih dekat dengan kelas *acq* yang ditandai



Gambar 4.5. Visualisasi t-SNE representasi data teks R2 dengan metode (a) TFIDF (b) model BERT

dengan warna biru. Dari kedua gambar pada Gambar 4.5, bisa dikatakan kedua representasi data teks dapat menghasilkan *cluster* yang salah ketika melakukan *text clustering*. Hal ini mengakibatkan performa kedua representasi data teks menjadi cukup similar. Performa kedua representasi dapat dilihat pada Tabel 4.28 dimana metode TFIDF dilanjutkan normalisasi memiliki performa yang sedikit lebih baik dibandingkan metode BERT dengan *mean pooling* dan *standard normalization*.

4.7.6. Analisis Perbandingan Metode Ekstraksi dan Normalisasi Fitur yang Berbeda pada Representasi Data Teks Dengan Model BERT

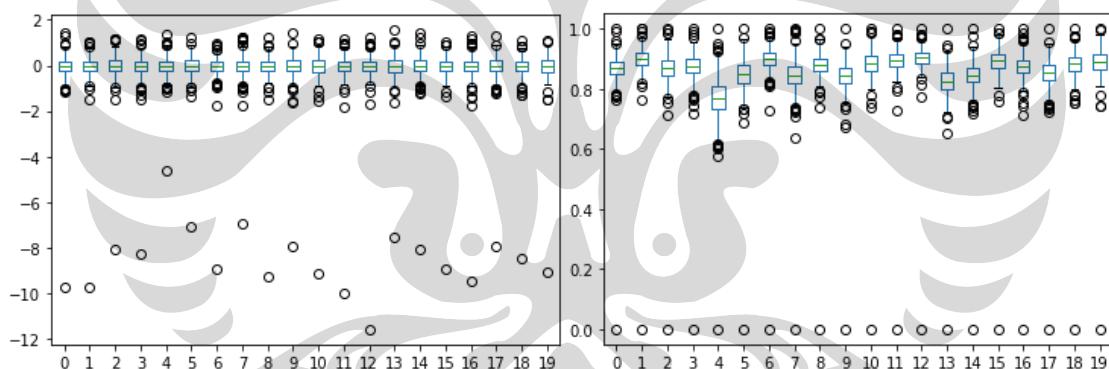
Ekstraksi dan normalisasi fitur pada metode BERT dapat dibagi menjadi 2 berdasarkan model *text clustering* yang digunakan. Pembagian tersebut tersusun atas model KM dengan EFCM dan model DEC dengan IDEC. Total metrik yang ditinjau pada masing-masing bagian adalah 18 metrik.

Pada model KM dan EFCM, kombinasi *max pooling* dan *layer normalization* memiliki performa terbaik pada 10 dari 18 metrik, dan 8 dari 18 metrik lainnya masih dapat menyaingi metrik yang dihasilkan kombinasi metode ekstraksi dan normalisasi lainnya meskipun bukan merupakan performa terbaik. Di sisi lain pada model DEC dan IDEC, kombinasi *mean pooling* dan *standard normalization* memiliki performa terbaik pada 10 dari 18 metrik, dan 8 dari 18 metrik lainnya masih dapat menyaingi metrik yang dihasilkan kombinasi metode ekstraksi dan normalisasi lainnya meskipun bukan

merupakan performa terbaik. Hasil ini menunjukkan bahwa ekstraksi dan normalisasi fitur yang paling baik bergantung pada model *text clustering* yang digunakan.

Hasil lain yang juga diperoleh adalah performa terendah untuk semua 36 metrik didapatkan ketika menggunakan representasi data teks metode BERT dengan *mean pooling* yang dilanjutkan *min-max normalization*. Performa terendah yang dihasilkan juga tidak mengungguli metode TFIDF yang menjadi *baseline* pada penelitian ini.

Salah satu penyebab dari performa yang kurang baik ini adalah proses *min-max normalization* yang rentan terhadap *outlier*. Apabila di dalam representasi data teks terdapat nilai yang jauh lebih tinggi atau jauh lebih rendah dibandingkan nilai-nilai lainnya, maka hasil *min-max normalization* tetap memiliki *outlier* tersebut dan dengan distribusi yang juga sama dengan sebelumnya. Hal ini sejalan dengan kondisi representasi data teks dengan metode BERT dilanjutkan dengan *mean pooling*. Representasi data teks ini memiliki nilai yang jauh lebih kecil dibandingkan nilai lainnya pada setiap dokumen. Hal ini dapat dilihat pada diagram kotak Gambar 4.6.



Gambar 4.6. Diagram kotak 20 representasi data *AG news* dengan metode BERT dan *mean pooling* (a) sebelum *min-max normalization* (b) sesudah *min-max normalization*.

Selain itu, nilai *outlier* yang ditunjukkan pada Gambar 4.6 merupakan elemen dengan posisi yang sama pada setiap vektor representasi data *AG news*. Dengan mentransformasi nilai *outlier* tersebut menjadi 0, setiap elemen vektor representasi *AG news* dengan posisi tersebut juga akan bernilai 0. Hal ini menghilangkan informasi yang mungkin terkandung di dalam elemen pada posisi tersebut.

BAB 5

PENUTUP

5.1. Kesimpulan

Implementasi model *bidirectional encoder representation from transformer* (BERT) sebagai representasi data teks dalam *text clustering* telah dilakukan pada Bab 4. Berdasarkan perolehan dari Bab 4, dapat diperoleh kesimpulan sebagai berikut:

1. Berdasarkan Tabel 4.20 – 4.31, metode BERT mampu mengungguli performa metode TFIDF sebagai representasi data teks dalam *text clustering* pada 28 dari 36 metrik yang diselidiki.
2. Berdasarkan Tabel 4.20 – 4.31, metode ekstraksi fitur dan normalisasi fitur yang diterapkan pada metode BERT menghasilkan performa yang berbeda-beda dan bergantung pada model *text clustering* yang digunakan.
 - a. Metode BERT dengan *max pooling* dan *layer normalization* mengungguli ekstraksi dan normalisasi fitur lainnya pada 10 dari 18 metrik yang diselidiki di model *k-means clustering* dan *eigenspace-based fuzzy c-means*.
 - b. Metode BERT dengan *mean pooling* dan *standard normalization* mengungguli ekstraksi dan normalisasi fitur lainnya pada 10 dari 18 metrik yang diselidiki di model *deep embedded clustering* dan *improved deep embedded clustering*.

5.2. Saran

Beberapa hal yang dapat menjadi pertimbangan untuk penelitian lebih lanjut mengenai analisis kinerja model BERT sebagai representasi data teks untuk *text clustering* adalah sebagai berikut:

1. Menganalisis kinerja representasi data teks pada algoritma *text clustering* pada data teks yang berbeda.
2. Mengimplementasikan representasi data teks dengan metode BERT pada algoritma *text clustering* yang berbeda.
3. Mencoba teknik ekstraksi fitur tetap yang berbeda dari model BERT, misalkan mengambil representasi data teks dari *output* yang diberikan lapisan *transformer encoder* terakhir (lapisan ke-12). Contoh teknik lain adalah mengambil

representasi data teks berupa penggabungan *output* yang diberikan 4 lapisan *transformer encoder* terakhir (lapisan ke-9 sampai 12), atau penjumlahan *output* dari semua lapisan *transformer encoder*.

4. Membandingkan dengan metode representasi data teks lainnya sebagai *baseline*. Beberapa metode yang dapat dicoba adalah word2vec, GloVe, fastText, dan ELMo.



DAFTAR PUSTAKA

- Aggarwal, C. C., & Zhai, C. (2012). A survey of text clustering algorithms. *Mining text data*, 77-128. doi: 10.1007/978-1-4614-3223-4_4
- Akgün, E., & Demir, M. (2018). Modeling course achievements of elementary education teacher candidates with artificial neural networks. *International Journal of Assessment Tools in Education*, 5(3), 491-509.
- Alatas, H., Murfi, H., & Bustamam, A. (2018). Topic detection using fuzzy c-means with nonnegative double singular value decomposition initialization. *International Journal of Advances in Soft Computing and its Applications*, 10(2), 206-222.
- Alammar, J. (2018). The illustrated transformer. <http://jalammar.github.io/illustrated-transformer/>. Diakses: Januari 2021
- Anton, H., & Rorres, C. (2013). *Elementary linear algebra: applications version*. John Wiley & Sons.
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint*, arXiv:1607.06450.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint*, arXiv:1409.0473.
- Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1), 3-31.
- Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3), 191-203.
- Bezdek, J. C., Hall, L., & Clarke, L. P. (1993). Review of MR image segmentation techniques using pattern recognition. *Medical Physics*, 20(4), 1033-1048.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer-Verlag New York
- Burden, R. L., & Faires, J. D. (2011). *Numerical analysis (9th ed)*. Boston, USA: Brooks/Cole Cengage Learning
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint*, arXiv:1406.1078.

- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, arXiv:1810.04805.
- d'Sa, A. G., Illina, I., & Fohr, D. (2020, February). Bert and fasttext embeddings for automatic detection of toxic speech. In *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)* (pp. 1-5). IEEE.
- Dams, T. (2021). Coronavirus spurs rise of digital news consumption. <https://www.ibc.org/news/coronavirus-spurs-rise-of-digital-news-consumption/6093.article>. Diakses: 1 Juli 2021
- Du, K.-L., & Swamy, M. N. (2013). *Neural networks and statistical learning*. Springer Science & Business Media.
- Feng, Q., Chen, L., Chen, C. P., & Guo, L. (2020). Deep fuzzy clustering – a representation learning approach. In *IEEE Transactions on Fuzzy Systems* (pp. 1- 1). IEEE
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Guan, R., Zhang, H., Liang, Y., Giunchiglia, F., Huang, L., & Feng, X. (2020). Deep feature-based text clustering and its explanation. *IEEE Transactions on Knowledge and Data Engineering*.
- Guo, X., Gao, L., Liu, X., & Yin, J. (2017, August). Improved Deep Embedded Clustering with Local Structure Preservation. In *IJCAI* (pp. 1753-1759).
- Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504-507.
- Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1), 193-218.
- Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

- Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8), 651-666.
- Jernite, Y., Bowman, S. R., & Sontag, D. (2017). Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint*, arXiv:1705.00557.
- Kaliyar, R. K. (2020, January). A Multi-layer Bidirectional Transformer Encoder for Pre-trained Word Embedding: A Survey of BERT. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 336-340). IEEE.
- Lionbridge (2021). What are transformer models in machine learning? <https://lionbridge.ai/articles/what-are-transformer-models-in-machine-learning/>. Diakses 1 Mei 2021
- Logeswaran, L., & Lee, H. (2018). An efficient framework for learning sentence representations. *arXiv preprint*, arXiv:1803.02893.
- Maji, P., Roy, A. R., & Biswas, R. (2002). An application of soft sets in a decision making problem. *Computers & Mathematics with Applications*, 44(8-9), 1077-1083.
- Murfi H. (2018) The accuracy of fuzzy c-means in lower-dimensional space for topic detection. In: *Qiu M. (eds) Smart Computing and Communication. Lecture Notes in Computer Science*, vol 11344. Springer, Cham. https://doi.org/10.1007/978-3-030-05755-8_32
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12, 2825-2830.
- Peng, X., Xiao, S., Feng, J., Yau, W. Y., & Yi, Z. (2016, July). Deep subspace clustering with sparsity prior. In *IJCAI* (pp. 1925-1931).
- Pradana, R.C. (2020). *Deep Embedded Clustering* untuk Pendeksi Topik Tweet Berita Berbahasa Indonesia. Skripsi. Tidak Diterbitkan. Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Indonesia: Depok.
- Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).
- Roser, M., Ritchie, H., & Ortiz-Ospina, E. (2015). Internet. <https://ourworldindata.org/internet> . Diakses: Juli 2021.

- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint*, arXiv:1609.04747.
- Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In New directions in statistical physics. In Wille L.T. (eds) *New Directions in Statistical Physics*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-08968-2_16
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks*. Paper presented at the Advances in neural information processing systems.
- Uppada, S. K. (2014). Centroid based clustering algorithms—A clarion study. *International Journal of Computer Science and Information Technologies*, 5(6), 7309-7313.
- Vanneschi, L., & Castelli, M. (2019). Multilayer perceptron. *Encyclopedia of Bioinformatics and Computational Biology*, 1, 612-620
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *arXiv preprint*, arXiv:1706.03762.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12).
- Winkler, R., Klawonn, F., & Kruse, R. (2011). Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications (IJFSA)*, 1(1), 1-16.
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., . . . Macherey, K. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint*, arXiv:1609.08144.
- Xie, J., Girshick, R., & Farhadi, A. (2016, June). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478-487). PMLR.
- Xiong, C., Hua, Z., Lv, K., & Li, X. (2016). An improved k-means text clustering algorithm by optimizing initial cluster centers. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* (pp. 265-268). IEEE.
- Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 267-273).

- Ye, Z., Jiang, G., Liu, Y., Li, Z., & Yuan, J. (2020). Document and word representations generated by graph convolutional network and BERT for short text classification. In *ECAI 2020* (pp. 2275-2281). IOS Press.
- Yeung, K. Y., & Ruzzo, W. L. (2001). Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9), 763-774.
- Yu, Q., Wang, Z., & Jiang, K. (2021). Research on text classification based on BERT-BiGRU model. In *Journal of Physics: Conference Series* (Vol. 1746, No. 1, p. 012019). IOP Publishing.
- Yusdiansyah, M. R., Murfi, H., & Wibowo, A. (2019). Randomspace-based fuzzy c-means for topic detection on Indonesia online news. In *International Conference on Multi-disciplinary Trends in Artificial Intelligence* (pp. 133-143). Springer, Cham.
- Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision* (pp. 19-27).



Lampiran 1. Fungsi Pengambilan Representasi Teks dengan Model BERT

Fungsi Representasi Data Teks dengan BERT

```

from bert_serving.client import BertClient
import numpy as np
import pandas as pd

%tensorflow_version 1.x

# Change version of tensorflow in order to use bert-as-service
!pip uninstall tensorflow-gpu==1.12.0
!pip install tensorflow==1.15.0

# Install bert-as-service
!pip install bert-serving-client
!pip install -U bert-serving-server[http]

!wget https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip
!unzip uncased_L-12_H-768_A-12.zip
!nohup bert-serving-start -pooling_strategy=None -model_dir=../uncased_L-12_H-768_A-12 > out.file 2>&1 &

# BERT Encoder
bc = BertClient()

def encode_sentences(sents):
    data_size = len(sents)
    all_emb = bc.encode(sents)
    result = []
    for i in range(data_size):
        emb = all_emb[i]
        emb = emb[np.sum(emb > 0, axis=-1) > 0]
        result.append(emb)
    return result

# Feature Extraction
def feat_extraction(sents, batch_size=32):
    data_size = len(sents)
    feat_max = []
    feat_mean = []
    feat_last = []
    for i in range(0, data_size, batch_size):
        batch_sents = sents[i: i + batch_size]
        # batch_feat_lst = encoder.embed_batch([s.split() for s in batch_sents])
        batch_feat_lst = encode_sentences(batch_sents)
        feat_max.extend([np.max(tmp, axis=0) for tmp in batch_feat_lst])
        feat_mean.extend([np.mean(tmp, axis=0) for tmp in batch_feat_lst])
        feat_last.extend([tmp[0] for tmp in batch_feat_lst])
        print(i)
    return np.stack(feat_max), np.stack(feat_mean), np.stack(feat_last)

# Feature Normalization
def ln(feat):
    return (feat - feat.mean(axis=1, keepdims=True)) / feat.std(axis=1, keepdims=True)

def norm(feat):
    return feat / np.linalg.norm(feat, axis=1, keepdims=True)

def minmax(feat):
    return (feat - feat.min(axis=1, keepdims=True)) / (feat.max(axis=1, keepdims=True) - feat.min(axis=1, keepdims=True))

labels = np.array(y_new)

feat_max, feat_mean, feat_last = feat_extraction(X_new)

```

Fungsi Representasi Data Teks dengan TFIDF

```

import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk

nltk.download('stopwords')

```

```
def get_stop_words():
    from nltk.corpus import stopwords
    stop_words = stopwords.words('english')
    return stop_words

def get_tf_idf_feat(sents, max_features=2000, norm=False):
    stop_words = get_stop_words()
    tfidf = TfidfVectorizer(max_features=max_features, stop_words=stop_words)
    feat = tfidf.fit_transform(sents)
    feat = feat.toarray()
    if norm==True:
        feat = feat*np.sqrt(feat.shape[1])
    return feat

feat = get_tf_idf_feat(X_new)
feat_norm = get_tf_idf_feat(X_new, norm=True)
```



Lampiran 2. Program *Text Clustering*

```

import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import normalized_mutual_info_score
from sklearn.metrics import adjusted_mutual_info_score
from coclust.evaluation.external import accuracy as cluster_acc
from skfuzzy.cluster import cmeans
from sklearn.decomposition import TruncatedSVD
from time import time

# K-Means
def cluster_alg(feat, n_clusters):
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, n_jobs=10, verbose=True)
    pred = kmeans.fit_predict(feat)
    return pred

# EFCM
svd = TruncatedSVD(n_components = 5)

def cmeans_alg(feat, n_clusters):
    cntr, u, ub, d, jm, p, fpc = cmeans(
        svd.fit_transform(feat).T, c=n_clusters, m=1.1, error=0.0001, maxiter=200, init=None)
    cluster_membership = np.argmax(u, axis=0)
    return cluster_membership

```

K-Means Clustering dengan representasi data teks dari model BERT

```

logfile = open('/content/' + 'bert_km.csv', 'a')
logwriter = csv.DictWriter(logfile, fieldnames=['model', 'acc', 'nmi', 'ari'])
logwriter.writeheader()

logwriter.writerow(dict(model='yahoo_answers', acc='', nmi='', ari=''))

# data bertencode1
feat_func_dict = {'ln': ln, 'n': norm, 'mm': minmax, 'mm1': minmax1, 'i': lambda x: x}
all_feat = {'bert_max':feat_max, 'bert_mean':feat_mean}
trial_num= 50
for feat_name, feat in all_feat.items():
    for func_name, feat_trans_func in feat_func_dict.items():
        best_acc = 0.0
        best_pred = None
        feat_tmp = feat_trans_func(feat)
        all_pred, all_acc, all_nmi, all_ari = [], [], [], []
        tmp_feat_name = feat_name + '_{}'.format(func_name)
        logwriter.writerow(dict(model=tmp_feat_name, acc='', nmi='', ari=''))
        for i in range(trial_num):
            pred = cluster_alg(feat_tmp, n_clusters=10)
            acc_i = cluster_acc(labels, pred)
            nmi_i = normalized_mutual_info_score(labels, pred)
            ari_i = adjusted_mutual_info_score(labels, pred)
            all_pred.append(pred.tolist())
            logwriter.writerow(dict(model=i+1, acc=acc_i, nmi=nmi_i, ari=ari_i))
            all_acc.append(acc_i)
            all_nmi.append(nmi_i)
            all_ari.append(ari_i)
            if acc_i > best_acc:
                best_pred = pred
                best_acc = acc_i
        print(np.mean(all_acc))
        print(np.mean(all_nmi))
        print(np.mean(all_ari))
        logwriter.writerow(dict(model='mean', acc=np.mean(all_acc), nmi=np.mean(all_nmi), ari=np.mean(all_ari)))

logfile.close()

```

EFCM dengan representasi data teks dari model BERT

```

logfile = open('/content/' + 'bert_efcm.csv', 'a')
logwriter = csv.DictWriter(logfile, fieldnames=['model', 'acc', 'nmi', 'ari'])
logwriter.writeheader()

logwriter.writerow(dict(model='yahoo_answers', acc='', nmi='', ari=''))

trial_num= 50
feat_func_dict = {'ln': ln, 'n': norm, 'mm': minmax, 'mm1': minmax1, 'i': lambda x: x}
all_feat = {'bert_max':feat_max, 'bert_mean':feat_mean}

for feat_name, feat in all_feat.items():
    for func_name, feat_trans_func in feat_func_dict.items():
        best_acc = 0.0
        best_pred = None
        feat_tmp = feat_trans_func(feat)
        all_pred, all_acc, all_nmi, all_ari = [], [], [], []
        tmp_feat_name = feat_name + '_{}'.format(func_name)
        logwriter.writerow(dict(model= tmp_feat_name, acc='', nmi='', ari=''))
        for i in range(trial_num):
            pred = cmeans_alg(feat_tmp, n_clusters=10)
            acc_i = cluster_acc(labels, pred)
            nmi_i = normalized_mutual_info_score(labels, pred)
            ari_i = adjusted_mutual_info_score(labels, pred)
            all_pred.append(pred.tolist())
            logwriter.writerow(dict(model=i+1, acc=acc_i, nmi=nmi_i, ari=ari_i))
            all_acc.append(acc_i)
            all_nmi.append(nmi_i)
            all_ari.append(ari_i)
            if acc_i > best_acc:
                best_pred = pred
                best_acc = acc_i
        print(np.mean(all_acc))
        print(np.mean(all_nmi))
        print(np.mean(all_ari))
        logwriter.writerow(dict(model='mean', acc=np.mean(all_acc), nmi=np.mean(all_nmi), ari=np.mean(all_ari)))

logfile.close()

```

K-Means Clustering dengan representasi data teks dari model TFIDF

```

logfile = open('/content/' + 'tfidf_km.csv', 'a')
logwriter = csv.DictWriter(logfile, fieldnames=['model', 'acc', 'nmi', 'ari'])
logwriter.writeheader()

logwriter.writerow(dict(model='yahoo_answers', acc='', nmi='', ari=''))

trial_num= 50
feat_name = 'tfidf'
best_acc = 0.0
best_pred = None
feat_tmp = feat
all_pred, all_acc, all_nmi, all_ari = [], [], [], []
tmp_feat_name = feat_name + '_{}'.format(func_name)
logwriter.writerow(dict(model= tmp_feat_name, acc='', nmi='', ari=''))

for i in range(trial_num):
    pred = cluster_alg(feat_tmp, n_clusters=10)
    acc_i = cluster_acc(labels, pred)
    nmi_i = normalized_mutual_info_score(labels, pred)
    ari_i = adjusted_mutual_info_score(labels, pred)
    all_pred.append(pred.tolist())
    logwriter.writerow(dict(model=i+1, acc=acc_i, nmi=nmi_i, ari=ari_i))
    all_acc.append(acc_i)
    all_nmi.append(nmi_i)
    all_ari.append(ari_i)
    if acc_i > best_acc:
        best_pred = pred
        best_acc = acc_i

print(np.mean(all_acc))
print(np.mean(all_nmi))
print(np.mean(all_ari))
logwriter.writerow(dict(model='mean', acc=np.mean(all_acc), nmi=np.mean(all_nmi), ari=np.mean(all_ari)))

logfile.close()

```

EFCM dengan representasi data teks dari model TFIDF

```

logfile = open('/content/' + 'tfidf_efcm.csv', 'a')
logwriter = csv.DictWriter(logfile, fieldnames=['model', 'acc', 'nmi', 'ari'])
logwriter.writeheader()

logwriter.writerow(dict(model='yahoo_answers', acc='', nmi='', ari=''))

trial_num= 50
feat_name = 'tfidf'
best_acc = 0.0
best_pred = None
feat_tmp = feat

all_pred = []
all_acc = []
all_nmi = []
all_ari = []

# tmp_feat_name = feat_name + '{}'.format(func_name)
logwriter.writerow(dict(model=feat_name, acc='', nmi='', ari=''))

for i in range(trial_num):
    pred = cmeans_alg(feat_tmp, n_clusters=10)
    acc_i = cluster_acc(labels, pred)
    nmi_i = normalized_mutual_info_score(labels, pred)
    ari_i = adjusted_mutual_info_score(labels, pred)
    all_pred.append(pred.tolist())
    logwriter.writerow(dict(model=i+1, acc=acc_i, nmi=nmi_i, ari=ari_i))
    all_acc.append(acc_i)
    all_nmi.append(nmi_i)
    all_ari.append(ari_i)
    if acc_i > best_acc:
        best_pred = pred
        best_acc = acc_i

print(np.mean(all_acc))
print(np.mean(all_nmi))
print(np.mean(all_ari))
logwriter.writerow(dict(model='mean', acc=np.mean(all_acc), nmi=np.mean(all_nmi), ari=np.mean(all_ari)))

logfile.close()

```

```

from DEC import DEC
import os, csv
from keras.optimizers import SGD
from keras.initializers import VarianceScaling
import numpy as np

from IDEC import IDEC

```

Text Clustering dengan DEC dan IDEC

```

def DEC_IDEC(main_path, csv_name, inner_folder_name, feat_tmp, trial_min=0, trial_max=50, n_clusters):
    logfile = open(main_path + csv_name, 'a')
    logwriter = csv.DictWriter(logfile, fieldnames=['trials', 'acc', 'nmi', 'ari'])
    logwriter.writeheader()

    logwriter.writerow(dict(trials='agnews', acc='', nmi='', ari=''))
    save_db_dir = os.path.join(main_path, inner_folder_name)

    if not os.path.exists(save_db_dir):
        os.mkdir(save_db_dir)

    pretrain_optimizer = 'adam'
    # setting parameters
    update_interval = 30
    pretrain_epochs = 500
    init = VarianceScaling(scale=1. / 3., mode='fan_in',
                           distribution='uniform')

    num_trial = trial_max - trial_min

    '''Training for base and nosp'''
    results = np.zeros(shape=(num_trial, 3))
    baseline = np.zeros(shape=(num_trial, 3))
    metrics0, metrics1, metrics2, metrics3 = [], [], [], []

```

```

for i in range(trial_min, trial_max): # base
    save_dir = os.path.join(save_db_dir, 'trial%d_dec' % i)
    if not os.path.exists(save_dir):
        os.mkdir(save_dir)

    dec = DEC(dims=[feat_tmp.shape[-1], 500, 500, 2000, 10], n_clusters=n_clusters, init=init)
    notes = []

    dec.pretrain(x=feat_tmp, y=labels, optimizer=pretrain_optimizer,
                 epochs=pretrain_epochs,
                 save_dir=save_dir)
    dec.compile(optimizer='adam', loss='kld')
    dec.fit(feat_tmp, y=labels, tol = 1e-6,
            update_interval=update_interval,
            save_dir=save_dir)

    log = open(os.path.join(save_dir, 'dec_log.csv'), 'r')
    reader = csv.DictReader(log)
    metrics = []
    for row in reader:
        metrics.append([row['acc'], row['nmi'], row['ari']])
    metrics0.append(metrics[0])
    metrics1.append(metrics[-1])
    log.close()

#####
# save_dir = os.path.join(save_db_dir, 'trial%d_idec' % i)
# if not os.path.exists(save_dir):
#     os.mkdir(save_dir)
optimizer = 'adam'

idec = IDEC(dims=[feat_tmp.shape[-1], 500, 500, 2000, 10], n_clusters= n_clusters , batch_size=256)
ae_dir = main_path + inner_folder_name + '/trial' + str(i) + '_dec/ae_weights.h5'
idec.initialize_model(ae_weights=ae_dir, gamma=0.1, optimizer=optimizer)

# begin clustering, time not include pretraining part.
t0 = time()
y_pred = idec.clustering(feat_tmp, y=labels, tol = 1e-6, update_interval= update_interval, save_dir=save_dir)

log = open(os.path.join(save_dir, 'idec_log.csv'), 'r')
reader = csv.DictReader(log)
metrics = []
for row in reader:
    metrics.append([row['acc'], row['nmi'], row['ari']])
metrics2.append(metrics[0])
metrics3.append(metrics[-1])
log.close()

print('acc:', cluster_acc(labels, y_pred))
print('clustering time: ', (time() - t0))

metrics0, metrics1 = np.asarray(metrics0, dtype=float), np.asarray(metrics1, dtype=float)
metrics2, metrics3 = np.asarray(metrics2, dtype=float), np.asarray(metrics3, dtype=float)

for t, line in enumerate(metrics0):
    logwriter.writerow(dict(trials=t, acc=line[0], nmi=line[1], ari=line[2]))
logwriter.writerow(dict(trials=' ', acc=np.mean(metrics0, 0)[0], nmi=np.mean(metrics0, 0)[1], ari=np.mean(metrics0, 0)[2]))
for t, line in enumerate(metrics1):
    logwriter.writerow(dict(trials=t, acc=line[0], nmi=line[1], ari=line[2]))
logwriter.writerow(dict(trials=' ', acc=np.mean(metrics1, 0)[0], nmi=np.mean(metrics1, 0)[1], ari=np.mean(metrics1, 0)[2]))

for t, line in enumerate(metrics2):
    logwriter.writerow(dict(trials=t, acc=line[0], nmi=line[1], ari=line[2]))
logwriter.writerow(dict(trials=' ', acc=np.mean(metrics2, 0)[0], nmi=np.mean(metrics2, 0)[1], ari=np.mean(metrics2, 0)[2]))
for t, line in enumerate(metrics3):
    logwriter.writerow(dict(trials=t, acc=line[0], nmi=line[1], ari=line[2]))
logwriter.writerow(dict(trials=' ', acc=np.mean(metrics3, 0)[0], nmi=np.mean(metrics3, 0)[1], ari=np.mean(metrics3, 0)[2]))

logfile.close()

```

Ilustrasi Penggunaan Fungsi DEC_IDEC

```
DEC_IDEC('./results-agnews/max_ln/', '/bert_max_ln.csv', 'agnews_max_ln', ln(feat_max), 0, 50, 4)
```

```
DEC_IDEC('./results-agnews/max_ln/', '/tfidf.csv', 'agnews_tfidf', feat_dec, 0, 50, 4)
```