BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Analisis sentimen telah dibahas (Lutfi dkk., 2018) dengan menerapkan metode *support vector machine* dan *naïve bayes*, data ulasan pengguna dikumpulkan dari salah satu marketplace di Indonesia, Bukalapak. Menurut (Lutfi dkk., 2018) metode *support vector machine* dengan *kernel linear* memberikan akurasi yang lebih tinggi dibandingkan *naïve bayes*. Namun menurut (M. Liu dkk., 2021) analisis sentimen tradisional hanya berfokus pada tugas tingkat kalimat atau tingkat dokumen untuk menemukan polaritas sentimen "negatif", "positif", atau "netral", tetapi polaritas sentimen terhadap aspek yang berbeda dari kalimat atau dokumen mudah diabaikan.

Analisis sentimen berbasis aspek telah dibahas (Lutfi dkk., 2018; Wahyudi dan Kusumaningrum, 2019) dengan menerapkan metode *Latent Dirichlet Allocation* (LDA) yang digunakan untuk mengetahui kinerja klasifikasi analisis sentimen pada ulasan pengguna *e-commerce* Indonesia. Menurut (Wahyudi dan Kusumaningrum, 2019) penggunaan metode LDA untuk perbandingan kinerja klasifikasi analisis sentimen pada ulasan pengguna *e-commerce* menunjukkan bahwa model dengan kombinasi penggunaan data pelatihan umum dan data pelatihan per kategori menghasilkan akurasi pengujian terbaik. Namun menurut (Ligthart dkk., 2021a) dalam menerapkan model *supervised* dibutuhkan banyak data berlabel sedangkan dalam kenyataannya untuk mendapatkan data berlabel cukup sulit.

Analisis sentimen berbasis aspek dengan menerapkan model *semi-supervised* telah dibahas (Anand dan Naorem, 2016) dengan memanfaatan data ulasan pada film. Menurut (Anand dan Naorem, 2016) tiga skema untuk pemilihan kata petunjuk aspek dieksplorasi: pelabelan manual (M), pengelompokan (C) dan tinjauan pengelompokan terpandu (RC). Ke tiga skema digunakan untuk mendeteksi aspek dan sentimen dievaluasi secara empiris terhadap set tes yang dibuat secara manual, hasilnya menunjukkan efektivitas pelabelan manual atas

pendekatan berbasis klaster yang menggunakan kata-kata petunjuk yang dipandu ulasan berkinerja lebih baik. Namun beberapa makalah dalam literatur menetapkan aturan untuk mendeteksi kata-kata aspek yang mungkin menjadi target opini berdasarkan pengetahuan akal sehat tentang bahasa alami (Poria dkk., 2014; A. Zhao dan Yu, 2021).

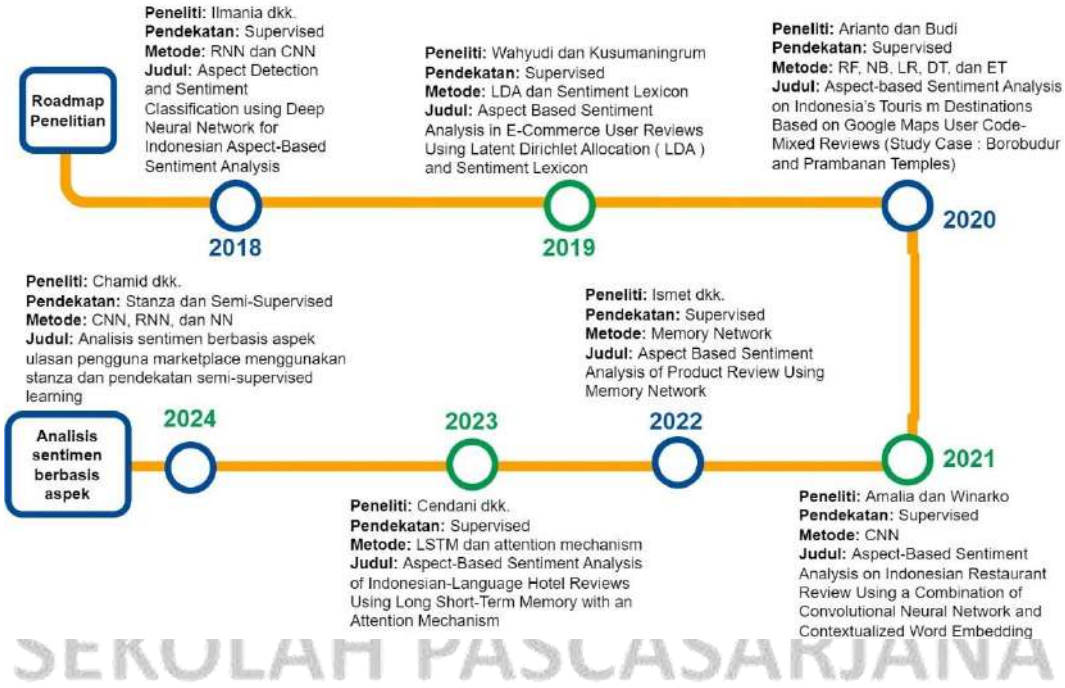
Analisis sentimen berbasis aspek dengan fokus sub-tugas *aspect mining* telah banyak dibahas dengan menerapkan beberapa model *supervised* (Poria dkk., 2016; T. Wang dkk., 2014; Y. Wang dkk., 2016), *unsupervised* (Dragoni dkk., 2019; Pathan dan Prakash, 2021; D. Zhu dkk., 2018) dan *semi-supervised* (Ansari dkk., 2020; N. Li dkk., 2020a). Sedangkan untuk sub-tugas *aspect sentiment classification* telah dibahas (N. Li dkk., 2019; X. Li dkk., 2018; Tang dkk., 2016). Namun, menurut (N. Li dkk., 2020b) sebagian besar model *semi-supervised* untuk analisis sentimen berbasis aspek hanya diusulkan untuk *aspect mining*. Sedangkan *aspect mining* dan *aspect sentiment classification* merupakan dua sub-tugas yang dapat digabungkan untuk kinerja bersama sesuai domain yang ada.

Kerangka *semi-supervised multi-task learning* (SEML) telah diterapkan (N. Li dkk., 2020b) untuk analisis sentimen berbasis aspek, SEML memiliki tiga fitur utama diantaranya 1) SEML menerapkan *Cross-View Training* (CVT) untuk mengaktifkan *semi-supervised sequence learning* atas sekumpulan kecil ulasan berlabel dan sekumpulan besar ulasan tidak berlabel, 2) SEML dapat menyelesaikan dua sub-tugas secara bersamaan dengan menggunakan tiga lapisan saraf berulang dua arah yang ditumpuk untuk mempelajari representasi ulasan, 3) SEML mengembangkan *Moving-window Attentive Gated Recurrent Unit* (MAGRU) untuk tiga lapisan saraf berulang untuk meningkatkan pembelajaran representasi dan akurasi prediksi. Namun menurut (N. Li dkk., 2020b) SEML memiliki kelemahan misalnya prediktor *aspect sentiment classification* dapat melabeli polaritas sentimen pada kata-kata non-aspek, karene SEML secara langsung memberikan representasi tersembunyi antara sub-tugas yang dapat membawa hasil *aspect mining* dan *aspect sentiment classification* yang tidak konsisten. Sehingga menurut (N. Li dkk., 2020b) SEML dapat dikembangkan dengan memanfaatkan *linguistic knowledge* untuk meningkatkan kinerja.

Model *knowledge-enabled bidirectional encoder representations from transformers* (BERT) telah diterapkan (A. Zhao dan Yu, 2021) untuk analisis sentimen berbasis aspek, dengan cara memanfaatkan informasi dari *sentiment knowledge graph* dengan memasukkan *sentiment domain knowledge* ke dalam model representasi bahasa, sehingga diperoleh entitas *embedding vectors* dalam *sentiment knowledge graph* dan kata-kata dalam teks dalam ruang vektor yang konsisten. Menurut (A. Zhao dan Yu, 2021) model *knowledge-enabled* BERT mampu mencapai kinerja yang lebih baik dengan sejumlah kecil data pelatihan dengan memasukkan pengetahuan domain eksternal ke dalam model representasi bahasa untuk mengimbangi data pelatihan yang terbatas.

2.2 *Roadmap* Penelitian

Berikut merupakan roadmap penelitian yang dapat dilihat pada Gambar 2.1.



Gambar 2.1 *Roadmap* Penelitian

Selanjutnya, ringkasan penelitian terkait topik analisis sentimen berbasis aspek berbahasa Indonesia dan Inggris seperti yang ditunjukkan pada Table 2.1.

Tabel 2.1 Ringkasan penelitian terkait

Peneliti	Motode	Dataset	Hasil
Analisis sentimen berbasis aspek berbahasa Indonesia			
Cendani dkk. (2023)	LSTM + attention mechanism	Ulasan hotel indonesia	F1 score = 0,76
(Ismet dkk., 2022)	Memory Network	Ulasan produk	Akurasi aspek = 0,83 Sentimen = 0,78
Amalia dan Winarko, (2021)	CNN	Ulasan Restoran Indonesia	Aspek (F1-score) = 0,86 Sentimen = (F1-score) = 0,91
Arianto dan Budi, (2020)	Random Forest, Naïve Bayes, Logistic Regression, Decision Tree, dan Extra Tree	Ulasan pengguna Google Maps tentang destinasi wisata Indonesia yaitu Candi Borobudur dan Prambanan	F1 score = 0,74
Ilmania dan Purwarianti (2019)	Bi-LSTM	Review dan komen terkait otomotif	Aspect (F1) = 0,94 Sentiment (Acc)= 0,90
Ilmania dkk. (2018)	RNN dan CNN	Review bahasa indonesia dari online marketplace Tokopedia	F1 score = 0,89
Aspect-based sentiment analysis berbahasa Inggris			
Kim dkk. (2022)	GCN + RNN	restaurant reviews of SemEval 2014, 2015, and 2016. laptop review of SemEval 2014. Twitter review.	F1 score = 0,77
Chakraborty (2022)	Spectral temporal GNN	Laptop and Restaurant review from SemEVal-14. Men's T-shirt and Television review.	F1 score = 0,79
K. Wang dkk. (2020a)	Relational graph attention network (R-GAT)	SemEval 2014 (domain: restaurant and laptop) and Twitter	F1 score = 0,81

Tabel 2.1 Ringkasan penelitian terkait (lanjutan)

Peneliti	Motode	Dataset	Hasil
R. Li dkk. (2021)	Dual GCN	SemEval 2014 (domain: restaurant and laptop) and Twitter	F1 score = 0,78
Liang dkk. (2022)	Sentic GCN	SemEval 2014, SemEval 2015 and SemEval 2016 (domain laptop and restaurants)	F1 score = 0,76
Usulan	Pendekatan *stanza* dan *semi-supervised learning* menggunakan *deep learning* (CNN, RNN, dan NN)	Ulasan marketplace (produk: *t-shirt*)	-

2.3 Dasar Teori

2.3.1 Pra-pemrosesan Data Teks

Pra-pemrosesan adalah langkah pertama dalam klasifikasi teks, dan memilih teknik pra-pemrosesan yang tepat dapat meningkatkan efektivitas klasifikasi (Effrosynidis dkk., 2017; Symeonidis dkk., 2018). Pra-pemrosesan dalam konteks ini adalah prosedur pembersihan dan penyiapan teks yang akan diklasifikasi.

Beberapa teknik pra-pemrosesan dapat dicoba disesuaikan studi kasus dan kebutuhan. Urutan penerapannya sangatlah penting, sehingga memungkinkan kombinasi keduanya dalam alur pra-pemrosesan yang sama. Berikut merupakan beberapa teknik pra-pemrosesan dijelaskan secara singkat masing-masing teknik, alasan penerapannya beserta contoh (Symeonidis dkk., 2018).

1) Hapus string unicode dan noise Not

Tidak semua dataset didapatkan dalam bentuk bersih. Jadi, pertama-tama, dengan menggunakan beberapa ekspresi reguler, menghapus karakter non-bahasa Inggris dan string unicode seperti “\ u002c” dan “\ x06” yang merupakan sisa dari prosedur crawling yang membuat kumpulan data. Teknik ini dianggap sebagai dasar percobaan.

2) Mengganti URL dan sebutan pengguna

Pada data teks, sebagian besar kalimat berisi URL, pengguna sebutkan, dan/atau simbol hashtag. Kehadiran mereka tidak mengandung sentimen apa pun dan salah satu pendekatannya adalah menggantinya dalam pra-pemrosesan dengan tag seperti, misalnya tag 'URL' dan 'AT_USER' dan menghapus simbol hashtag. Beberapa pemikiran lain mungkin adalah menghapus hanya tanda baca pada penyebutan pengguna dan mempertahankan nama pengguna atau menghapusnya sepenuhnya.

3) Mengganti bahasa gaul dan singkatan

Pengguna media sosial biasanya menulis secara informal dan teksnya banyak mengandung bahasa gaul dan singkatan. Bahasa gaul adalah jenis bahasa yang terdiri dari kata dan frasa yang dianggap sangat informal dan biasanya terbatas pada konteks atau sekelompok orang tertentu, sedangkan singkatan adalah bentuk singkat dari sebuah kata atau frasa. Kata-kata dan frasa-frasa ini, agar dapat ditafsirkan dengan benar, harus diganti agar dapat menjelaskan maknanya. Secara manual dibuat tabel pencarian yang terdiri dari 290 kata dan frasa tersebut, dan penggantinya. Beberapa contohnya adalah frasa “ty”, “qq”, dan “ya ampun”, yang masing-masing berarti dan menggantikan “terima kasih”, “menangis”, dan “ya Tuhan”.

4) Menghapus tanda baca

Untuk prapemrosesan teks, teknik klasik dalam informasi pengambilan dan penambangan data adalah dengan menghilangkan tanda baca. Namun, kehadiran tanda baca seringkali menunjukkan adanya suatu sentimen, misal data didapatkan pada sosial media.

5) Huruf kecil (lowercase)

Salah satu teknik pra-pemrosesan yang paling umum adalah dengan menggunakan huruf kecil pada semua kata. Dengan melakukan hal ini, kata-kata yang sama digabungkan dan dimensi masalah dikurangi.

6) Removing stopwords

Stopwords adalah kata-kata fungsi dengan frekuensi kehadiran yang tinggi di semua kalimat. Dianggap tidak perlu dianalisis, karena tidak banyak mengandung

informasi berguna. Kumpulan kata-kata ini tidak sepenuhnya ditentukan sebelumnya dan dapat diubah dengan menghapus atau menambahkan kata lain, tergantung pada aplikasinya.

7) Lemmatization

Metode lain untuk menggabungkan banyak kata menjadi satu adalah lemmatisasi. Metode ini menganalisis suatu kata secara morfologis dan menghilangkan akhiran infleksionalnya, sehingga menghasilkan bentuk dasar atau lemma seperti yang terdapat dalam kamus.

8) Tokenisasi

Dalam pengolahan teks, tokenisasi adalah prosedur pemisahan suatu teks menjadi kata, frasa, atau bagian lain yang bermakna, yaitu token. Dengan kata lain, tokenisasi adalah salah satu bentuk segmentasi teks. Biasanya, segmentasi dilakukan hanya dengan mempertimbangkan karakter alfabet atau alfanumerik yang dibatasi oleh karakter non-alfanumerik (misalnya tanda baca, spasi).

9) Stemming

Proses menghilangkan akhiran kata untuk mendeteksi bentuk akar atau batangnya. Dengan melakukan ini, banyak kata yang digabungkan dan dimensinya dikurangi. Ini adalah metode yang banyak digunakan dan umumnya memberikan hasil yang baik.

2.3.2 Stanza Dependency Parser

Proses menguraikan struktur tata bahasa sebuah kalimat menggunakan *stanza dependency parser* yang dikembangkan oleh stanford NLP Group. Sesuai yang tertulis pada laman resminya (https://stanfordnlp.github.io/stanza/index.html) URL (diakses pada 02-12-2022), *Stanza* adalah kumpulan alat yang akurat dan efisien untuk analisis linguistik banyak bahasa manusia. Mulai dari teks mentah hingga analisis sintaksis dan pengenalan entitas, *Stanza* menghadirkan model NLP tercanggih ke berbagai bahasa pilihan. *Stanza* adalah paket analisis bahasa alami Python. Berisi alat, yang dapat digunakan dalam pipeline, untuk mengubah string yang berisi teks bahasa manusia menjadi daftar kalimat dan kata, untuk menghasilkan bentuk dasar dari kata-kata tersebut, jenis kata dan fitur

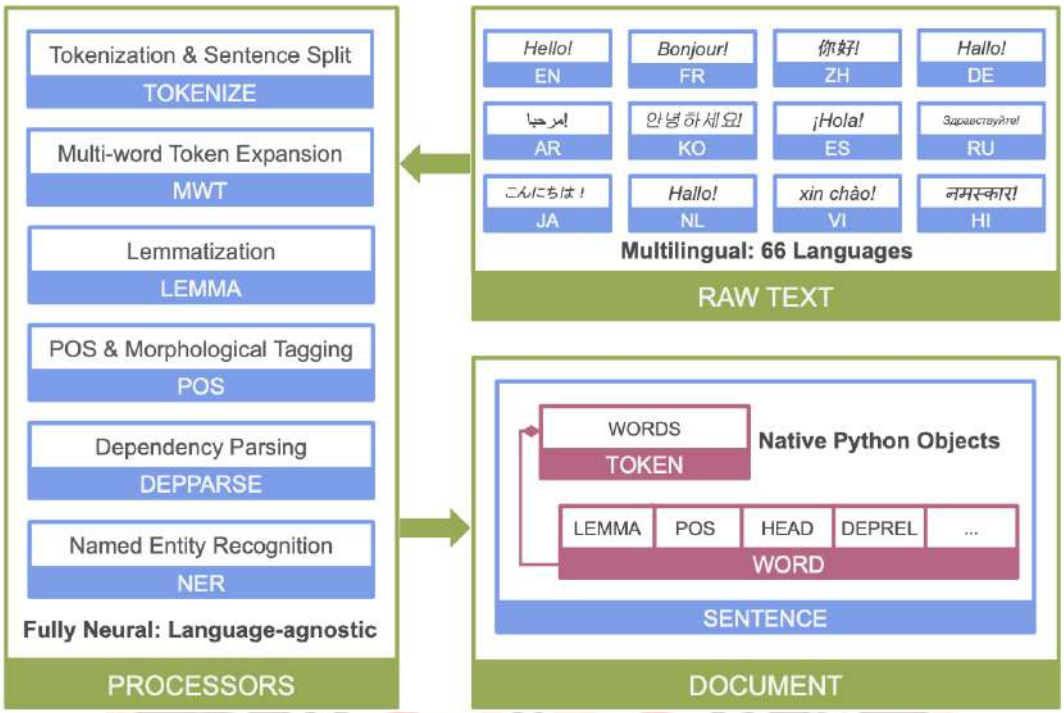
morfologinya, untuk memberikan penguraian ketergantungan struktur sintaksis, dan untuk mengenali entitas bernama. *Toolkit* ini dirancang agar dapat digunakan secara paralel di lebih dari 70 bahasa, menggunakan formalisme Ketergantungan Universal (*Universal Dependencies*).

Stanza sepenuhnya *open source* dan menyediakan model terlatih untuk semua bahasa dan kumpulan data yang didukung untuk diunduh publik. Stanza dapat memfasilitasi penelitian dan penerapan NLP multibahasa, serta mendorong penelitian masa depan yang menghasilkan wawasan dari bahasa manusia (Qi dkk., 2020).

Beberapa fitur yang ada pada stanza diantaranya sebagai berikut:

1) Implementasi Python asli (*Native Python*) memerlukan sedikit usaha untuk menyiapkannya;
2) Saluran jaringan saraf penuh (*full neural network pipeline*) untuk analisis teks yang kuat, termasuk tokenisasi, perluasan token multi-kata/*multi-word token* (MWT), lemmatisasi/*lemmatization*, *part-of-speech* (POS) dan penandaan fitur morfologi (*morphological features tagging*), penguraian ketergantungan (*dependency parsing*), dan pengenalan entitas Bernama (*named entity recognition*);
3) Model saraf terlatih (*pretrained neural models*) yang mendukung 70 bahasa (manusia);
4) Antarmuka Python yang stabil dan dipelihara secara resmi ke CoreNLP.

Berikut merupakan gambaran secara umum *stanza neural network* NLP pipeline dapat dilihat pada Gambar 2.2.



Gambar 2.2 Stanza neural network NLP pipeline

Berdasarkan uraian yang telah ditulis (Qi dkk., 2018, 2020), secara detail Gambar 2.2 dapat dijelaskan sebagai berikut:

1) Dari teks mentah hingga anotasi (*From raw text to annotations*).

Stanza menampilkan saluran saraf sepenuhnya yang mengambil teks mentah sebagai masukan, dan menghasilkan anotasi termasuk tokenisasi, perluasan token multi-kata, lemmatisasi, penandaan fitur *part-of-speech* dan morfologis, penguraian ketergantungan, dan pengenalan entitas bernama.

2) Tokenisasi dan Pemisahan Kalimat (*Tokenization and Sentence Splitting*).

Saat menyajikan teks mentah, *Stanza* memberi token dan mengelompokkan token menjadi kalimat sebagai langkah pemrosesan pertama. Tidak seperti kebanyakan *toolkit* yang ada, *Stanza* menggabungkan tokenisasi dan segmentasi kalimat dari teks mentah ke dalam satu modul. Hal ini dimodelkan sebagai masalah penandaan pada rangkaian karakter, dimana model memprediksi apakah karakter tertentu merupakan akhir dari sebuah token, akhir dari sebuah kalimat, atau akhir dari sebuah token multi-kata. *Stanza* memilih untuk memprediksi MWT bersamaan dengan tokenisasi karena tugas ini peka konteks dalam beberapa bahasa.

3) Ekspansi Token Multi-kata (*Multi-word Token Expansion*).

Setelah MWT diidentifikasi oleh tokenizer, MWT tersebut diperluas menjadi kata-kata sintaksis yang mendasarinya sebagai dasar pemrosesan hilir (*downstream processing*). Hal ini dicapai dengan ansambel leksikon frekuensi dan model neural *sequence-to-sequence* (seq2seq), untuk memastikan bahwa perluasan yang sering diamati dalam set pelatihan selalu diperluas dengan kuat sambil mempertahankan fleksibilitas untuk memodelkan kata-kata yang tidak terlihat secara statistik.

4) POS dan Penandaan Fitur Morfologi (*POS and Morphological Feature Tagging*).

Untuk setiap kata dalam sebuah kalimat, Stanza menetapkannya sebagai part-of-speech (POS), dan menganalisis fitur morfologi universalnya (UFeats, misalnya, tunggal/jamak, orang ke-1/2/3, dll.). Untuk memprediksi POS dan UFeats, stanza mengadopsi jaringan memori jangka pendek dua arah (Bi-LSTM) sebagai arsitektur dasar. Untuk konsistensi antara universal POS (UPOS), POS khusus treebank/ treebank-specific POS (XPOS), dan UFeats, mengadopsi mekanisme penilaian biaffine dari (Dozat dan Manning, 2017) untuk mengkondisikan prediksi XPOS dan UFeats pada UPOS.

5) Lemmatisasi (*Lemmatization*).

Stanza juga memberi lemmatisasi pada setiap kata dalam kalimat untuk memulihkan bentuk kanoniknya (misalnya, did→do). Mirip dengan ekspander token multi-kata, lemmatizer Stanza diimplementasikan sebagai ansambel lemmatizer berbasis kamus dan lemmatizer neural seq2seq. Pengklasifikasi tambahan dibangun pada keluaran encoder model seq2seq, untuk memprediksi pintasan seperti huruf kecil dan salinan identitas untuk ketahanan pada rangkaian masukan yang panjang seperti URL.

6) Penguraian Ketergantungan (*Dependency Parsing*).

Stanza mengurai setiap kalimat berdasarkan struktur sintaksisnya, di mana setiap kata dalam kalimat diberi kepala sintaksis yang merupakan kata lain dalam kalimat, atau dalam kasus kata dasar (*root word*), simbol akar buatan (*artificial root symbol*). Menerapkan pengurai ketergantungan saraf biaffine dalam berbasis Bi-LSTM (Dozat dan Manning, 2017). Selanjutnya melengkapi model ini dengan dua

fitur yang dimotivasi secara linguistik: fitur yang memprediksi urutan linearisasi dua kata dalam bahasa tertentu, dan fitur lainnya yang memprediksi jarak tipikal dalam urutan linier di antara keduanya. Penelitian sebelumnya telah menunjukkan bahwa fitur-fitur ini secara signifikan meningkatkan akurasi penguraian (Qi dkk., 2018).

7) Pengakuan Entitas Bernama (*Named Entity Recognition*).

Untuk setiap kalimat masukan, Stanza juga mengenali entitas bernama di dalamnya (misalnya nama orang, organisasi, dll.). Untuk NER mengadopsi penanda urutan berbasis representasi string yang dikontekstualisasikan dari (Akbik dkk., 2018). *Stanza* pertama-tama melatih model bahasa LSTM tingkat karakter maju dan mundur, dan pada waktu penandaan *stanza* menggabungkan representasi di akhir setiap posisi kata dari kedua model bahasa dengan penyematan kata, dan memasukkan hasilnya ke dalam Bi-LSTM satu lapisan standar penanda urutan dengan dekoder berbasis bidang acak bersyarat (*conditional random field*) (CRF).

Representasi *Stanford Dependencies* (SD) pada awalnya dikembangkan sebagai representasi praktis sintaksis bahasa Inggris, yang ditujukan untuk aplikasi pemahaman bahasa alami/*natural language understanding* (NLU). Namun, hal ini berakar kuat pada tradisi sintaksis berbasis hubungan tata bahasa, yang telah lama menekankan deskripsi lintas bahasa. Selain taksonomi ketergantungan universal (*universal dependency taxonomy*), kita juga perlu mengenali hubungan tata bahasa yang khusus untuk satu bahasa atau sekelompok kecil bahasa yang terkait. Hubungan bahasa-bahasa tertentu diperlukan untuk secara akurat menangkap kejeniusan bahasa tertentu namun tidak akan melibatkan konsep-konsep yang dapat digeneralisasikan secara luas. Sarannya di sini adalah bahwa relasi ini harus selalu dianggap sebagai subtipe dari relasi SD Universal/ Universal Stanford Dependencies (USD) yang sudah ada (De Marneffe dkk., 2014; Qi dkk., 2018).

Universal Dependency Relations berbahasa Indonesia dapat dibaca pada laman (https://universaldependencies.org/treebanks/id-comparison.html) URL (diakses pada 02-12-2022). Secara umum *Universal Dependency Relations* pada laman (https://universaldependencies.org/u/dep/index.html) URL (diakses pada 02-12-2022), tertulis masing-masing bahasa dapat mendefinisikan hubungan yang

lebih spesifik sebagai subtipe dari tipe universal yang didefinisikan di sini. Relasi yang disubtipe selalu dimulai dengan tipe dasar, diikuti dengan titik dua dan string subtipe. Secara umum, subtipe bersifat spesifik bahasa dan opsional. Namun, beberapa subtipe diasumsikan berlaku untuk banyak bahasa dan harus dianggap semi-wajib: Jika bahasa tersebut memiliki fenomena yang menjadi fokus subtipe tersebut, maka subtipe tersebut harus digunakan. Ada 37 hubungan sintaksis universal (*universal syntactic relations*), untuk mengetahui secara keseluruhan *Universal Dependency Relations* dapat mengakses lamannya secara langsung, beberapa contoh *Universal Dependency Relations* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh *Universal Dependency Relations*

Relasi	Keterangan
advmod	adverbial modifier; Pengubah adverbial suatu kata adalah kata keterangan (non-klausal) atau frasa adverbial yang berfungsi untuk mengubah suatu predikat atau kata pengubah.
amod	adjectival modifier; Pengubah kata sifat dari kata benda (atau kata ganti) adalah frasa kata sifat apa pun yang berfungsi untuk mengubah kata benda (atau kata ganti). Relasi tersebut berlaku baik makna kata benda diubah secara komposisional (misalnya, rumah besar) atau secara idiomatis (hot dog).
appos	appositional modifier; Pengubah aposisional suatu kata benda adalah suatu nominal tepat setelah kata benda pertama yang berfungsi untuk mendefinisikan, memodifikasi, memberi nama, atau mendeskripsikan kata benda tersebut. Ini mencakup contoh-contoh dalam tanda kurung, serta definisi singkatan dalam salah satu struktur ini.
aux	auxiliary; Aux (auxiliary/tambahan) suatu klausa adalah kata fungsi yang diasosiasikan dengan predikat verbal yang mengungkapkan kategori seperti tense, mood, aspek, suara atau bukti. Ini sering kali merupakan kata kerja (yang mungkin juga memiliki kegunaan non-bantu) tetapi banyak bahasa memiliki penanda TAME nonverbal dan ini juga diperlakukan sebagai contoh aux.
cc (cconj)	coordinating conjunction; cc adalah hubungan antara konjungsi dan konjungsi koordinatif sebelumnya. Konjungsi koordinatif adalah kata yang menghubungkan kata-kata atau unsur-unsur yang lebih besar tanpa mensubordinasikan satu sama lain secara sintaksis dan mengungkapkan hubungan semantik di antara kata-kata tersebut.

Tabel 2.2 Contoh Universal Dependency Relations (lanjutan)

Relasi	Keterangan
conj	conjunct; Konjungsi adalah relasi antara dua elemen yang dihubungkan oleh konjungsi koordinatif, seperti dan, atau, dan seterusnya. Kita memperlakukan konjungsi secara asimetris: Kepala relasi adalah konjungsi pertama dan semua konjungsi lainnya bergantung padanya melalui relasi konj.
compound	compound; Relasi majemuk (*compound relation*) digunakan untuk menganalisis senyawa, yaitu gabungan leksem-leksem yang secara morfosintaksis berperilaku sebagai kata tunggal. Kasus yang sering terjadi adalah: Kata majemuk nominal yang ditulis sebagai kata tersendiri, misalnya jus apel bahasa Inggris.
nmod	nominal modifier; Relasi nmod digunakan untuk dependen nominal dari kata benda atau frasa kata benda lain dan secara fungsional berhubungan dengan suatu atribut, atau pelengkap genitif.
nsubj	nominal subject; Subjek nominal (nsubj) adalah nominal yang merupakan subjek sintaksis dan proto-agen suatu klausa. Artinya, klausa tersebut berada pada posisi yang lolos uji tata bahasa untuk subjektivitas, dan argumen ini lebih bersifat agentif, pelaku, atau proto-agen dari klausa tersebut. Nominal ini bisa diawali dengan kata benda, atau bisa berupa kata ganti atau kata ganti relatif, atau, dalam konteks elipsis, benda lain seperti kata sifat.
obj	object; Objek suatu kata kerja merupakan argumen paling inti kedua dari suatu kata kerja setelah subjek. Biasanya, ini adalah frasa kata benda yang menunjukkan entitas yang ditindaklanjuti atau yang mengalami perubahan keadaan atau gerak (proto-pasien).
punct	punctuation; Ini digunakan untuk tanda baca apa pun dalam klausa, jika tanda baca dipertahankan dalam dependensi yang diketik. Perhatikan bahwa simbol yang diberi tag SYM (symbol) bukanlah tanda baca dan tidak dapat dilampirkan melalui hubungan tanda baca.
root	root; Hubungan akar gramatikal menunjuk pada akar kalimat. Node palsu ROOT digunakan sebagai gubernur. Node ROOT diindeks dengan 0, karena pengindeksan kata sebenarnya dalam kalimat dimulai dari 1. (Node ROOT tidak direpresentasikan secara eksplisit di CoNLL-U.)

Objek Word menyimpan kata sintaksis dan semua anotasi tingkat kata. Dalam hal token multi-kata (MWT), kata-kata dihasilkan sebagai hasil penerapan Prosesor MWT, dan digunakan dalam semua analisis sintaksis hilir seperti penandaan,

lemmatisasi, dan penguraian. Jika sebuah Word merupakan hasil perluasan MWT, teksnya biasanya tidak akan ditemukan dalam teks mentah masukan. Selain token multi-kata, Kata-kata harus serupa dengan “token” yang biasa kita lihat di tempat lain. Untuk membaca data object and annotations pada stanza dapat mengunjungi laman resminya (https://stanfordnlp.github.io/stanza/data_objects.html) URL (diakses pada 02-12-2022). Fokus pada word annotations, Word berisi properti seperti yang terlihat pada Tabel 2.3.

Tabel 2.3 Properti pada word

Properti	Tipe	Deskripsi
id	int	Indeks kata ini dalam kalimat, berbasis 1 (indeks 0 dicadangkan untuk simbol buatan yang mewakili akar pohon sintaksis).
text	str	Teks kata ini. Contoh: 'The'.
lemma	str	Lemma dari kata ini.
upos (pos)	str	Bagian universal part-of-speech dari kata ini. Contoh: 'KATA BENDA'.
xpos	str	Bagian treebank-specific part-of-speech dari kata ini. Contoh: 'NNP'.
feats	str	Ciri-ciri morfologi kata ini. Contoh: 'Jenis Kelamin=Wanita|Orang=3'.
head	int	Id kepala sintaksis kata ini dalam kalimat, berdasarkan 1 untuk kata sebenarnya dalam kalimat (0 dicadangkan untuk simbol buatan yang mewakili akar pohon sintaksis).
deprel	str	Hubungan ketergantungan antara kata ini dan kepala sintaksisnya. Contoh: 'nmod'.
deps	str	Kombinasi head dan deprel yang menangkap semua informasi ketergantungan sintaksis. Terlihat di file CoNLL-U yang dirilis dari Ketergantungan Universal, tidak diprediksi oleh Pipeline *stanza*.
misc	str	Anotasi lain-lain sehubungan dengan kata ini. Pipeline menggunakan bidang ini untuk menyimpan informasi offset karakter secara internal, misalnya.
parent	token	Sebuah “penunjuk kembali” ke token induk tempat kata ini menjadi bagiannya. Dalam kasus token multi-kata, token dapat menjadi induk dari beberapa kata.

2.3.3 Semi-Supervised Learning

Semi-supervised learning adalah cabang *machine learning* yang memanfaatkan sekumpulan kecil data berlabel dan sekumpulan besar data tidak

berlabel untuk meningkatkan akurasi pembelajaran (Hirst dan Søgaard, 2013; Ligthart dkk., 2021b). *Semi-supervised learning* umumnya ada 5 tipe (Hemmatian dan Sohrabi, 2017) di antaranya:

a. Self-training

Dari beberapa metode yang ada dalam *semi-supervised learning*, pendekatan *self-training* dianggap sebagai salah satu pendekatan yang terkenal dan populer di antara pendekatan yang telah banyak digunakan (Gao dkk., 2014; He dan Zhou, 2011; Zimmerann dkk., 2014). Dalam metode *self-training*, pada awalnya *classifier* harus dilatih sedikit sampel training berlabel, kemudian *classifier* yang dilatih akan digunakan untuk mengklasifikasikan sampel uji (unlabeled). Sampel uji yang telah diberi label dengan keandalan maksimum akan ditambahkan ke set sampel pelatihan. Pengklasifikasi akan dilatih dengan kumpulan semua (termasuk baru) sampel pelatihan lagi dan prosesnya akan diulang. Pelatihan mandiri, menghasilkan model melalui data berlabel sentimen, dan menggunakan model ini untuk memprediksi data tanpa label sentimen. Pada hasil prediksi, data tanpa label sentimen dengan tingkat kepercayaan tinggi dipilih dan ditambahkan ke data dengan label sentimen dengan label sentimen terlampir (Hong dkk., 2014).

b. Co-training

Co-training adalah metode *semi-supervised learning*, yang menggunakan data berlabel dan tidak berlabel. Diasumsikan bahwa, setiap contoh dapat dijelaskan oleh dua set fitur yang berbeda yang memiliki informasi yang berbeda dan saling melengkapi tentang setiap sampel. Dalam pelatihan bersama, dua pengklasifikasi akan dilatih secara terpisah dan informasi yang diperoleh dari pelatihan ini dibagikan satu sama lain. Kemudian salah satu dari dua pengklasifikasi ini melatih ulang dengan menggunakan sampel pelatihan yang baru saja ditambahkan oleh pengklasifikasi lainnya. Dengan cara ini satu set besar data pelatihan akan terbentuk. Metode ini pertama kali diterapkan oleh (Blum dan Mitchell, 1998).

c. Multi view learning

Dalam metode ini, asumsi utama adalah bahwa kumpulan hipotesis kompatibel bersama. Dalam metode ini, tujuan akhirnya adalah menghasilkan k model berdasarkan k view. Dengan memanfaatkan kesepakatan ini, representasi masalah

yang berbeda digunakan untuk meningkatkan kinerja klasifikasi secara keseluruhan. Salah satu penerapan *multi view learning* adalah dalam analisis sentimen lintas bahasa. Analisis sentimen lintas bahasa dianggap penting untuk mengklasifikasikan sentimen dan telah diselidiki secara luas dalam beberapa tahun terakhir (Hajmohammadi dkk., 2014).

d. Graph-based methods

Pendekatan *graph-based learning* telah banyak digunakan (Ansari dkk., 2020; J. Wang dkk., 2015; Xing dkk., 2017; Yoon dkk., 2020), pendekatan *graph-based learning* cukup efektif ketika digunakan untuk tugas NLP. *Graph-based learning* merepresentasikan data sebagai graph berbobot di mana simpul mewakili contoh dan tepi mencerminkan contoh kesamaan. Dapat diasumsikan bahwa contoh yang terhubung erat kemungkinan termasuk dalam kelas yang sama.

e. Generative models

Model generatif mendefinisikan distribusi pada input dan pelatihan dilakukan untuk setiap kelas. Kemudian aturan *Bayes* dapat digunakan untuk memprediksi kepemilikan sampel terhadap kelas pada saat tahap pengujian. Tiga model dasar yang berbeda secara konseptual digabungkan dalam sistem ini: satu berdasarkan pendekatan generatif (model bahasa), satu berdasarkan representasi kontinu kalimat, dan satu berdasarkan penimbangan ulang *TF-IDF* yang cerdas dari representasi *bag-of-word* dokumen (Hemmatian dan Sohrabi, 2017).

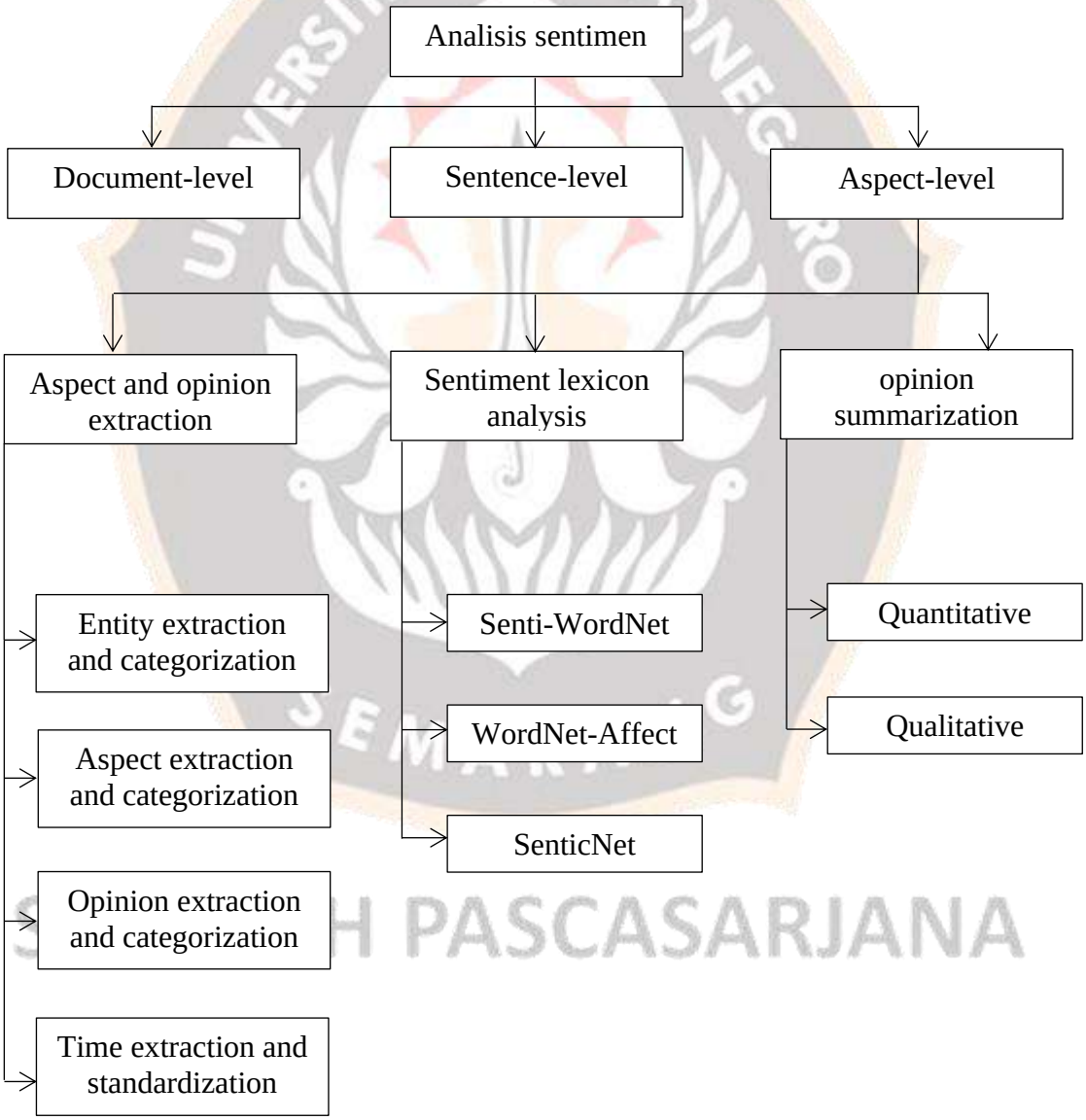
2.3.4 Analisis sentiment berbasis aspek

Analisis sentimen merupakan salah satu tugas dari *opinion mining* dan memiliki tujuan untuk pengenalan sentimen dan pemeriksaan opini publik yang dianggap sebagai area penelitian di bidang penambangan teks (B. Liu, 2012). Analisis sentimen memiliki tiga tingkat di antaranya tingkat dokumen, tingkat kalimat, dan tingkat aspek (B. Liu, 2012).

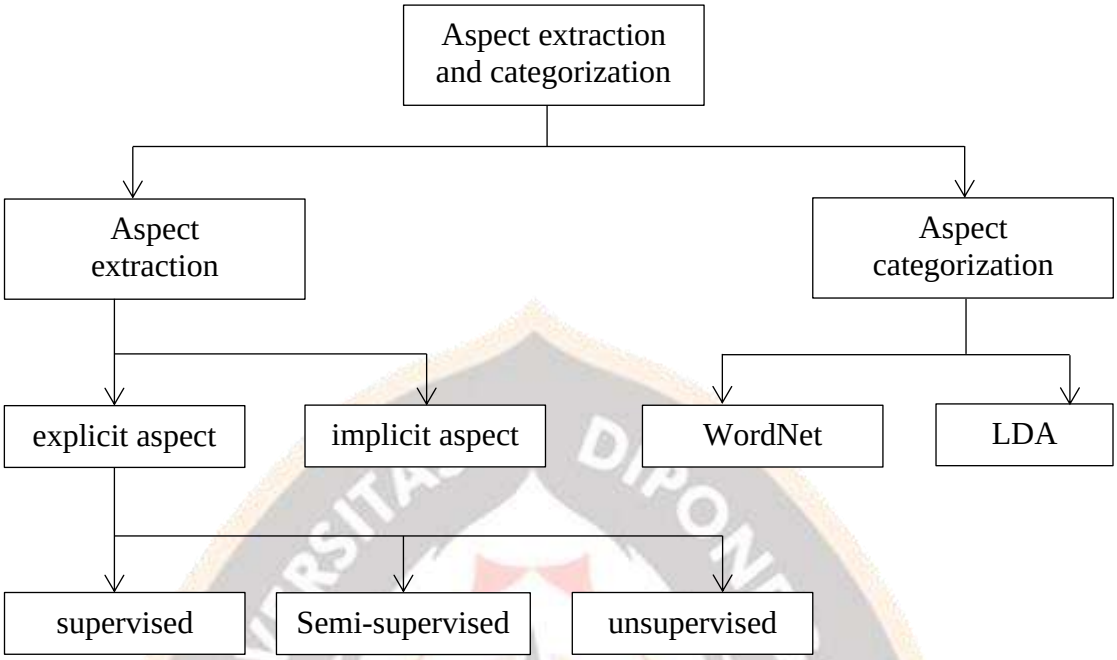
Penelitian ini fokus pada tingkat aspek. Menurut (Hemmatian dan Sohrabi, 2017) klasifikasi sentimen teks pada tingkat dokumen dan kalimat sangat membantu dalam banyak kasus tetapi tidak memberikan semua detail yang diperlukan. Analisis sentimen pada tingkat aspek menurut (Chinsha dan Joseph, 2015) dapat menghasilkan ringkasan sentimen tentang berbagai aspek entitas yang

diinginkan, serta dapat dilihat bahwa tingkat aspek ini memberikan hasil yang lebih akurat.

Salah satu tahapan terpenting dalam klasifikasi sentimen adalah ekstraksi aspek (Rana dan Cheah, 2016). Ada tiga sub-tugas dalam analisis sentimen berbasis aspek di antaranya *aspect and opinion extraction*, *sentiment lexicon analysis* dan *opinion summarization* (Rana dan Cheah, 2016) seperti yang terlihat pada Gambar 2.3. Sedangkan untuk pendekatan yang digunakan dalam ekstraksi aspek dapat dilihat pada Gambar 2.4.

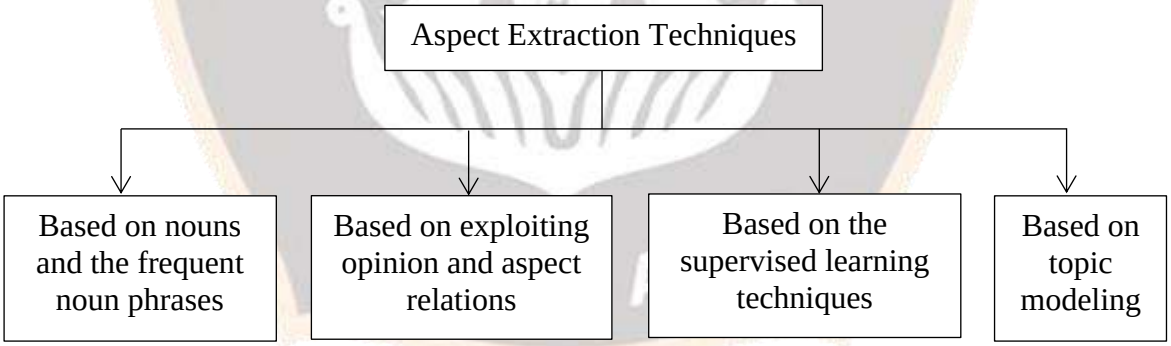


Gambar 2.3 Tingkat klasifikasi sentimen analisis.



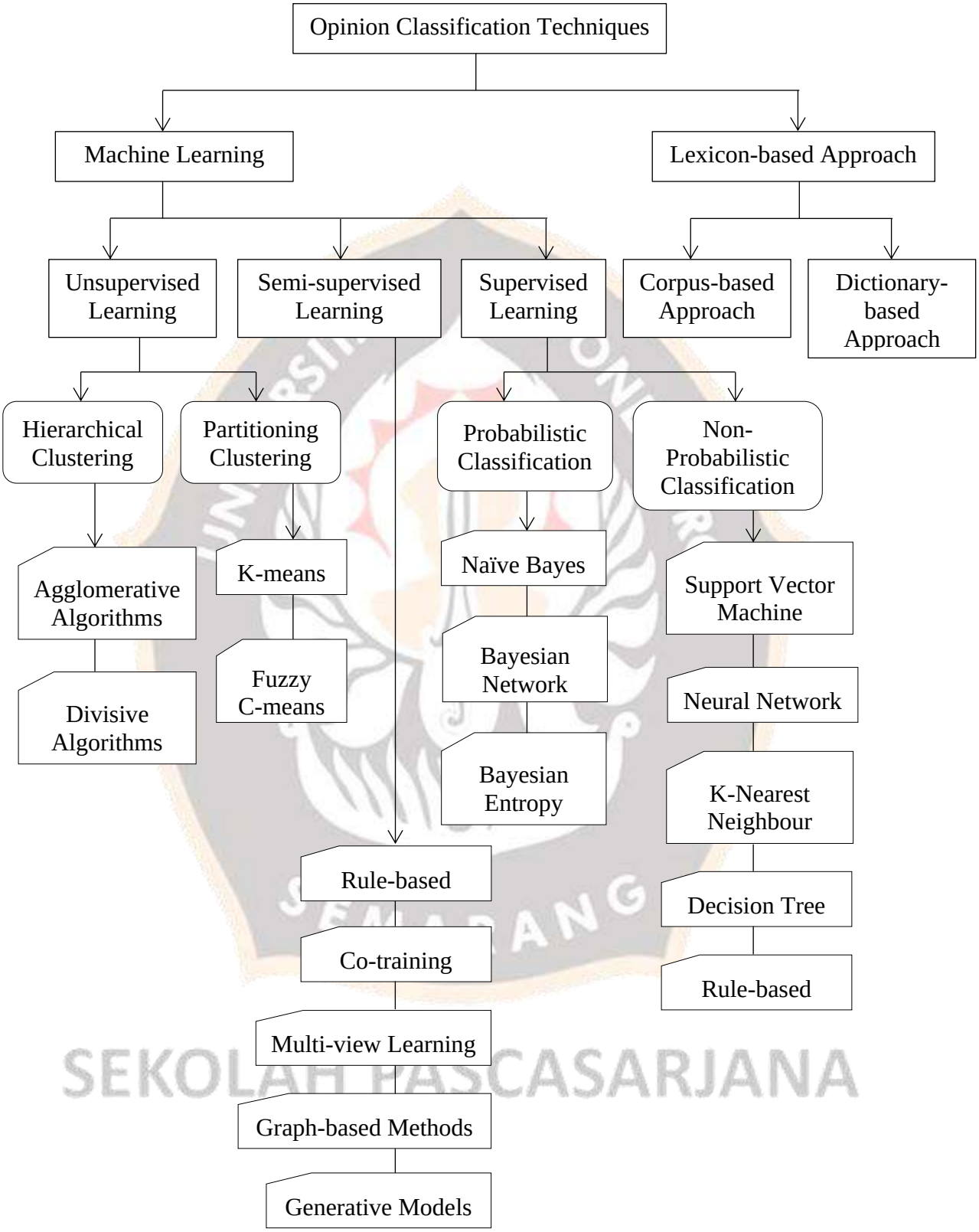
Gambar 2.4 Pendekatan ekstraksi aspek.

Sebagian besar penelitian di bidang ekstraksi aspek telah difokuskan pada ulasan online (S. Li dkk., 2015; Lv dkk., 2017). Secara umum ada empat teknik untuk ekstraksi aspek (B. Liu, 2012) seperti yang terlihat pada Gambar 2.5.



Gambar 2.5 Teknik ekstraksi aspek.

Langkah paling penting dan kritis dalam penambangan opini adalah memilih teknik yang tepat untuk mengklasifikasikan sentimen (Hemmatian dan Sohrabi, 2017). Teknik klasifikasi sentimen dalam penambangan opini yang diusulkan dalam literatur (Hemmatian dan Sohrabi, 2017) ada dua: pendekatan *machine learning* dan *lexicon-based*, secara keseluruhan dapat dilihat pada Gambar 2.6.



Gambar 2.6 Teknik klasifikasi sentimen dalam penambangan opini.

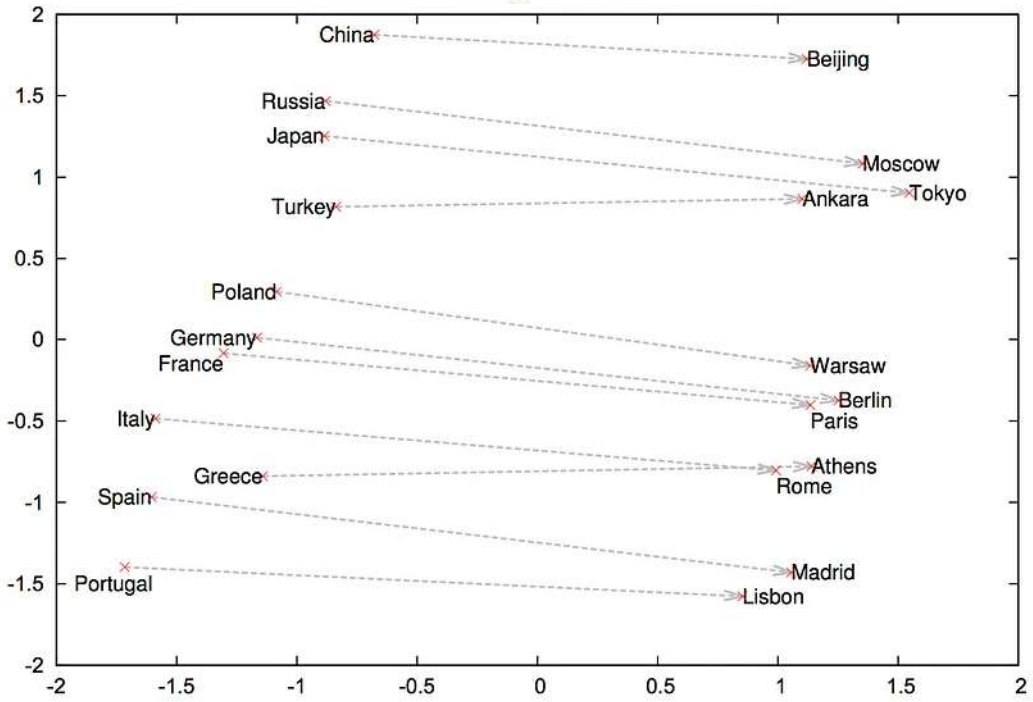
2.3.5 *Word Embedding*

Word embedding merupakan representasi vektor dari kata-kata dalam sebuah ruang multidimensi, di mana kata-kata yang memiliki makna atau penggunaan serupa cenderung memiliki representasi yang dekat satu sama lain. *Word embedding* memungkinkan komputer untuk memahami makna relatif antara kata-kata dalam teks secara lebih baik daripada representasi kata yang sederhana seperti *one-hot encoding* atau matriks frekuensi dokumen.*Word embedding* sering kali dihasilkan menggunakan teknik-teknik pembelajaran mesin, terutama melalui model-model seperti *Word2Vec*, *GloVe* (*Global Vectors for Word Representation*), atau *fastText*. Model-model ini mempelajari representasi kata-kata berdasarkan konteks di mana kata-kata tersebut muncul dalam teks yang cukup besar.

Hasil dari *word embedding* adalah vektor yang mewakili kata-kata dalam teks, di mana hubungan semantik antara kata-kata tercermin dalam jarak dan arah di antara vektor-vektor tersebut. Misalnya, dalam ruang *word embedding*, kata-kata yang sering muncul bersama-sama atau sering kali digunakan dalam konteks yang serupa akan memiliki representasi vektor yang lebih dekat satu sama lain. *Word embedding* telah menjadi alat yang sangat penting dalam pemrosesan bahasa alami (*Natural Language Processing*, NLP), serta berbagai aplikasi seperti mesin pencarian, sistem rekomendasi, analisis sentimen, dan banyak lagi. Dengan menggunakan word embedding, model-model NLP dapat belajar dari dan menginterpretasikan teks dengan cara yang lebih efisien dan serupa dengan cara manusia memahami bahasa.

Pada penelitian ini teknik *word embedding* yang akan digunakan yaitu *word2vec*. *Word2vec* merupakan salah satu teknik *embedding* kata yang bermanfaat dalam mengubah kata menjadi vektor berdimensi N (Mikolov dkk., 2013). Contohnya, kata “Indonesia” direpresentasikan sebagai vektor N-dimensi, misalnya [0.2, 0.4, -0.8, 0.9, -0.5]. Vektor tersebut bukan hanya mencerminkan aspek sintaksis kata, tetapi juga maknanya secara semantik. Misalnya, apabila *word2vec* dilatih dengan menggunakan korpus teks yang cukup luas, vektor representasi untuk kata “Indonesia” akan berdekatan dengan vektor “Jakarta”, mirip dengan bagaimana vektor untuk “Perancis” akan berdekatan dengan vektor “Paris”. Secara

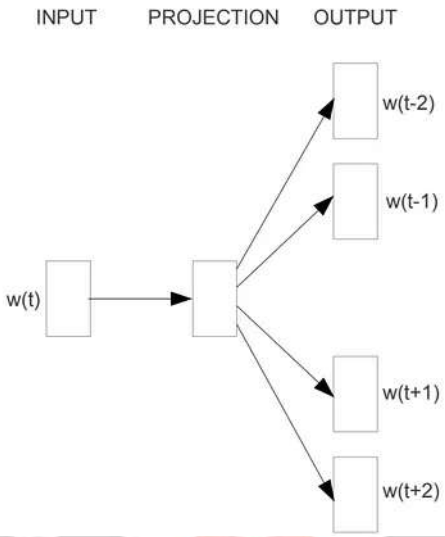
sederhana, model *word2vec* akan mengenali bahwa "Indonesia" dan "Jakarta" memiliki keterkaitan yang serupa dengan "Perancis" dan "Paris", yaitu sebagai negara dan ibu kotanya. Contoh hubungan negara dan ibu kotanya seperti yang ditunjukkan pada Gambar 2.7.



Gambar 2.7 Ilustrasi model *word2vec* dalam mengenali hubungan negara dan ibu kotanya (Demeester dkk., 2016)

Word2Vec menggunakan *neural network* untuk mendapatkan vektor tersebut. Arsitektur *Word2vec* hanya terdiri dari 3 (tiga) *layer* yaitu *Input layer*, *Hidden Layer*, dan *Output layer*. Input pada *Word2vec* berbentuk *one-hot encoded vector* dengan panjang = jumlah kata unik pada data training (Mikolov dkk., 2013). Terdapat 2 jenis arsitektur *neural network* dari *Word2Vec* yaitu "*Skip-gram*" dan *"Continous Bag of Word" (CBOW)*.

Pada *Skip-gram*, tujuan dari arsitektur *skip-gram* adalah untuk memprediksi konteks (*output*) di sekitar *current word* (*input*). Contoh arsitektur *Skip-gram* seperti yang ditunjukkan pada Gambar 2.8.

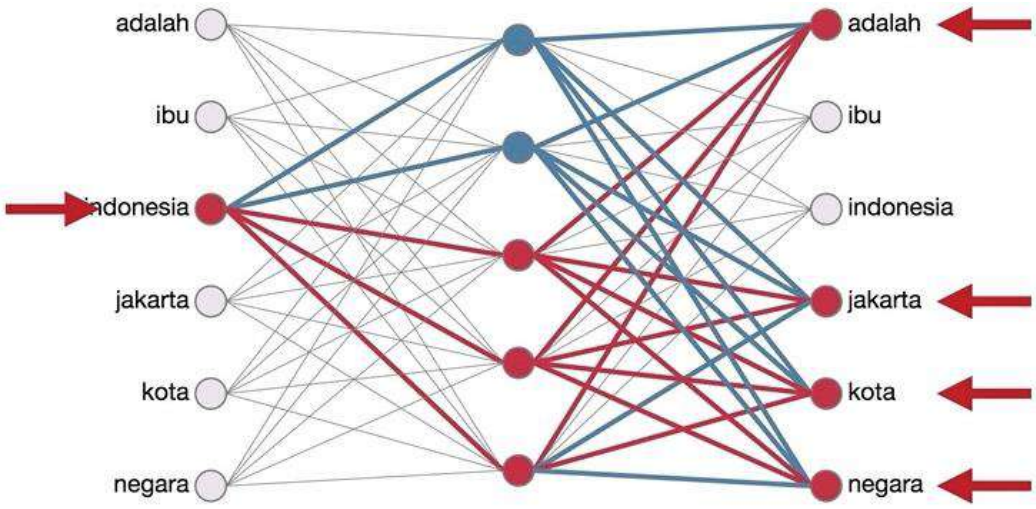


Gambar 2.8 Ilustrasi arsitektur *skip-gram* (Mikolov dkk., 2013)

Pada *skip-gram* ada istilah *window size*, untuk melihat kata yang ada pada kanan dan kiri kata *input*. Misal data *trainingnya* adalah sebuah kalimat "Ibu kota Negara Indonesia adalah Jakarta" dengan *window size* = 2. Untuk memahaminya bisa dilihat ilustrasi pada Gambar 2.9.

Gambar 2.9 Ilustrasi penggunaan *windowing skip-gram* (Firdaus, 2019)

Apabila kalimat di atas diinputkan pada *neural network skip-gram*, hasil ilustrasinya seperti yang ditunjukkan pada Gambar 2.10.



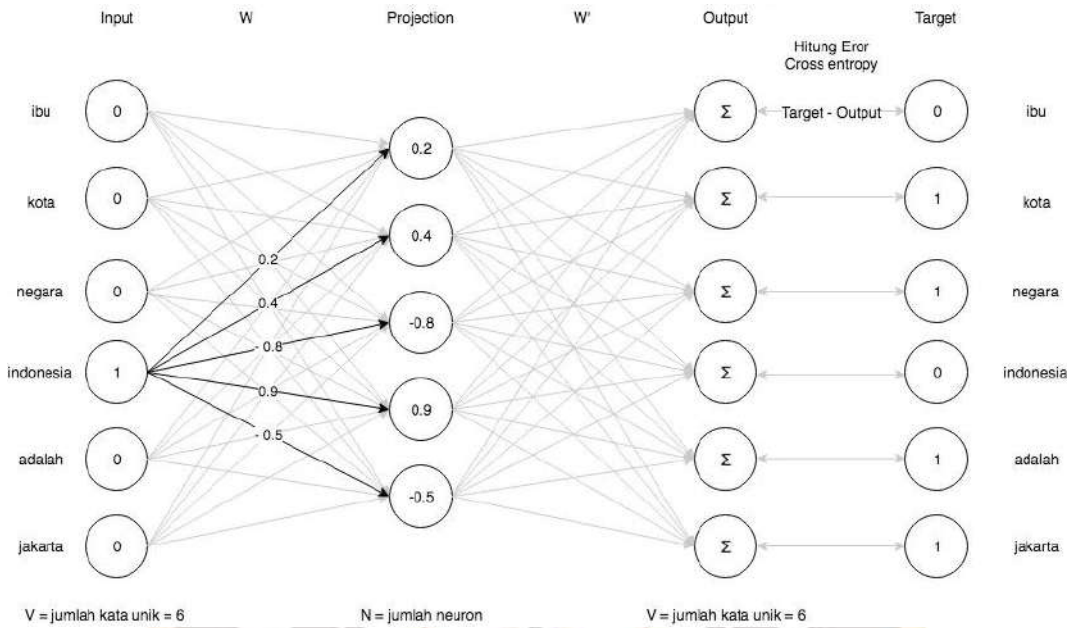
Gambar 2.10 Ilustrasi penggunaan *neural network skip-gram* (Firdaus, 2019)

Penjelasan detailanya, dimulai dari data input berbentuk *one-hot encoded vector* sehingga bentuk datanya adalah seperti yang ditunjukkan pada Gambar 2.11. (misal seluruh data telah dilakukan *lowercase*).

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

Gambar 2.11 Ilustrasi proses *one-hot encode* (Firdaus, 2019)

Selanjutnya, setelah proses *one-hot encode* akan ada proses berikutnya misal *current word* = Indonesia, maka ilustrasinya seperti yang ditunjukkan pada Gambar 2.12.

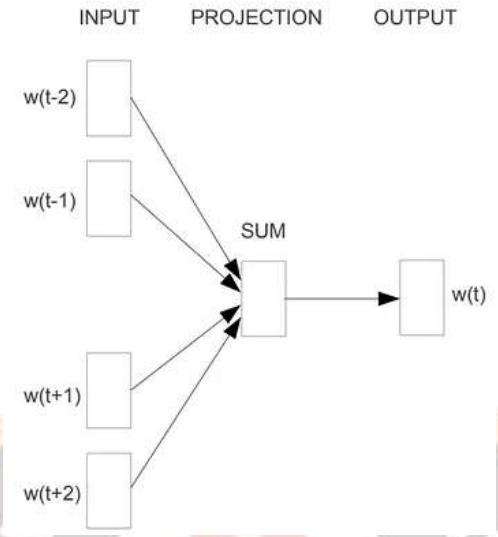


Gambar 2.12 Ilustrasi cara kerja *Word2Vec Skip-gram* (Firdaus, 2019)

Pada awalnya, bobot untuk matriks W dan W' diinisialisasi secara acak. Matriks W memiliki dimensi V x N, sementara W' memiliki dimensi N x V. Pada tahap f*eedforward*, vektor input didotkan dengan bobot W, menghasilkan nilai di layer proyeksi. Selanjutnya, layer proyeksi didotkan dengan bobot W', menghasilkan vektor output. Setelah mendapatkan output, nilai *error* dihitung menggunakan metode *cross entropy*, yaitu selisih antara target dan output. Proses berlanjut dengan tahap *backpropagation* menggunakan teknik *gradient descent* untuk memperbarui bobot W dan W'. Iterasi antara *feedforward* dan *backpropagation* dilakukan sampai mencapai nilai *error minimum*.

Setelah mencapai nilai *error minimum* dalam *cross entropy*, vektor yang merepresentasikan kata diambil dari matriks W dengan cara mengalikan *dot product* antara vektor *one-hot encoded* untuk setiap kata dengan matriks W, sedangkan bobot pada matriks W' diabaikan.

Berikutnya adalah *"Continous Bag of Word" (CBOW)*. Arsitektur *Word2Vec CBOW* adalah kebalikan dari *Word2vec skip-gram*. Tujuannya adalah untuk memprediksi kata (*output*) ketika diberikan konteks disekitar kata tersebut (*input*). Ilustrasinya seperti yang terlihat pada Gambar 2.13.

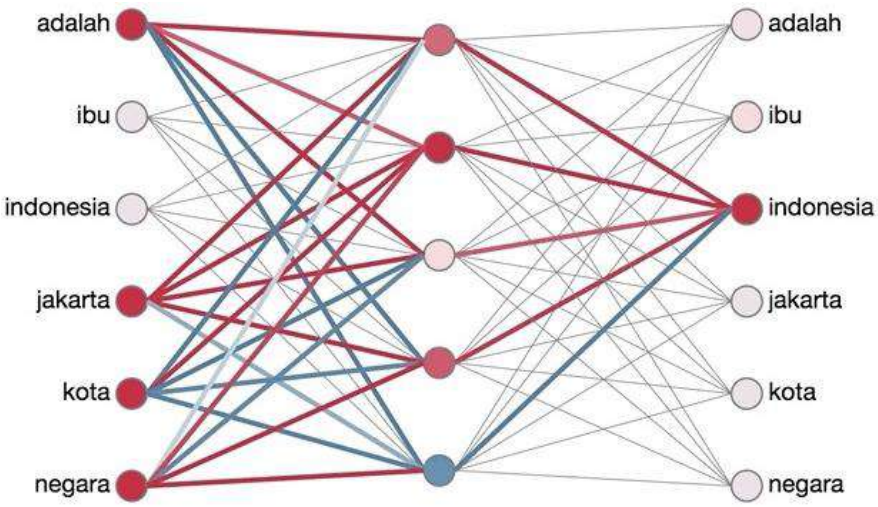


Gambar 2.13 Ilustrasi arsitektur *CBOW* (Mikolov dkk., 2013)

Contoh kalimat yang sama seperti sebelumnya, misal ada kalimat "Ibu kota Negara Indonesia adalah Jakarta". Kemudian apabila diproses menggunakan *word2vec CBOW* ilustrasinya akan seperti yang terlihat pada Gambar 2.14.

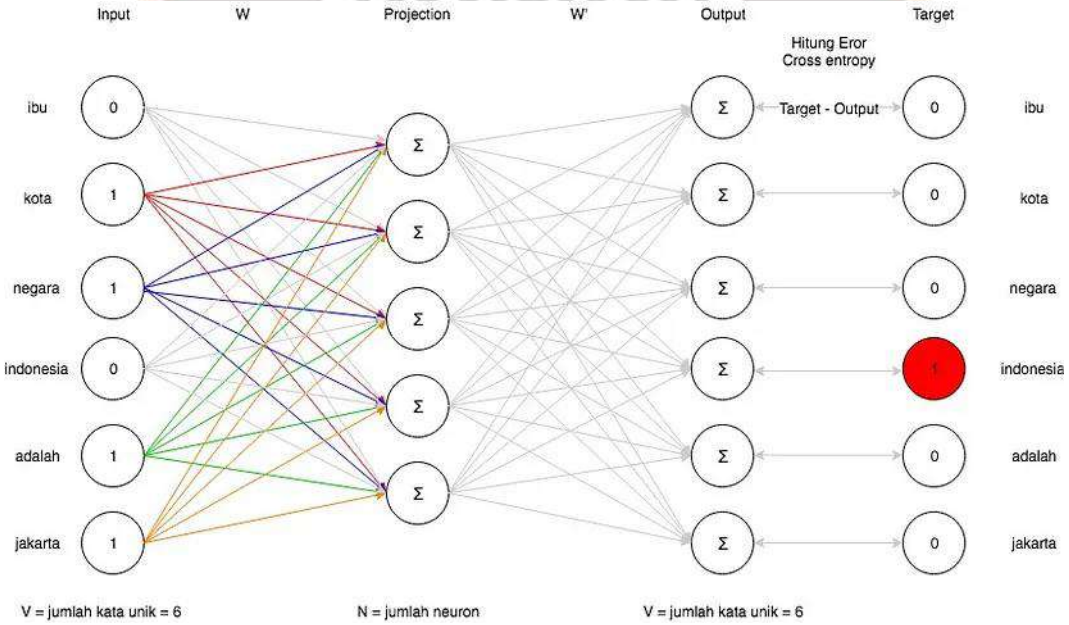
Gambar 2.14 Ilustrasi penggunaan *windowing CBOW* (Firdaus, 2019)

Apabila kalimat di atas diilustrasikan lebih detail untuk mengetahui cara kerja *word2vec CBOW* maka ilustrasinya seperti yang terlihat pada Gambar 2.15.



Gambar 2.15 Ilustrasi penggunaan *neural network CBOW* (Firdaus, 2019)

Pada *Word2Vec CBOW* data *input* yang digunakan berbentuk *n-hot encoded* vektor. Apabila pada tahap *training*, kata input = 1 sedangkan yang lain akan bernilai 0. Contoh kasus ini menggunakan target hanya 1 kata, maka target *output* akan berbentuk *one-hot encoded* vektor. Misal diilustrasikan apabila target outputnya adalah kata "Indonesia" maka hasil ilustrasinya seperti yang terlihat pada Gambar 2.16.



Gambar 2.16 Ilustrasi cara kerja *Word2Vec CBOW* (Firdaus, 2019)

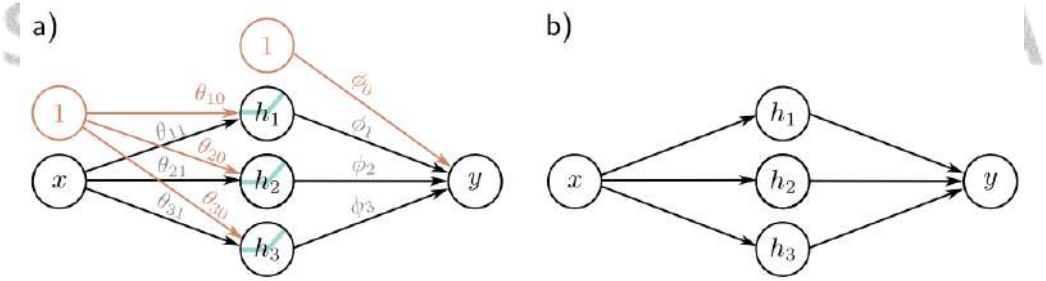
Pada Gambar 2.16 prosesnya hampir sama dengan struktur *Skip-gram*. Perbedaannya terletak pada data *input* yang diwakili oleh vektor *n-hot encoded* dan target *output* yang direpresentasikan sebagai vektor *one-hot encoded*. Setelah proses pelatihan selesai dan mencapai kesalahan *minimum*, bisa mendapatkan representasi vektor kata dengan mengalikan vektor *one-hot encoded* untuk setiap kata dengan bobot matriks W.

2.3.6 *Deep Learning*

Deep learning adalah teknik di dalam bidang kecerdasan buatan (AI) yang mengajarkan komputer untuk mengolah data dengan cara yang terinspirasi oleh fungsi otak manusia. Model *deep learning* mampu mengidentifikasi pola yang kompleks dalam berbagai jenis data, seperti gambar, teks, suara, dan informasi lainnya, untuk menghasilkan pemahaman dan prediksi yang akurat. Teknik *deep learning* dapat digunakan untuk mengotomatiskan tugas-tugas yang sebelumnya hanya dapat dilakukan oleh manusia, seperti memberikan deskripsi pada gambar atau mengubah file suara menjadi teks. Beberapa algoritma yang ada pada *deep learning* diantaranya *Neural Network* (NN), *Convolutional Neural Network* (CNN), dan *Recurrent Neural Network* (RNN) (Prince, 2024).

1. *Neural Network*

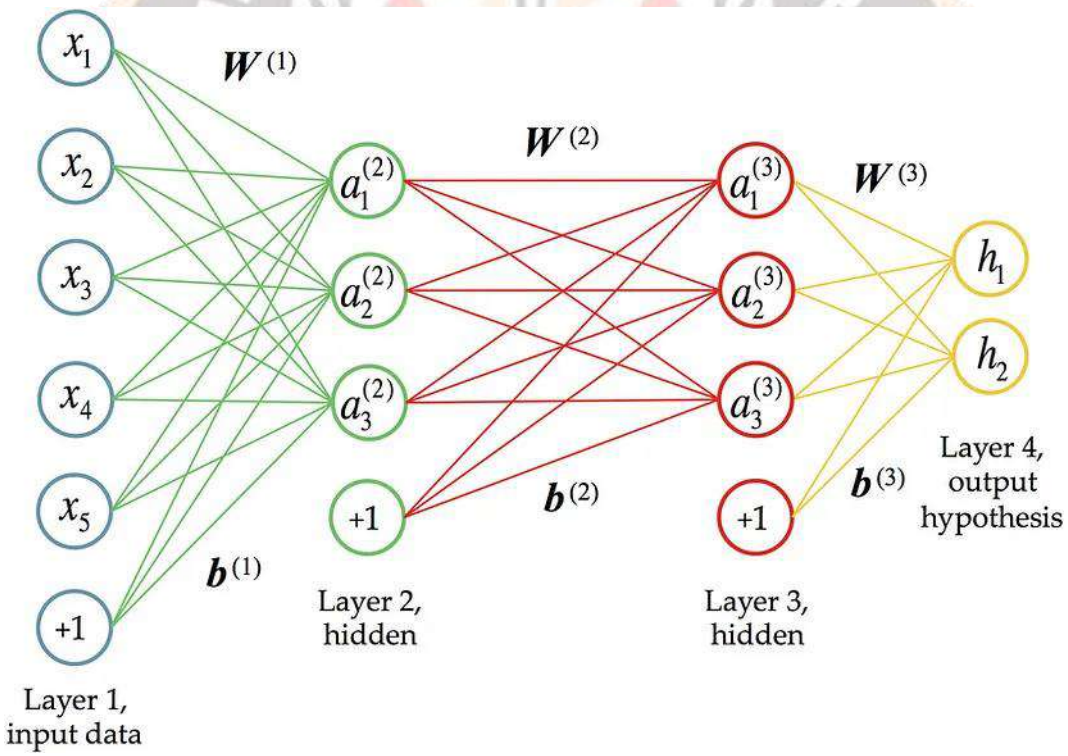
Konsep dasar *neural network* diawali dengan pengertian mengenai *neuron* yang merupakan unit dasar dari *neural network*. *Neuron* menerima masukan dari *neuron-neuron* lainnya atau dari sumber masukan eksternal, dan menghasilkan keluaran berdasarkan suatu fungsi aktivasi yang ditentukan. Misal ada *input layer*, *output layer*, dan *hidden layers* apabila diilustrasikan seperti yang terlihat pada Gambar 2.17.



Gambar 2.17 Ilustrasi *neural network* (Prince, 2024)

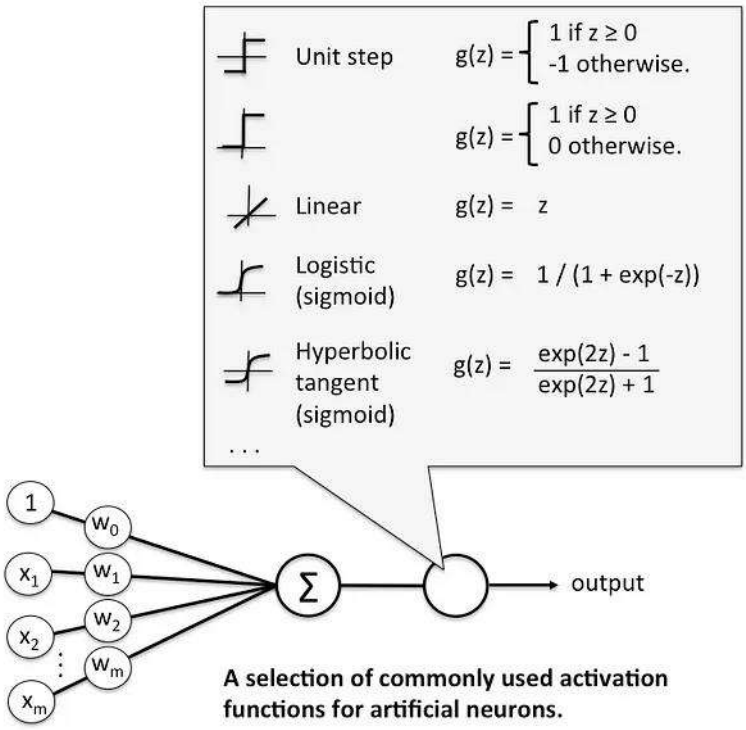
Pada Gambar 2.17a diketahui *input layer* ada di sebelah kiri, *hidden layer* ada di tengah, dan *output layer* ada di sebelah kanan. Setiap koneksi mewakili satu dari sepuluh parameter. Untuk menyederhanakan representasi ini, biasanya tidak ditampilkan parameter intersepnya, sehingga jaringan ini biasanya digambarkan seperti pada Gambar 2.17b. Lebih detail mengenai ilustrasi *layer* pada arsitektur *neural network* dapat dilihat pada Gambar 2.18.

Neuron memproses *input* dengan memberikan bobot khusus pada setiap input, yang mengindikasikan signifikansi masing-masing *input* bagi *neuron* tersebut. *Output* dari *neuron* dihasilkan dengan menjumlahkan semua *input* yang diberi bobot dan menerapkannya pada fungsi aktivasi tertentu. Bobot dapat dianggap sebagai parameter yang disesuaikan oleh jaringan saraf selama proses pelatihan.



Gambar 2.18 Konsep arsitektur *neural network* (Sheehan dan Song, 2016)

Selanjutnya ada istilah yang disebut fungsi aktivasi. Fungsi aktivasi adalah fungsi matematika yang diterapkan pada keluaran suatu *neuron* dalam *neural network*. Tujuan dari fungsi aktivasi adalah untuk mengenonaktifkan atau mengatur tingkat aktivasi suatu *neuron* berdasarkan masukan yang diterimanya (Prince, 2024). Ilustrasi penggunaan jenis-jenis fungsi aktivasi pada *neural network* seperti yang ditunjukkan pada Gambar 2.19.



Gambar 2. 19 Ilustrasi penggunaan fungsi aktivasi pada *neural network* (Xmllmx, 2017)

Beberapa fungsi aktivasi yang umum digunakan dalam *neural network* yaitu:

a. *Sigmoid*. Fungsi *sigmoid* menghasilkan keluaran di rentang (0, 1). Fungsi ini sering digunakan pada lapisan tersembunyi (*hidden layers*) *neural network* yang memiliki beberapa keterbatasan dalam menangani *gradien* yang kecil (*vanishing gradient*) pada jaringan yang sangat dalam.

b. *ReLU* (*Rectified Linear Unit*). *ReLU* adalah fungsi aktivasi yang sangat umum digunakan karena sederhana dan efektif. *ReLU* menghasilkan keluaran 0 untuk nilai masukan yang negatif, dan langsung meneruskan nilai positif masukan

tersebut. *ReLU* membantu mengatasi masalah *gradien* yang menghilang dan mempercepat konvergensi dalam pelatihan jaringan.

c. *TanH* (*Hyperbolic Tangent*). Fungsi *TanH* menghasilkan keluaran di rentang (-1, 1). Mirip dengan *sigmoid*, tetapi memiliki rentang keluaran yang lebih besar (-1 hingga 1) dan simetris terhadap sumbu nol. Fungsi *TanH* sering digunakan dalam lapisan tersembunyi untuk menangani masalah *gradien* yang menghilang.

d. *Softmax*. Fungsi *softmax* digunakan pada lapisan *output* dari *neural network* yang digunakan untuk klasifikasi multikelas. Fungsi ini mengubah keluaran dari lapisan *output* menjadi distribusi probabilitas yang menunjukkan probabilitas relatif dari setiap kelas.

2. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah bentuk *deep learning* yang menggunakan lapisan konvolusi untuk membentuk struktur jaringan saraf yang disusun. Diketahui bahwa penggunaan lapisan konvolusi dalam CNN dapat mengurangi beban komputasi dari jaringan saraf (Widhiyasana dkk., 2021).

Apbila CNN digunakan untuk memproses data teks, maka harus menentukan data teksnya terlebih dahulu. Misal dalam hal ulasan film, terdapat ratusan ulasan teks. Masing-masing terdiri dari beberapa kalimat yang panjang dan berisi pendapat yang panjangnya bervariasi. Jaringan saraf hanya dapat belajar menemukan pola dalam data numerik sehingga, sebelum memasukkan ulasan ke dalam jaringan saraf sebagai masukan, harus mengubah setiap kata menjadi nilai numerik. Proses ini sering disebut pengkodean kata (*word encoding*) atau tokenisasi. Proses pengkodean (*encoding*) yang khas adalah sebagai berikut:

a. Untuk semua data teks, dalam hal ini ulasan film, akan dicatat setiap kata unik yang muncul dalam kumpulan dataset tersebut dan mencatatnya sebagai kosakata (*vocabulary*) pada model.

b. Selanjutnya mengkodekan (*encode*) setiap kosakata sebagai bilangan bulat unik, yang disebut *token*. Seringkali, *token* ini ditetapkan berdasarkan frekuensi kemunculan sebuah kata dalam kumpulan data. Jadi, kata yang paling sering muncul di seluruh kumpulan data, akan memiliki token terkait: 0. Misalnya, jika

kata yang paling umum adalah “the”, kata tersebut akan memiliki nilai token terkait sebesar 0. Maka kata paling umum berikutnya akan menjadi diberi token sebagai 1, dan proses itu berlanjut.

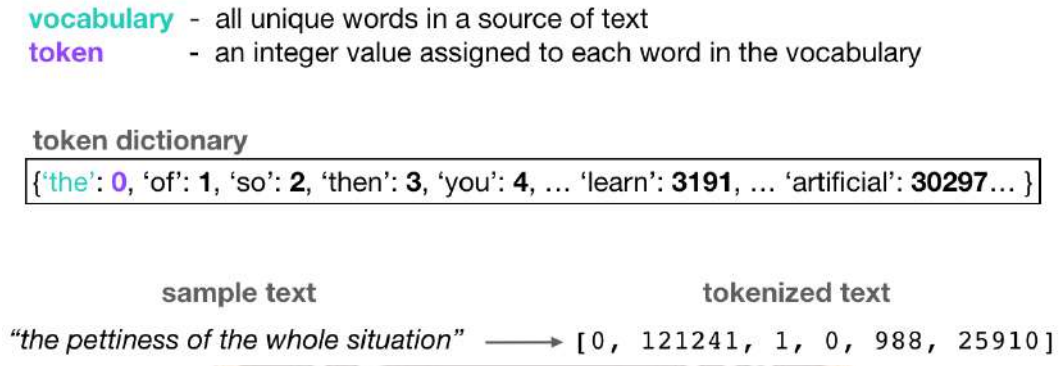
c. Dalam *code*, asosiasi kata-token ini direpresentasikan dalam kamus (*dictionary*) yang memetakan setiap kata unik ke tokennya, nilai *integer*:

{'the': 0, 'of': 1, 'so': 2, 'then': 3, 'you': 4, …}

Seringkali ada begitu banyak kata dalam kumpulan data tertentu sehingga *token* ini berkisar dari nilai 0 hingga 100.000 atau lebih.

d. Terakhir, setelah menetapkan token ini ke setiap kata, kemudian dapat memberi token pada seluruh korpus.

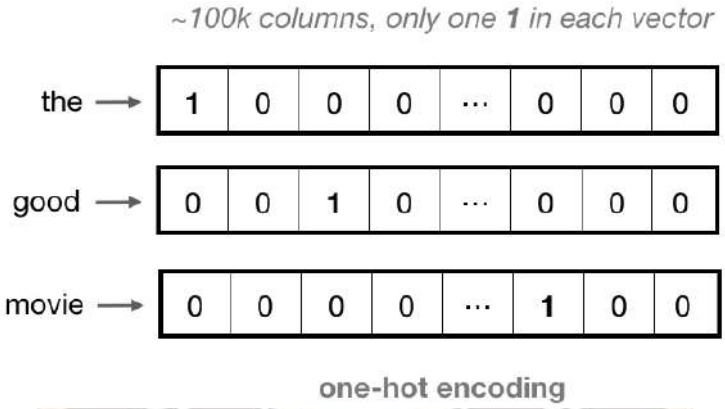
Untuk dokumen apa pun dalam kumpulan data, seperti ulasan film tunggal, akan diperlakukan sebagai daftar kata secara berurutan. Kemudian menggunakan kamus *token* untuk mengubah daftar kata ini menjadi daftar nilai *integer*. Contoh ilustrasi kata dan *token index* dapat dilihat pada Gambar 2.20.



Gambar 2.20 Ilustrasi kata dan *token index* (Camacho, 2019)

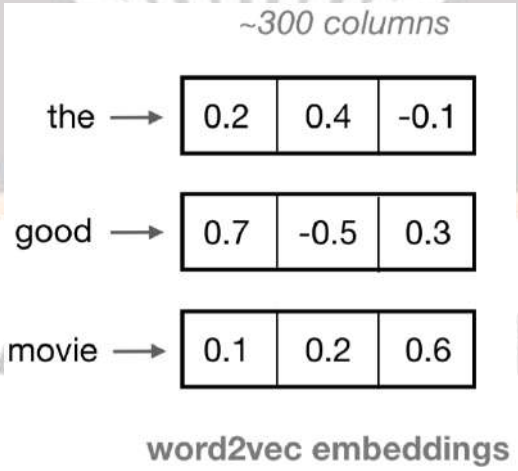
Langkah pengkodean (*encoding*) yang umum adalah dengan mengkodekan setiap token secara *one-hot*, mewakili setiap kata sebagai vektor yang memiliki nilai sebanyak jumlah kata dalam kosa kata. Artinya, setiap kolom dalam vektor mewakili satu kemungkinan kata dalam kosakata. Vektor diisi dengan 0 kecuali indeks pada nilai token kata tersebut, katakanlah indeks 0 untuk “the”. Untuk kosakata yang banyak, vektor ini bisa menjadi sangat panjang, dan berisi semua

angka 0 kecuali satu nilai. Contoh one-hot encoding seperti yang ditunjukkan pada Gambar 2.21.



Gambar 2.21 Contoh *one-hot encoding* (Camacho, 2019)

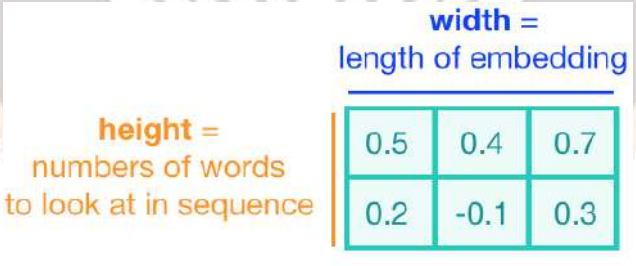
Pada kenyataannya sering kali diinginkan representasi yang lebih padat. Salah satu representasi tersebut adalah vektor kata (*word vector*) yang dipelajari, yang dikenal sebagai *embedding*. Penyematan kata (*word embedding*) adalah vektor dengan panjang tertentu, biasanya pada urutan 100, dan setiap vektor dengan nilai sekitar 100, mewakili satu kata. Nilai di setiap kolom mewakili fitur sebuah kata, bukan kata tertentu. Secara detail mengenai proses cara kerja *word embedding* telah dijelaskan di atas. Contoh penggunaan *word2vec embedding* seperti yang ditunjukkan pada Gambar 2.22.



Gambar 2.22 Contoh *word2vec embedding* (Camacho, 2019)

Selanjutnya pada CNN ada istilah *kernel*. Pada lapisan konvolusional (*convolutional layers*) dirancang untuk menemukan pola spasial (*spatial patterns*) dalam suatu gambar dengan menggeser jendela *kernel* kecil (*small kernel window*) di atas gambar. *Windows* ini sering kali berukuran kecil, mungkin berukuran 3x3 piksel, dan setiap sel kernel memiliki bobot yang terkait. Saat kernel meluncur di atas gambar, piksel demi piksel, bobot kernel dikalikan dengan nilai piksel pada gambar di bawahnya, lalu semua nilai yang dikalikan dijumlahkan untuk mendapatkan keluaran, nilai piksel yang difilter.

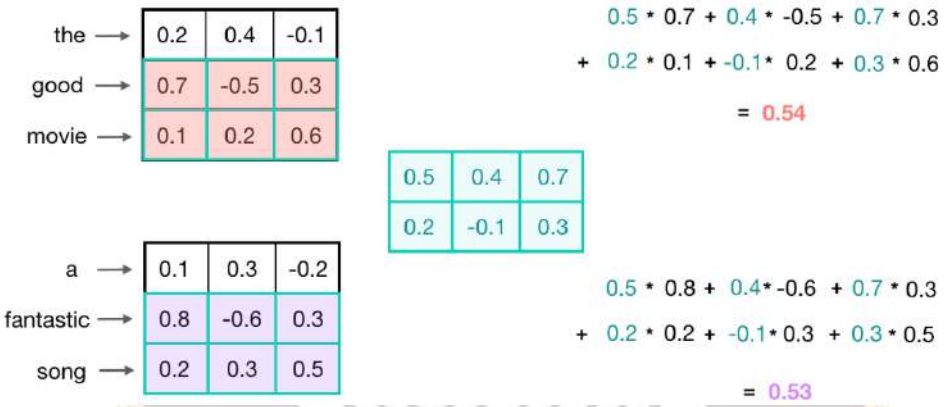
Dalam kasus klasifikasi teks, kernel konvolusional (*convolutional kernel*) akan tetap menjadi jendela geser, hanya tugasnya adalah melihat penyematan (*embeddings*) beberapa kata, bukan area kecil piksel dalam sebuah gambar. Dimensi kernel konvolusional juga harus diubah, sesuai dengan tugas ini. Untuk melihat urutan *word embeddings*, agar jendela melihat beberapa *word embeddings* secara berurutan. *Kernel* tidak lagi berbentuk persegi, melainkan persegi panjang dengan lebar dimensi seperti 3x300 atau 5x300 (dengan asumsi panjang penyematan 300). Ketinggian *kernel* adalah jumlah *embeddings* yang akan dilihatnya sekaligus, mirip dengan representasi *n-gram* dalam model kata. Lebar *kernel* harus mencakup panjang keseluruhan kata yang disematkan (*word embedding*). Ilustrasinya seperti yang ditunjukkan pada Gambar 2.23.



Gambar 2.23 Ilustrasi *kernel* (Camacho, 2019)

Ada aspek positif lain dari operasi konvolusional yang perlu diingat. Perhatikan bahwa kata-kata yang serupa akan memiliki embedding yang serupa, dan operasi konvolusi sebenarnya adalah operasi linier pada vektor-vektor tersebut. Jadi, ketika kernel konvolusional diterapkan pada himpunan kata-kata serupa yang berbeda, hal itu akan menghasilkan nilai keluaran yang serupa.

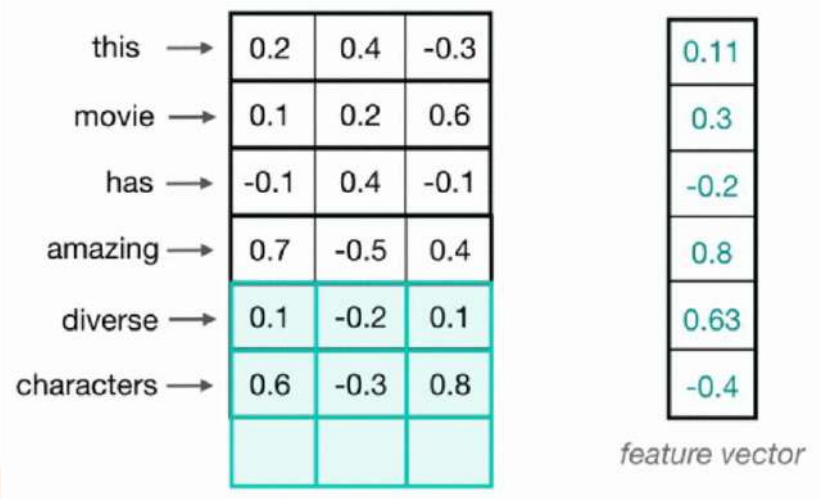
Pada Gambar 2.24 dapat dilihat bahwa nilai keluaran konvolusional untuk pasangan kata “good movie” dan “fantastic song” hampir identik karena *embedding* kata untuk kedua pasangan kata tersebut juga sangat mirip. Dalam contoh ini, kernel konvolusional telah belajar menangkap fitur yang lebih umum, bukan hanya kebaikan film atau lagu, tetapi aspek positif secara umum. Mengenali fitur-fitur tingkat tinggi seperti ini sangat berguna dalam tugas klasifikasi teks yang sering mengandalkan pola umum. Misalnya, dalam analisis sentimen, sebuah model akan menjadi lebih efektif karena mampu merepresentasikan kategori kata yang negatif, netral, dan positif. Sebuah model dapat menggunakan fitur-fitur umum tersebut untuk mengklasifikasikan teks secara keseluruhan.



Gambar 2.24 Contoh ilustrasi penggunaan *convolutional kernel* (Camacho, 2019)

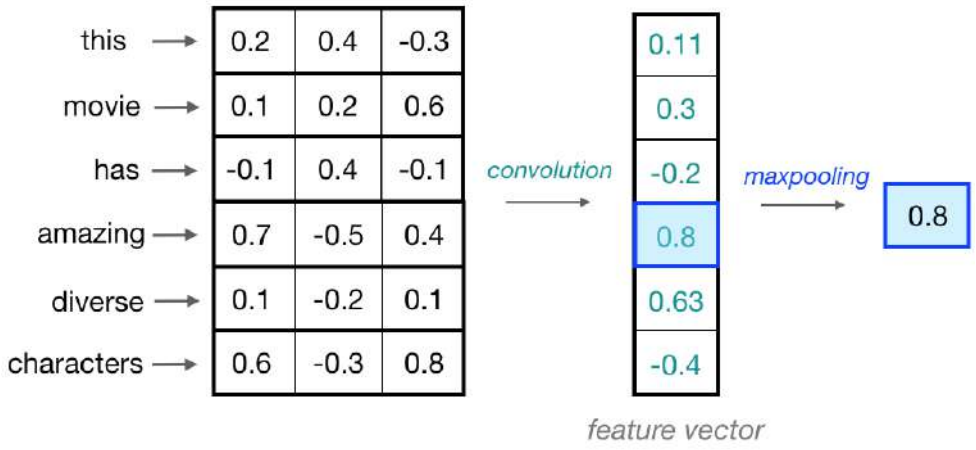
Berikutnya pada CNN ada istilah *1D convolution* (Prince, 2024). Sebelumnya telah diketahui bagaimana kernel konvolusional dapat digunakan pada berbagai embedding kata. Untuk memproses seluruh rangkaian kata, kernel ini akan bergeser melalui daftar embedding kata secara berurutan. Teknik ini dikenal sebagai konvolusi 1D (*1D convolution*) karena kernel hanya bergerak dalam satu dimensi, sepanjang waktu. Sebuah kernel akan berpindah satu per satu melalui daftar embedding input, mengevaluasi embedding kata pertama (dan jendela kecil dari embedding kata berikutnya), kemudian embedding kata berikutnya, dan seterusnya. Hasilnya adalah vektor fitur yang berisi nilai sebanyak jumlah embedding input, sehingga panjang urutan input menjadi krusial, karena kadang-kadang kernel

konvolusional tidak selalu cocok secara sempurna dengan embedding kata, sehingga beberapa tambahan padding mungkin diperlukan untuk menyesuaikan dengan tinggi kernel. Contoh keluaran konvolusi 1D ketika diterapkan pada rangkaian *word embedding* secara pendek seperti yang terlihat pada Gambar 2.25.



Gambar 2.25 Contoh ilustrasi *1D convolution* (Camacho, 2019)

Selanjutnya pada CNN ada istilah *maxpooling* (Prince, 2024). Sebelumnya telah terlihat bagaimana operasi konvolusional menghasilkan vektor fitur yang bisa mewakili dapat mewakili fitur lokal dalam rangkaian *word embedding*. Salah satu pertimbangan penting adalah bagaimana vektor fitur terlihat saat diterapkan pada frasa penting dalam teks sumber. Misalnya, ketika akan mengklasifikasikan ulasan film, jika menemukan frasa “great plot”, tidak masalah di mana frasa tersebut muncul dalam ulasan, dan bisa menjadi indikator positif bahwa ulasan tersebut positif, tidak peduli posisinya dalam teks. Untuk menunjukkan keberadaan fitur-fitur tingkat tinggi ini, diperlukan cara untuk mengidentifikasinya dalam satu vektor, tanpa memperhatikan urutan masukan yang lebih besar. Salah satu pendekatan untuk mengidentifikasi fitur-fitur penting terlepas dari urutannya adalah dengan menghilangkan informasi lokasi yang kurang relevan. Untuk melakukannya bisa menggunakan operasi *maxpooling*, yang memaksa jaringan untuk mempertahankan hanya nilai maksimum dalam vektor fitur, yang seharusnya merupakan fitur lokal yang paling berguna. Contoh ilustrasi operasi *maxpooling* seperti yang ditunjukkan pada Gambar 2.26.

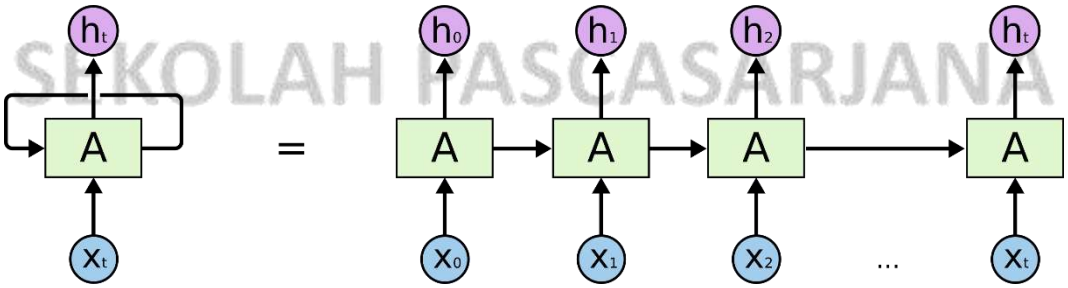


Gambar 2.26 Contoh ilustrasi *maxpooling* (Camacho, 2019)

3. *Recurrent Neural Network*

Recurrent Neural Network (RNN) adalah jenis arsitektur jaringan saraf yang dirancang khusus untuk menangani urutan data atau data sekuensial (Prince, 2024). RNN sangat cocok untuk memproses data yang memiliki struktur urutan waktu, seperti teks, audio, atau video. Apa yang membedakan RNN dari jaringan saraf biasa adalah kemampuannya untuk "mengingat" atau "memiliki memori" dari informasi yang sudah diproses sebelumnya dalam urutan. RNN memiliki sifat yang menyimpan status internal atau "memori" yang memungkinkannya untuk mempertimbangkan informasi kontekstual dari urutan data yang lebih panjang.

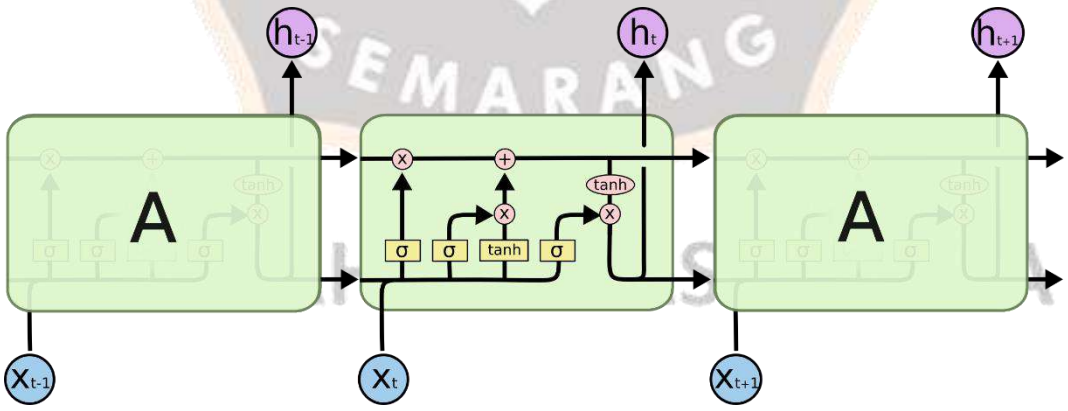
Arsitektur RNN terdiri dari unit rekurensi yang mengambil input pada waktu tertentu dan menghasilkan output serta mengirimkan informasi ke unit berikutnya dalam urutan. Dalam setiap langkah waktu, unit rekurensi menggunakan input saat ini bersama dengan informasi dari langkah waktu sebelumnya untuk menghasilkan output. Contoh arsitektur RNN seperti yang terlihat pada Gambar 2.27.



Gambar 2.27 Contoh ilustrasi arsitektur *Recurrent Neural Network* (Olah, 2015)

Meskipun RNN memiliki kemampuan memodelkan dependensi jangka panjang dalam data sekuensial, mereka memiliki beberapa kendala, seperti kesulitan dalam mempertahankan informasi jangka panjang dan masalah *vanishing gradient* saat melatih model. Untuk mengatasi beberapa kendala ini, variasi RNN seperti *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) dikembangkan, yang memungkinkan model untuk mempertahankan informasi dalam jangka waktu yang lebih lama dan mengurangi masalah gradien yang hilang.

Long Short-Term Memory (LSTM) adalah jenis RNN khusus yang mampu mempelajari dependensi jangka panjang (Cendani dkk., 2023). Prinsip-prinsip tersebut bekerja dengan sangat baik pada berbagai macam masalah, dan sekarang digunakan secara luas. LSTM secara eksplisit dirancang untuk menghindari masalah ketergantungan jangka panjang. Mengingat informasi untuk jangka waktu yang lama pada dasarnya merupakan perilaku default mereka, bukan sesuatu yang sulit mereka pelajari. Semua jaringan saraf berulang berbentuk rantai modul jaringan saraf berulang. Dalam RNN standar, modul berulang ini akan memiliki struktur yang sangat sederhana, seperti lapisan tanh tunggal. LSTM juga memiliki struktur seperti rantai, tetapi modul berulang memiliki struktur yang berbeda. Alih-alih memiliki satu lapisan jaringan saraf, ada empat lapisan, yang berinteraksi dengan cara yang sangat khusus. Contoh ilustrasi arsitektur LSTM seperti yang ditunjukkan pada Gambar 2.28.



Gambar 2.28 Contoh ilustrasi arsitektur *Long Short-Term Memory* (Olah, 2015)

2.3.7 Tes Konsistensi

Hasil pelabelan data dengan melibatkan dua pelabel/rater perlu adanya tes konsistensi. Variasi antar rater dapat diukur dalam situasi apa pun di mana dua atau lebih rater independen mengevaluasi hal yang sama. Metode cohen kappa dapat digunakan untuk mengukur konsistensi antar rater. Penghitungan didasarkan pada perbedaan antara seberapa banyak kesepakatan yang benar-benar ada (kesepakatan yang diamati) dibandingkan dengan seberapa banyak kesepakatan yang diharapkan muncul secara kebetulan saja (kesepakatan yang diharapkan) (Viera dan Garrett, 2005). Metode cohen kappa (Bujang dan Baharum, 2017; Pérez dkk., 2020; Rau dan Shih, 2021; Viera dan Garrett, 2005) seperti pada persamaan 2.3.

$$k = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \qquad (2.3)$$

Pr(a) = Persentase jumlah pengukuran yang konsisten di antara penilai

Pr(e) = Persentase jumlah perubahan pengukuran antara penilai

Visualisasi interpretasi/penafsiran cohen kappa seperti terlihat pada Tabel 2.4.

Tabel 2.4 Interpretasi kappa.

	Poor	*Slight*	*Fair*	*Moderate*	*Substantial*	*Almost Perfect*
Kappa	0.0	.20	.40	.60	.80	1.0

Keterangan:

Kappa	Agreement
<0	Less than chance agreement
0.01-0.20	Slight agreement
0.21-0.40	Fair agreement
0.41-0.60	Moderate agreement
0.61-0.80	Substantial agreement
0.81-0.99	Almost perfect agreement