BAB II
TINJAUAN PUSTAKA

2.1 Pendahuluan

Literature review tidak hanya bermakna membaca literatur, tapi lebih ke arah evaluasi yang mendalam dan kritis tentang penelitian sebelumnya pada suatu topik. *Literature review* yang baik adalah yang melakukan evaluasi terhadap kualitas dan temuan baru dari suatu paper ilmiah [15]. Perlu dipahami bahwa yang disebut dengan literatur ilmiah dapat berupa:

1. Paper dari *Journal* Ilmiah
2. Paper dari *Conference* (Proceedings)
3. *Thesis* dan Disertasi
4. *Report* (Laporan) dari Organisasi yang Terpercaya
5. Buku *Textbook*

Manfaat dari *literature review* yang kita lakukan diantaranya adalah sebagai berikut. Tanda dalam kurung saya berikan untuk memberi *clue* tentang jenis literatur jenis apa yang kita gunakan.

1. Memperdalam pengetahuan tentang bidang yang diteliti (**Buku *Textbook***)
2. Mengetahui hasil penelitian yang berhubungan dan yang sudah pernah dilaksanakan (*related research*) (**Paper**)
3. Mengetahui perkembangan ilmu pada bidang yang kita pilih (*state-of-the-art research*) (Paper)
4. Memperjelas masalah penelitian (*research problems*) (**Paper**)
5. Mengetahui metode-metode terkini yang diusulkan para peneliti untuk menyelesaikan masalah penelitian (*state-of-the-art methods*) (**Paper**)

Setelah melakukan *literature review*, peneliti tidak berhenti sampai hanya *membaca literatur*, tetapi juga merangkumkan, membuat analisis dan melakukan sintesis secara kritis dan mendalam dari paper-paper yang direview atau ditinjau.

Salah satu jenis dan metode yang digunakan para peneliti untuk melakukan *literature review* atau tinjauan pustakan dan kemudian merangkumnya ke dalam suatu paper, yaitu

- S*ystematic Literature Review* atau *Systematic Review*

  Systematic literature review atau sering disingkat SLR atau dalam bahasa indonesia disebut tinjauan pustaka sistematis adalah metode *literature review* yang mengidentifikasi, menilai, dan menginterpretasi seluruh temuan-temuan pada suatu topik penelitian, untuk menjawab pertanyaan penelitian (*research question*) yang telah ditetapkan sebelumnya [16]. Metode SLR dilakukan secara sistematis dengan mengikuti tahapan dan protokol yang memungkinkan proses *literature review* terhindar dari bias dan pemahaman yang bersifat subyektif dari penelitinya.

Tabel 2.1 Studi Literatur Penelitian Terdahulu

Penulis	**Masalah**	**Dataset**	**Metode**	**Tahun Jurnal**	**Akurasi Model**
Tavishee Chauhan, M.E., Hemant Palivela, PhD	Metode *neural network* tidak dapat memprediksi kata berdasarkan informasi lampau yang disimpan dalam jangka waktu lama dan data bersifat *time series*.	Deteksi Berita Palsu: *Fake and Real news dataset* dan *pre-trained data GloVe Twitter*	*Neural network : deep learning LSTM model* dan *Embedding layer GloVe*	2021	99.88%
Feyza Altunbey Ozbay, Bilal Alatas	Metode *neural network* tidak optimal untuk data teks sehingga memerlukan metode pengoptimalan berdasarkan koefisien penurunan nonlinier dan inersia berosilasi *weight*	Deteksi Berita Palsu: *BuzzFeed political news data set*	*Improved Salp Swarm Optimization (SSO) and GWO (Grey Wolf Optimization)*	2021	99.50%

Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang	Metode *neural network* tidak dapat memprediksi kata berdasarkan informasi lampau sehingga dapat menyebabkan ambiguitas pada pemahaman bahasa alami.	Deteksi Berita Palsu: *real-world fake news dataset*	*Deep learning models CNN dan LSTM, FakeBERT*	2021	98.90%
Chatat Raj, Priyanka Meel	Framework yang didedikasikan untuk deteksi berita palsu visual sangat sedikit, sehingga membutuhkan framework multi-modal menggunakan CNN	Deteksi Berita Palsu: Ti-CNN (Web Scraping data teks dan gambar berita palsu)	*Proposed Coupled ConvNet (composed of Text-CNN module for textual fake news classification)*	2021	93.56%
Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang	Metode *neural network* sederhana dinilai tidak efektif untuk klasifikasi berita palsu, sehingga diperlukan efisiensi berupa faktorisasi tensor	Deteksi Berita Palsu: *BuzzFeed* dan *PolitiFact*	*A coupled matrix–tensor factorization approach, EchoFakeD (efficient deep neural network)*	2021	92.30%
Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang	Metode *neural network* pada penelitian deteksi berita palsu masih menggunakan teknik tradisional sehingga perlu peningkatan akurasi model.	Deteksi Berita Palsu: *BuzzFeed and PolitiFact dataset*	*Machine learning model: XGBoost algorithm, DeepFake: multi-layer deep neural network*	2020	88.64%

Pada Tabel 2.1 memuat penelitian sebelumnya yang terkait dengan objek penelitian deteksi berita palsu dengan metode model *neural network* untuk menyelesaikan kasus pemrosesan bahasa alami atau NLP. Berdasarkan masalah yang disampaikan pada Tabel diatas, menunjukkan beberapa kekurangan arsitektur model *neural network* dalam mengatasi data yang bersifat teks dan *time series*. Metode *state of the art* yang digunakan pada penelitian-penelitian tersebut

merupakan langkah peneliti dalam mengatasi kekurangan model *neural network* ini menggunakan berbagai metode optimasi model.

Pada penelitian [1], masalah yang dihadapi adalah metode *neural network* tidak dapat memprediksi kata berdasarkan informasi lampau yang disimpan dalam jangka waktu lama dan data bersifat *time series* sehingga sulit di proses model *neural network*. Maka dari itu, untuk memperbaiki metode *neural network* agar lebih optimal, Penulis menggunakan metode yang diusulkan, yaitu *deep learning* LSTM karena metode ini sangat baik untuk menyimpan data lebih lama dan cocok untuk data berbasis teks dan *time series*. Penelitian ini menggunakan dataset publik yang tersedia di situs penyedia dataset di kaggle, yaitu *Fake and Real news dataset* yang merupakan dataset jenis teks, *time series*, dan NLP. Metrik pengukuran yang digunakan adalah akurasi, dan *confussion matrix*, namun metrik pengukuran yang diambil dan dibandingkan hanya akurasi model. Untuk hasil evaluasi model ini, akurasi yang didapat adalah 99.88% untuk akurasi latih dan untuk akurasi tes nya diperkirakan dibawah dari akurasi latih serta menjadikan metode ini menjadi metode paling mutakhir untuk deteksi berita palsu saat ini.

Pada penelitian ini [11], masalah yang ada sama seperti penelitian sebelumnya, yaitu optimasi metode *neural network*, menggunakan *Improved Salp Swarm Optimization (SSO)* dan *GWO (Grey Wolf Optimization)* untuk menemukan solusi optimal terbaik untuk mendeteksi berita palsu berdasarkan koefisiensi penurunan nonlinier dan bobot inersiaberosilasi. *Dataset* yang digunakan untuk deteksi berita palsu adalah *BuzzFeed political news dataset*. Akurasi model yang didapat dari model optimasi ini adalah 99.50%.

Pada penelitian [8], metode yang digunakan untuk mengatasi masalah model *neural network*, adalah *deep learning models CNN dan LSTM* yang disebut metode *FakeBERT* (*Bidirectional Encoder Representations from Transformers*). *Dataset* yang digunakan untuk deteksi berita palsu adalah *real-world fake news dataset* yang didapatkan dari *Google Drive* milik Penulis. Akurasi model yang didapat dari metode model ini adalah 98.90%.

Penggunaan metode optimasi juga bisa digunakan menggunakan memodifikasi algoritma *neural network* nya, seperti penggunaan metode jaringan CNN yang digunakan oleh penelitian [12]. Pada penelitian ini, menggunakan *multi-*

modal CNN untuk memajukan penelitian yang sedang berlangsung pada deteksi berita palsu berbasis model CNN, modul tekstual dan visual dikembangkan untuk memeriksa kinerja pada kumpulan data multimodal. Pola laten dieksploitasi dan ditemukan dalam teks dan gambar menggunakan lapisan konvolusi. Penulis menawarkan arsitektur berlapis *ConvNet* multi-model, dengan menggabungkan kedua modul data, mengkategorikan berita online secara efisien berdasarkan konteks tekstual dan visualnya. *Dataset* pada penelitian ini didapat dari hasil *web scraping* untuk data teks dan gambar mengenai berita palsu. Sekitar 20015 item berita dari *websites* dan 5733 data gambar berita palsu yang diekstrak dari TICNN. Dengan menggunakan *framework* ini, akurasi yang didapat untuk modul metode *Text-CNN* adalah 93.56%.

Selain itu, pada penelitian sebelumnya terdapat metode efisiensi *deep neural network* yang digunakan pada penelitian [10] menggunakan pendekatan *matrix-tensor factorization* untuk meningkatkan efisiensi model *neural network* dimana model pembelajaran mesin tradisional biasanya menggunakan teknik faktorisasi normal untuk mendeteksi berita palsu karena sifatnya yang *unsupervised*. Dengan menggunakan *dataset* publik untuk deteksi berita palsu, yaitu *BuzzFeed* dan *PolitiFact*, akurasi model yang didapat adalah 92.30%.

Metode *neural network* dapat ditingkatkan dengan menggunakan algoritma *boosting* [17] seperti yang digunakan pada penelitian [9] yang menggunakan metode model *XGBoost* dan *multi-layer deep neural network*. Metode deteksi yang ada bergantung pada data berbasis pengguna tertentu serta konten berita atau konteks sosial. Peneliti mempertimbangkan isi berita surat kabar dan keberadaan *echo chambers* (kelompok pengguna media sosial yang memiliki pendapat yang sama) di jejaring sosial saat mengidentifikasi berita palsu. Dengan menggunakan *dataset* yang sama dengan penelitian [10], yaitu *BuzzFeed* dan *PolitiFact*, akurasi yang didapat sebesar 88.64%.

Tingkat efektifitas metode terhadap optimasi model *neural network* di lima penelitian sebelumnya diungguli oleh sebuah penelitian yang menggunakan optimasi dan peningkatan model untuk deteksi berita palsu menggunakan metode optimasi LSTM pada penelitian [1] dan menjadi penelitian dengan metode optimasi paling mutakhir untuk deteksi berita palsu yang akurat.

2.2 *Natural Language Processing* (NLP)

Pemrosesan bahasa alami adalah sekumpulan metode untuk membuat bahasa manusia dapat diakses oleh komputer. Selama dekade terakhir, pemrosesan bahasa alami telah ada di mana-mana dalam kehidupan kita sehari-hari: terjemahan mesin umum di web dan media sosial; klasifikasi teks mencegah email berantakan di bawah tumpukan *spam*; mesin pencari telah bergerak melampaui pencocokan string dan analisis jaringan ke tingkat kompleksitas linguistik yang tinggi; Sistem dialog menyediakan cara yang semakin populer dan efektif untuk mengumpulkan dan berbagi informasi. Aplikasi yang berbeda ini didasarkan pada kumpulan ide yang sama, menggunakan algoritme, linguistik, logika, statistik, dan lainnya [18].

2.2.1 Pengenalan *Natural Language Processing* (NLP)

Pemrosesan Bahasa Alami atau NLP Pemrosesan Bahasa Alami adalah subbidang kecerdasan buatan (AI) untuk memproses, menganalisis, memahami, dan menghasilkan bahasa manusia [19]. NLP adalah subbidang AI karena pemrosesan bahasa dianggap sebagai bagian dari kecerdasan manusia. Penggunaan bahasa adalah keterampilan terpenting yang membedakan manusia dari hewan lain.

NLP mencakup banyak algoritma, metode, dan pemecahan masalah dengan menggunakan teks sebagai masukan. NLP menghasilkan informasi seperti label, representasi semantik, dll., sebagai *output*. Teknik NLP digunakan dalam semua aplikasi cerdas yang berkaitan dengan bahasa alami dan merupakan elemen penting dalam berbagai aplikasi perangkat lunak yang digunakan dalam kehidupan sehari-hari [20].

2.2.2 Teknik *Natural Language Processing*

Seperti yang telah dijelaskan sebelumnya, sebuah aplikasi NLP menerapkan berbagai teknik untuk menyelesaikan permasalahan yang berbeda. Berikut adalah pembahasan secara singkat beberapa tugas inti atau teknik NLP yang sering digunakan dalam aplikasi NLP.

2.2.2.1 Pembuatan Teks (*Text generation)*

Pembuatan teks, disebut juga *Natural Language Generation* (NLG) adalah proses menghasilkan keluaran teks bahasa alami dari bahasa lain. Teknik ini juga sering disebut sebagai *text-to-text generation*. Sebagai contoh, mesin penerjemah, yang telah kita bahas sebelumnya merupakan salah satu penerapan pembuatan teks. Selain itu, penyederhanaan teks, pembuatan ringkasan, serta koreksi kesalahan tata bahasa juga merupakan contoh penerapan dari teknik ini. Semua tugas ini menghasilkan teks bahasa alami sebagai keluaran.

2.2.2.2 Pencarian Informasi (*Information Retrieval)*

Pencarian informasi memiliki arti yang sangat luas. Membuka kontak dalam ponsel untuk menemukan nomor telepon seseorang adalah bentuk pencarian informasi. Namun, dalam bidang akademis, pencarian informasi dapat didefinisikan sebagaiberikut:

Information retrieval (IR) adalah proses menemukan informasi (biasanya berupa dokumen atau teks) yang relevan dengan kebutuhan. Informasi ini diambil dari kumpulan sumber informasi yang tersedia (biasanya di komputer atau internet) [21].

2.2.2.3 Pemodelan Bahasa

Dalam pemodelan bahasa, kami memprediksi kata berikutnya dalam sebuah kalimat berdasarkan sejarah kata sebelumnya. Tujuannya adalah untuk mempelajari urutan yang mungkin dari kata-kata yang muncul. Pemodelan bahasa digunakan untuk menemukan solusi untuk berbagai masalah, termasuk pengenalan suara, pengenalan tulisan tangan, terjemahan mesin, dan koreksi ejaan atau tata bahasa.

2.2.2.4 Klasifikasi Teks

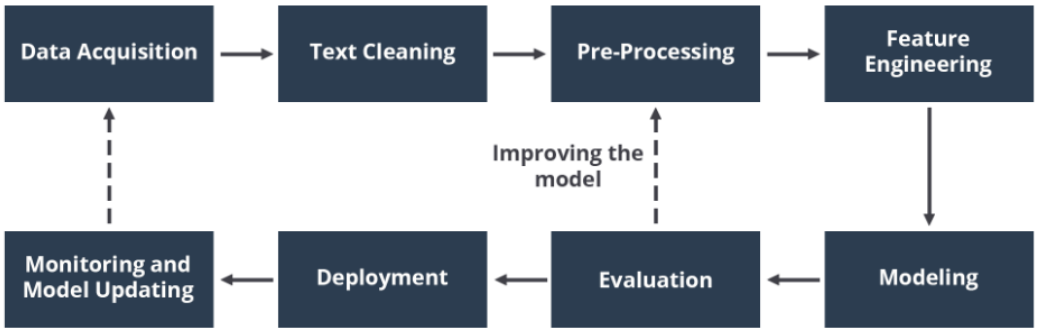
Ini adalah proses membagi teks ke dalam kategori tertentu berdasarkan konten. Klasifikasi teks adalah teknik paling populer di NLP dan digunakan dalam berbagai aplikasi seperti deteksi spam email, analisis sentimen, dan deteksi berita palsu.

2.2.3 *Natural Language Processing Pipeline*

Proyek NLP juga memiliki beberapa tahapan mulai dari pengumpulan data hingga implementasi dan pemantauan model. Beberapa langkah berlaku untuk semua *pipeline* ML, sementara langkah lainnya spesifik dan khusus untuk NLP.

Ingatlah bahwa setiap contoh proyek pembelajaran mesin adalah unik. Namun, tentu saja ada alur atau prosedur yang biasa digunakan dalam *pipeline* NLP. Nah, memahami prosedur umum dari sebuah proyek NLP akan memudahkan kita dalam mengerjakan proyek NLP kedepannya.

Secara umum, komponen utama proyek pengembangan sistem NLP adalah sebagai berikut [20].



Gambar 2.1 Tahapan *Pipeline* Pada Proyek NLP [20]

Langkah pertama dalam proses pengembangan sistem NLP adalah proses pengumpulan data. Biasanya, data yang berhasil dikumpulkan tidak selalu bersih. Di sinilah proses pembersihan teks berperan. Setelah dibersihkan, data teks seringkali masih memiliki banyak variasi dan perlu diubah menjadi bentuk standar. Pada tahap inilah kita melakukan pra-pemrosesan atau *pre-processing*.

Selanjutnya masuk ke tahapan rekayasa fitur, tujuannya adalah menemukan indikator yang cocok untuk tugas kita. Indikator-indikator ini diubah menjadi format yang dapat dimengerti oleh algoritma pemodelan.

Seperti proyek *machine learning* pada umumnya, tahapan selanjutnya adalah fase pemodelan, evaluasi, penerapan model dalam produksi (*deployment*), *monitoring* dan pembaruan model. Tahapan ini kurang lebih sama seperti alur proyek lain.

Namun, prosesnya di dunia asli mungkin tidak selalu linier seperti yang ditunjukkan pada gambar di atas. Sering kali, NLP *pipeline* melibatkan beberapa

tahapan bolak-balik dan perulangan (*loop*). Perulangan yang paling sering adalah dari tahap evaluasi ke tahap pra-pemrosesan, rekayasa fitur, pemodelan, dan kembali ke evaluasi. Ada juga perulangan secara keseluruhan mulai dari proses monitoring, akuisisi data, dst. Perulangan ini biasanya terjadi di tingkat proyek.
Pada fase awal, sebagian besar waktu digunakan dalam pemodelan dan evaluasi, sedangkan setelah sistem matang, rekayasa fitur membutuhkan waktu lebih lama. Berikut adalah beberapa tahapan yang spesifik untuk NLP saja, antara lain:

2.2.4.1 Akusisi Data

Data adalah inti dari setiap sistem *machine learning*. Keberhasilan suatu sistem *machine learning* sangat dipengaruhi oleh kualitas dan jumlah data yang dimiliki oleh sistem. Sehingga, tugas kita adalah mengumpulkan, mengamati, menganalisis, dan memproses data terlebih dahulu sebelum memutuskan untuk memasukkannya ke dalam sistem.
Terdapat beberapa cara akuisisi data, antara lain:

A. Menggunakan Data Publik

Bagi Anda yang telah lama bekerja di bidang data, tentu tidak asing lagi dengan situs penyedia data seperti *Kaggle*, *UCI*, mesin pencari *dataset* Google, maupun *GitHub*. Selain sebagai *tools version control* dan ruang kolaborasi, para *developer* menggunakan *GitHub* untuk berbagi informasi mengenai berbagai sumber *dataset* untuk NLP. Contohnya adalah repository dari *Nicolas Iderhoff* dan *IndoNLU*.

Selain itu, setiap layanan *cloud* seperti *Google Cloud Service*, *Azure*, maupun *AWS* juga menyediakan *dataset* publik yang bisa diakses oleh para penggunanya. Jika menemukan *dataset* publik yang relevan dan sesuai dengan pekerjaan yang sedang kita lakukan, gunakan *dataset* tersebut, buat modelnya, dan lakukan evaluasi.

B. Mengumpulkan Data dari Internet (*Web Scraping*)

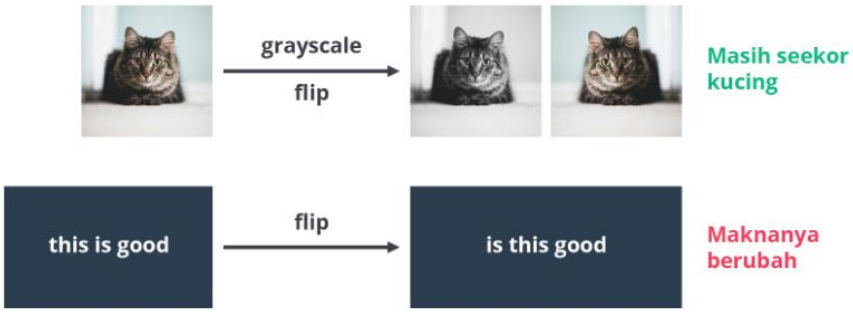
Internet merupakan sumber data yang melimpah. *Web scraping* adalah metode untuk memproses dan mengumpulkan data dari berbagai situs di

internet. Dengan metode tersebut, Anda dapat mengekstrak konten dan informasi untuk analisis lebih lanjut.

Para praktisi data biasanya melakukan proses scraping pada situs media sosial seperti *twitter*, *wikipedia*, bahkan *e-commerce*. Analisis data dari proses *web scraping* dapat digunakan untuk berbagai tujuan, antara lain: keperluan riset, memantau sentimen pasar, dan mendapatkan informasi mengenai kompetitor. Beberapa *tools* yang dapat digunakan untuk proses ini antara lain *Beautifulsoup, Selenium*, dan *Scrapy*.

C. Data Augmentation

Pada bidang *computer vision*, *image augmentation* adalah teknik standar yang sering digunakan. Sementara itu, pada NLP, *text augmentation* (augmentasi teks) masih jarang digunakan. Operasi pada gambar seperti proses memperbesar, membalik (*flip*) dan memutar (*rotate*) tidak mengubah gambar aslinya. Sehingga, proses transformasi melalui *image augmentation* merupakan teknik penting dalam bidang *computer vision*. Berbeda dengan *computer vision*, transformasi dengan *text augmentation* pada NLP menghadapi beberapa tantangan. Dua di antaranya adalah tantangan perubahan semantik (makna) dan tantangan kompleksitas bahasa yang tinggi seperti adanya konteks pembicaraan.



Gambar 2.2 Tantangan Perubahan Semantik (Makna) [22]

Meskipun demikian, terdapat beberapa pendekatan atau *metode text augmentation* yang bisa dilakukan untuk menghasilkan lebih banyak data. Disadur dari Chaudhary dalam tulisannya yang berjudul "*Data Augmentation for NLP*" [22], berikut adalah beberapa metode *text augmentation*:

D. Substitusi Leksikal (*Lexical Substitution*)

Substitusi leksikal merupakan teknik yang bekerja dengan mengganti kata pada teks tanpa mengubah arti dalam kalimat. Teknik ini sendiri memiliki beberapa pendekatan, dua diantaranya adalah *Thesaurus-based substitution* (substitusi berdasarkan tesaurus) dan *Word-Embeddings Substitution* (substitusi dengan teknik *word-embeddings*).

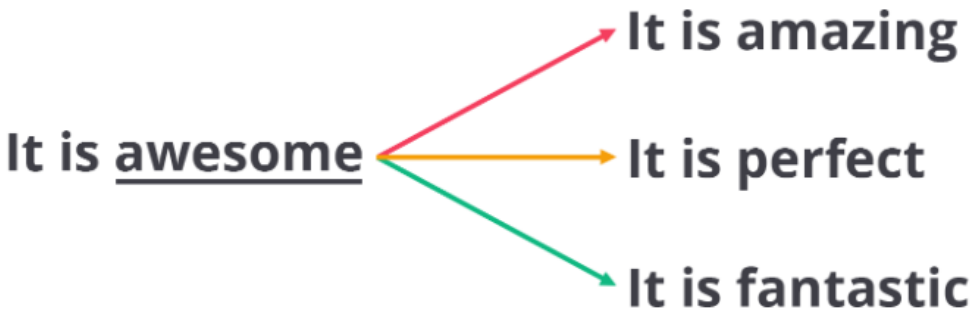
1. *Thesaurus-based substitution*

Dengan teknik ini, kita mengambil kata acak dari kalimat kemudian mengganti dengan sinonimnya menggunakan *tesaurus*. Sebagai contoh, untuk bahasa inggris, kita bisa menggunakan database *WordNet* dalam mencari sinonimnya, kemudian melakukan penggantian. *WordNet* merupakan *database* yang dikurasi secara manual dengan hubungan antar kata.

Gambar 2.3 *Thesaurus-based Substitution* [22]

2. *Word-Embeddings Substitution*

Dalam teknik ini, kita mengambil *pre-trained word embeddings* seperti *Word2Vec*, *GloVe*, *FastText*, *Sent2Vec*. Kemudian, kita menggunakan kata terdekat pada *word embedding* tersebut sebagai pengganti beberapa kata dalam kalimat. Sebagai contoh, kita bisa mengganti kata berikut dengan 3 kata yang paling mirip artinya untuk mendapatkan variasi pada teks.



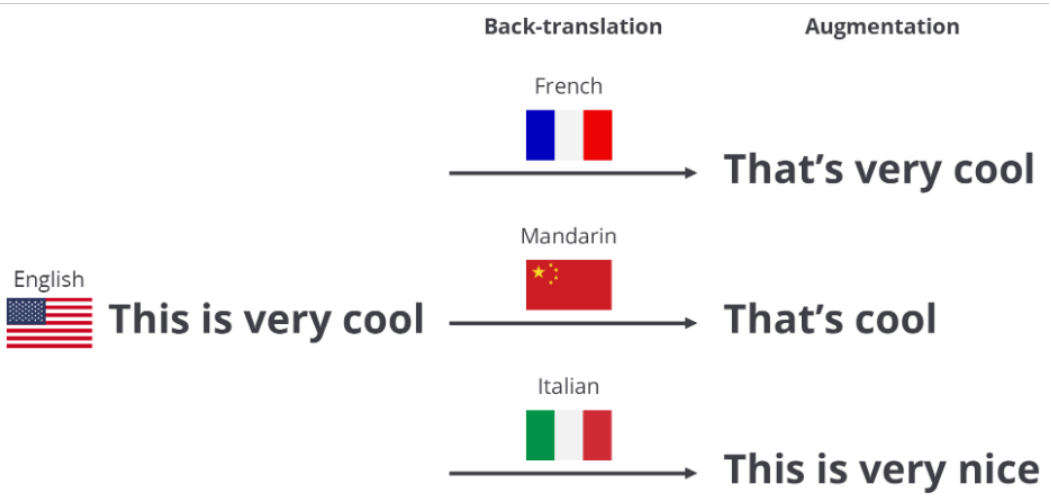
Gambar 2.4 Penggunaan Kata Terdekat Pada *Word Embedding* [22]

E. Back Translation

Dalam teknik ini, kita menggunakan aplikasi mesin penerjemah untuk memparafrasakan teks sambil melatih kembali maknanya. Xie dkk [23] menggunakan metode ini untuk menambah teks yang tidak berlabel dan mempelajari model *semi-supervised* pada *dataset IMDB* dengan hanya 20 contoh berlabel. Model mereka mengungguli model sebelumnya yang dilatih pada 25.000 contoh berlabel.

Tahapan *back-translation* adalah sebagai berikut:

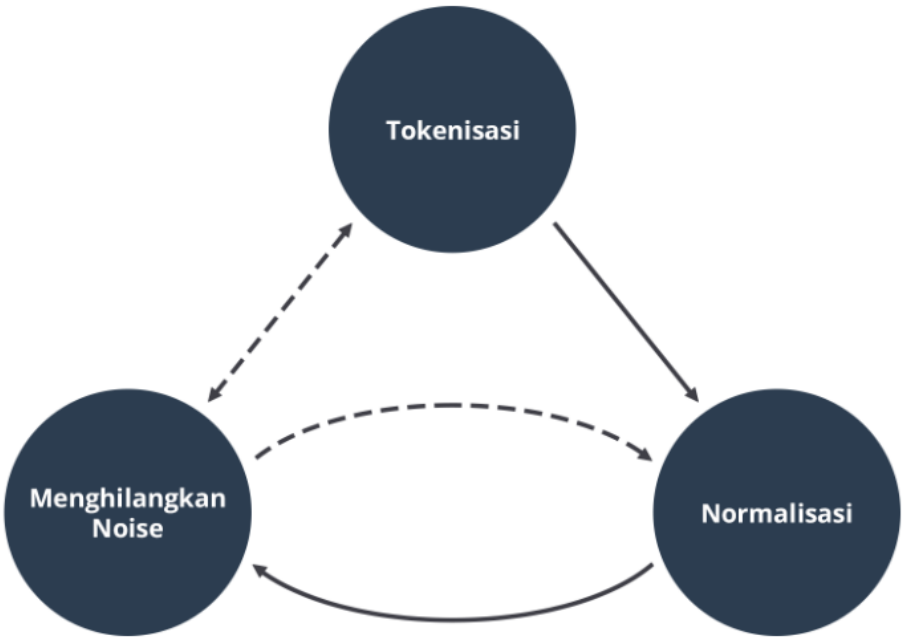
1. Ambil beberapa kalimat (misalnya dalam bahasa Inggris) dan terjemahkan ke bahasa lain, misalnya Perancis
2. Terjemahkan kalimat bahasa Perancis menjadi kalimat bahasa Inggris kembali
3. Periksa apakah kalimat baru berbeda dari kalimat asli. Jika ya, maka kita akan menggunakan kalimat baru ini sebagai versi tambahan dari teks aslinya.
4. Anda juga dapat menjalankan terjemahan balik menggunakan beberapa bahasa yang berbeda sekaligus untuk menghasilkan lebih banyak variasi. Hal ini membuat prosesnya menjadi lebih efektif dan tentu saja dapat menekan sumber daya, seperti pada gambar dibawah ini:



Gambar 2.5 *Back-Translation* Menggunakan Beberapa Bahasa Yang Berbeda [22]

2.2.4.2 *Text Cleaning dan Pre-Processing*

Pembersihan teks dan *pre-processing* merupakan tahapan *data wrangling* dalam proyek NLP. Ini adalah proses mengekstrak teks mentah (*raw text*) dari data masukan, menghapus semua informasi yang tidak diperlukan, dan mengonversi teks ke dalam format yang dibutuhkan. Sehingga, teks siap dimasukkan ke dalam sistem NLP.

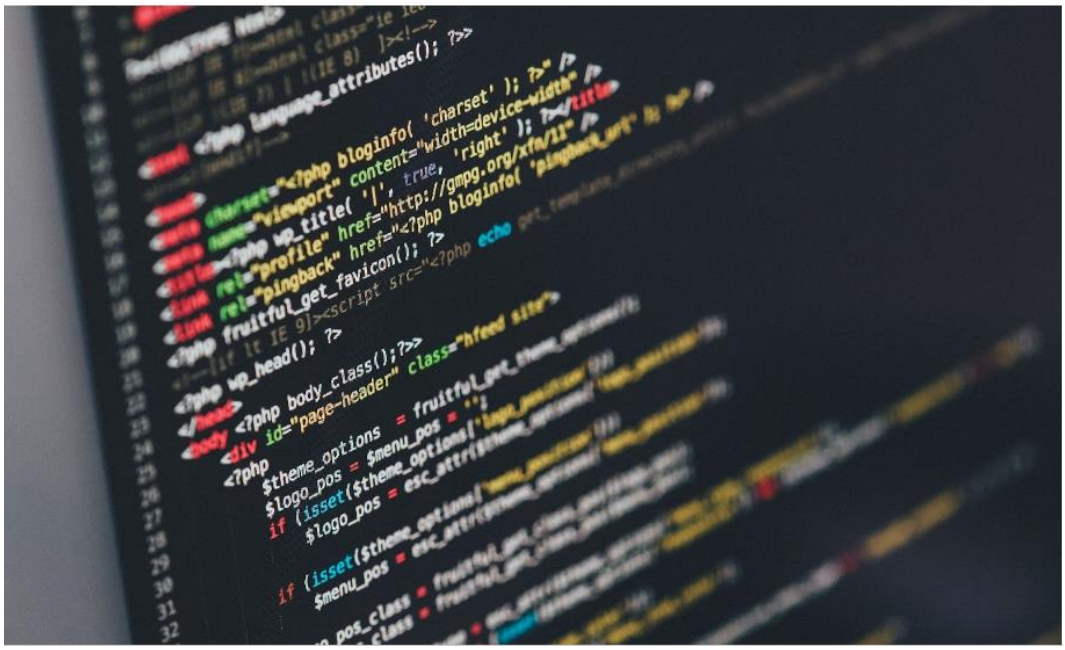


Gambar 2.6 Tiga Komponen Utama Pada Tahap *Text Cleaning* Dan *Pre-Processing* [24]

Secara umum, terdapat tiga komponen utama pada tahap ini, yaitu:

A. Menghilangkan *Noise* (gangguan) pada Teks

Bayangkan Anda melakukan proses *scraping* dari situs wikipedia. Setelah mendapatkan datanya, Anda menyadari data tersebut dalam format html. Isi data mengandung banyak *noise* yang tidak relevan dengan fitur yang ingin Anda ekstrak dari teks. Anda dapat menghapus semua *noise* (informasi yang tidak perlu) tersebut agar menjadi teks yang utuh maknanya.



Gambar 2.7 *Noise* Pada Data Hasil *Web Scraping* Dalam Format Html [24].

Beberapa tahapan untuk menghapus *noise* antara lain:

1. Remove HTML Tag

Teks tidak terstruktur biasanya mengandung banyak *noise*. Apalagi jika Anda menggunakan teknik seperti *web scraping* atau *screen scraping* untuk mengambil data dari halaman web. Tag *HTML* dan *JavaScript* biasanya tidak memiliki makna untuk proses analisis teks. Tujuan kita adalah mengekstrak konten tekstual yang bermakna dari data yang diambil dari web. Oleh karena itu, kita perlu melakukan proses penghapusan tag html.

2. Spelling Correction

Pengetikan cepat pada keyboard komputer maupun layar ponsel cerdas berpotensi memiliki kesalahan ejaan Penulisan. Hal ini sangat lazim terjadi. Jika

dibiarkan, ini bisa menjadi permasalahan yang dapat merusak pemahaman linguistik data.

3. *Remove Stop Words*

Stopwords adalah kata-kata bahasa Inggris yang tidak menambahkan banyak makna kalimat. Mereka dengan mudah dihilangkan tanpa mengorbankan makna kalimat. Contoh *stopword* antara lain kata-kata seperti *the, he, dan have* [25]. Kata-kata tersebut sebelumnya telah dikumpulkan dan disimpan dalam sebuah paket bernama *corpus*, yang dapat dilihat di direktori NLTK. Itu dapat diinstal di lingkungan *Python* itu sendiri. *Natural Language Toolkit* atau NLTK adalah kumpulan alat dan aplikasi berbasis Python untuk pemrosesan bahasa alami (NLP) simbolik dan statistik untuk bahasa Inggris. Untuk menghilangkan kata berhenti dari sebuah kalimat, teks dibagi menjadi kata-kata, dan kemudia diperiksa untuk melihat apakah kata tersebut ada dalam daftar *stopwords* NLTK. Jika kata tertentu ada dalam kumpulan *corpus*, kata tersebut kemudia dihilangkan.

4. *Remove Special Character*

Special character (karakter khusus) adalah karakter selain alfanumerik. Karakter ini paling sering ditemukan dalam referensi, komentar, nilai tukar mata uang, dsb. Karakter ini tidak menambah makna pada pemahaman teks sehingga dapat menyebabkan noise ke dalam algoritma. Untuk mengatasi dan menghilangkannya, kita bisa menggunakan *regular expression* atau *regex*.

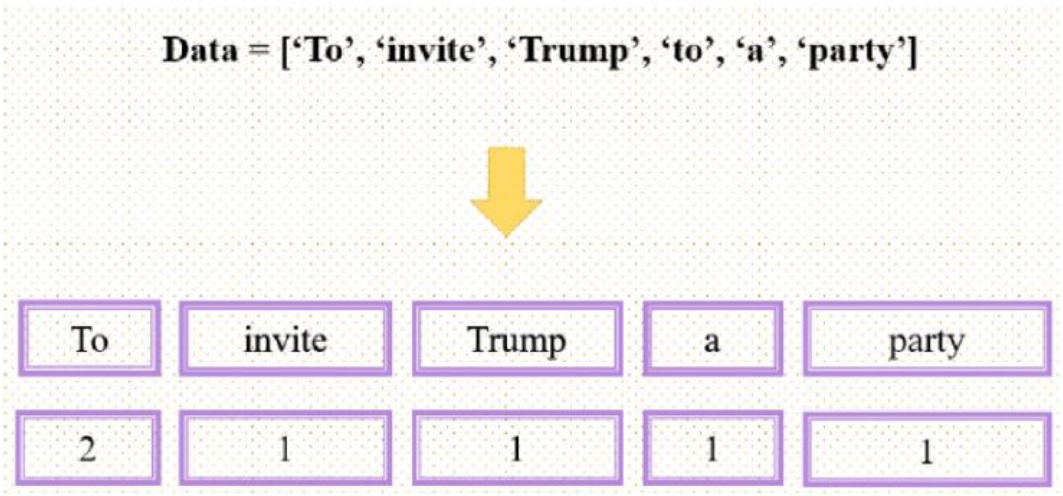
B. Proses Tokenisasi

Tokenisasi adalah proses merepresentasikan setiap kata dalam bentuk angka. Untuk menggunakan data tekstual untuk permodelan prediktif, teks diurai untuk menghilangkan kata-kata tertentu yang dikenal sebagai proses tokenisasi. Kemudian dilakukan ekstraksi fitur atau vektorisasi, yaitu kata-kata diubah menjadi bilangan atau angka *floating-point* untuk digunakan sebagai input dalam metode pembelajaran mesin. Teknik pembelajaran mesin digunakan secara luas di bidang analisis teks. Namun, karena sebagian besar algoritme memerlukan vector fitur numerik dengan ukuran tetap daripada dokumen teks mentah dengan panjang

variabel, data mentah, urutan simbol, tidak dapat diberikan langsung kepada mereka. Berikut ini adalah metode yang paling sering digunakan untuk mengekstraksi fitur numerik dari konten teks:

1) Dengan menggunakan *whitespaces* dan tanda baca sebagai pemisah token, teks dapat diberi token dan menetapkan id integer ke setiap token potensial.
2) Kemunculan token di setiap dokumen dapat dihitung.
3) Token yang muncul di sebagian besar sampel/dokumen dinormalisasi dan diberi bobot dengan berkurangnya kepentingan token.

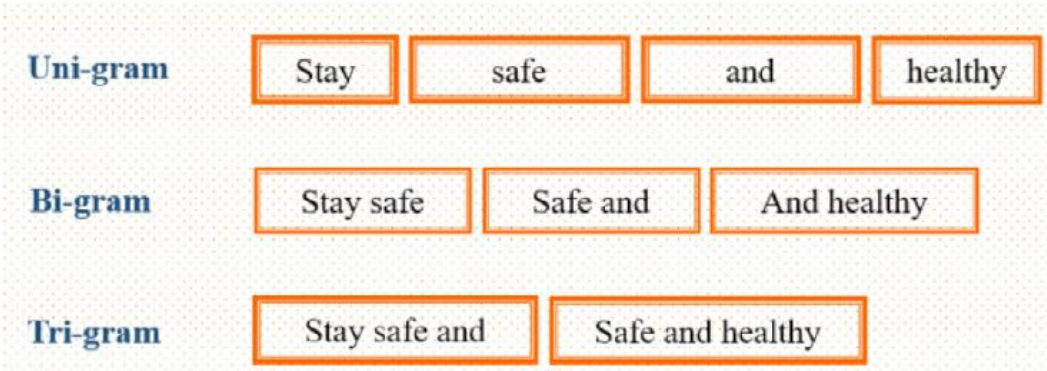
Sampel multidimensi adalah vektor yang berisi semua frekuensi token untuk dokumen tertentu. Matriks yang masing-masing memiliki satu baris dengan dokumen dan satu kolom per token (misalnya kata) yang muncul di *corpus* dapat digunakan untuk mewakili kumpulan dokumen. Proses mengubah kumpulan dokumen teks menjadi vektor fitur numerik disebut sebagai vektorisasi.



Gambar 2.8 Proses Vektorisasi [1]

Representasi *Bag of Words* atau "*Bag of n-grams*" mengacu pada metode khusus ini (Tokenisasi, perhitungan, dan normalisasi). Kemunculan kata digunakan untuk mendeskripsikan dokumen, tetapi informasi lokasi relatif dari kata-kata dalam dokumen sebagian besar diabaikan. Di sini, *CountVectorizer* telah digunakan yang merupakan alat sederhana untuk menandai kumpulan dokumen teks serta membangun kosa kata dari kata-kata yang dikenal. *CountVectorizer* adalah *tools* Python fantastis yang digunakan untuk mengonversi teks tertentu menjadi vektor

berdasarkan frekuensi (hitungan) setiap kata dalam dokumen lengkap. Ini berguna ketika kita memiliki banyak teks seperti itu dan ingin mengonversi setiap kata di setiap teks menjadi vektor. Gambar 2.8, menunjukkan proses vektorisasi kata ke kata dalam bentuk *sequence*. Penambangan teks sangat bergantung pada *n-gram*, yang merupakan kumpulan urutan *n-item* yang terjadi secara bersamaan atau berkelanjutan dari teks besar atau urutan kalimat. Item dalam hal ini bisa berupa kata atau huruf. Kata unik dalam kalimat tersebut disebut sebagai 1 gram atau *unigram*. *Unigram* adalah model yang hanya bergantung pada frekuensi kata tanpa mempertimbangkan kata sebelumnya. *Bigram* (*2-gram*) adalah kombinasi dua kata. *Bigram* mengacu pada model yang hanya menggunakan kata sebelumnya untuk memprediksi kata saat ini. *Trigram* adalah perpaduan tiga kata. Model *trigram* jika dua kata sebelumnya diperhitungkan seperti yang diberikan.



Gambar 2.9 Contoh Proses Untuk *Uni-Gram, Bi-Gram,* Dan *Tri-Gram* [1]

Model *N-gram* memprediksi kata yang paling mungkin mengikuti urutan kata N-1 yang diberikan urutannya. Ini adalah model probabilistik yang dilatih pada korpus teks. Model seperti ini berguna dalam berbagai aplikasi NLP, termasuk pengenalan suara, dan terjemahan mesin. Model *N-gram* dibangun dengan menghitung berapa kali urutan kata tertentu muncul dalam teks korpus dan kemudian memprediksi kemungkinannya. Kemungkinan *N-gram* tertentu di dalam rangkaian kata apa pun dalam suatu bahasa diprediksi oleh model bahasa *N-gram*. Penulis artikel ini [26] menyajikan model saraf berbasis karakter untuk tugas bersama yang diperkuat dengan perhatian *multi-channel n-gram* dalam penelitian ini. Dalam modul perhatian/*attention*, karakteristik *n-gram* dibagi menjadi kelompok-kelompok berdasarkan berbagai parameter, dan *n-gram* dalam setiap

kelompok diberi bobot dan di diskriminasi menurut kepentingannya untuk tugas bersama dalam skenario yang diberikan. Posting media sosial telah diklasifikasikan berdasarkan filter *sport*, dimana Penulis telah melakukan analisis perbandingan antara dua teknik ekstraksi fitur untuk analisis sentimen yaitu karakter *n-gram* dan kata *n-gram* [27]. Sinyal eksternal yang diawasi sendiri (atau pengetahuan yang diperoleh memulai pembelajaran tanpa pengawasan, seperti *n-gram*) telah ditunjukkan dalam penelitian terbaru efektif dalam memberikan pembenaran semantik yang bermakna untuk memahami bahasa seperti bahasa Cina, serta meningkatkan kinerja pada banyak tugas *downstream*. Untuk meningkatkan pembuat enkode lebih jauh, Penulis mengusulkan pembuat enkode yang ditingkatkan *pre-trained n-gram* dengan volume data yang sangat besar dan metodologi pelatihan lanjutan [28].

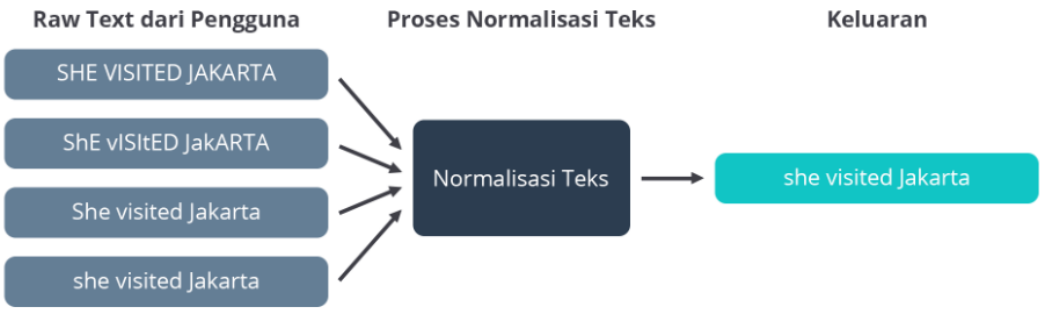
Frekuensi prediksi yang sesuai untuk *n-gram* dan (n-1)-gram dapat digunakan untuk menghitung probabilitas *n-gram*, yang merupakan wawasan penting pertama terhadap solusi tersebut. Oleh karena itu, frekuensi untuk *n-gram* dan (n-1)-gram dapat menjadi :

$$P(w_n|w_1\ w_2\ \ldots.w_{n-1}) = c(w_1\ \ldots.w_n|L)\ c(w_1\ \ldots.w_{n-1}|L) \quad (3.1)$$

Di mana $c(w|L)$ menunjukkan beberapa kali substring w muncul dalam frase L.

C. Proses Normalisasi Teks

Kata-kata dapat memiliki banyak akhiran. Angka terkadang ditulis sebagai angka dan terkadang sebagai teks. Misalnya, angka '9' terkadang ditulis sebagai 'sembilan' atau '9'. Dalam bahasa Inggris, kata-kata dapat dieja secara berbeda dan memiliki bentuk yang berbeda. Misalnya, "*am*", "*is*", dan "*are*" adalah bentuk dasar dari "*to be*". Normalisasi teks adalah proses mengubah teks menjadi bentuk yang mencakup semua variannya.



Gambar 2.10 Contoh Normalisasi Pada Teks [24]

Beberapa contoh normalisasi pada teks antara lain:

1) Mengubah semua teks menjadi huruf kecil atau huruf besar,
2) Mengonversi semua angka menjadi teks,
3) Menghilangkan *trailing*,
4) Mengubah bentuk kata menjadi bentuk dasar.

Ada beberapa cara untuk melakukan normalisasi. Dua diantaranya yaitu proses *stemming*, dan *lemmatization*. *Stemming* adalah proses mengeliminasi imbuhan untuk mendapatkan kata dasarnya.

Gambar 2.11 Proses *Stemming* [24]

Lematisasi adalah transformasi kata menjadi bentuk dasar yang bermakna, dengan mempertimbangkan konteks. Mengelompokkan kata-kata dengan bentuk akar yang sama sehingga dapat diidentifikasi sebagai objek tunggal. Bentuk dasarnya disebut wordmark, terkadang disebut bentuk standar.

Gambar 2.12 Proses *Lemmatization* [24]

2.2.4.3 *Feature Engineering* untuk Representasi Teks

Data teks memerlukan persiapan khusus sebelum pemodelan. Pertama, kita perlu menganalisis teks menggunakan teknik simbolik. Ini harus dikodekan menjadi angka yang dapat digunakan sebagai masukan ke algoritme pembelajaran mesin. Proses ini disebut pemisahan atau perencanaan fitur. Metode ini mengubah data input menjadi format numerik teks untuk digunakan dalam pemodelan. Ini adalah salah satu langkah terpenting dalam menentukan hasil model NLP. Pemetaan fitur adalah langkah umum dalam proyek pembelajaran mesin, apa pun jenis datanya: teks, gambar, audio, atau video. Namun, representasi properti data teks seringkali jauh lebih kompleks daripada format data lainnya. Untuk memahami ini, mari kita lihat bagaimana format data gambar dan video direpresentasikan secara digital.

Gambar 2.13 Ilustrasi Komputer Membaca Gambar Dalam Bentuk Matriks Piksel [29]

Gambar di atas menunjukkan seperti apa rupa seekor kucing di mata kita dan bagaimana sebuah mesin merender garis-garis gambar seekor kucing. Komputer membaca gambar sebagai matriks piksel, dengan nilai piksel yang mewakili intensitas warna. Representasi matriks ini memberikan representasi akurat dari keseluruhan gambar secara keseluruhan.

Video pada dasarnya adalah kumpulan gambar diam selama periode waktu tertentu, masing-masing gambar. Oleh karena itu, setiap video juga direpresentasikan sebagai sekumpulan matriks yang diurutkan berdasarkan bingkai.

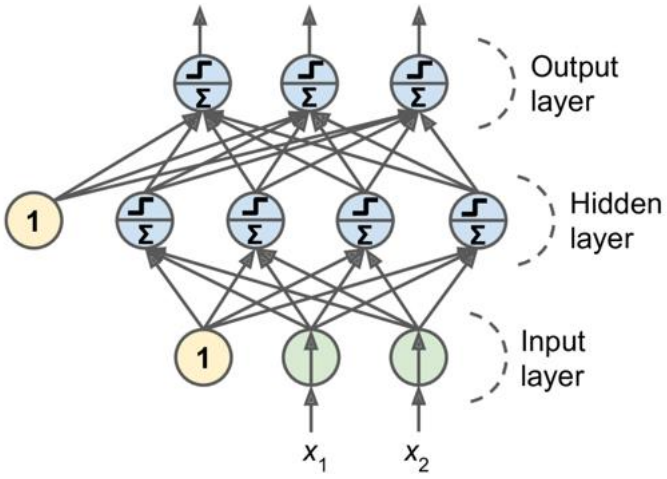
Gambar 2.14 Representasi Mesin Terhadap Kumpulan *Frame* Video [29]

Ada beberapa cara menggambar fungsi untuk merepresentasikan teks. Metode-metode ini terbagi dalam empat kategori [20], antara lain:

1. *Basic vectorization approaches* (contoh: *One-Hot Encoding, Bag of Words, Bag of N-Grams,* dan *TF-IDF*).
2. *Distributed representations* (contoh: *Words Embeddings*).
3. *Universal text representation* (menggunakan *pre-trained embedding*).
4. *Handcrafted features* (mengandalkan *domain-specific knowledge*).

2.3 Arsitektur *Neural Network*

Ini adalah kasus untuk pembelajaran mesin (*machine learning*) dan jaringan saraf tiruan. Jaringan saraf tiruan atau *artificial neural network* adalah model pembelajaran mesin yang terinspirasi oleh neuron di otak manusia. Ini merupakan model ML yang serbaguna, kuat, dan sangat skalabel. Dengan keunggulan tersebut, jaringan saraf tiruan sangat mampu mengklasifikasikan miliar gambar, mengenali ratusan bahasa dunia, merekomendasikan video ke ratusan juta pengguna, dan sampai belajar mengalahkan juara dunia permainan papan GO. Sangat cocok digunakan untuk masalah ML yang kompleks.



Gambar 2.15 Arsitektur *Neural Network*

Jaringan Saraf Tiruan [30] merupakan teknik dalam *machine learning* yang terdiri atas lapis masukan (*input layer*) dan lapisan keluar (*output layer*). Setiap lapis terdiri atas satu atau beberapa unit neuron yang mempunyai sebuah fungsi aktivasi yang menentukan keluaran dari unit tersebut. Kita bisa menambahkan lapis tersembunyi (*hidden layer*) untuk menambah kemampuan dari NN tersebut. NN dapat dilatih dengan menggunakan *data training*. Semakin banyak data training maka akan semakin bagus untuk kerja dari NN tersebut. Namun, kemampuan NN juga terbatas pada jumlah lapisan, semakin banyak jumlah lapisan semakin tinggi kapasitas NN tersebut. Semakin banyak lapisan juga membawa kekurangan yaitu banyaknya jumlah iterasi atau *training* yang dibutuhkan.

2.4 *Word Embeddings*

Komponen utama dari pendekatan jaringan saraf adalah penggunaan penyematan yang mewakili setiap fitur sebagai vektor dalam ruang berdimensi rendah.

2.4.1 *Unsupervised pre-training*

Ini adalah kasus umum bahwa kami tidak memiliki tugas tambahan dengan data beranotasi yang cukup (atau kami ingin mulai melatih tugas tambahan dengan vektor yang lebih baik). Dalam kasus seperti itu, kami menggunakan "metode *unsupervised*" yang memungkinkan kami melatih teks tanpa komentar dalam jumlah besar, dan mengambil banyak contoh *supervised learning* dari teks mentah

sehingga tugas yang kami buat menjadi tugas akhir yang penting bagi kami. menjadi sama (atau cukup dekat). Kesamaan kata sulit untuk didefinisikan dan biasanya sangat bergantung pada keterkaitan, tetapi pendekatan saat ini didasarkan pada hipotesis distribusi [31], yang measlikan bahwa kata-kata itu serupa jika berada dalam konteks yang serupa. muncul. Berbagai metode semuanya menghasilkan contoh pelatihan yang diawasi yang tujuannya adalah untuk memprediksi kata dari konteksnya, atau konteks dari sebuah kata.

Keuntungan penting dari menyematkan kata-kata pelatihan dalam sejumlah besar data yang tidak dianotasi adalah menyediakan representasi vektor dari kata-kata yang tidak muncul dalam set pelatihan yang diawasi. Idealnya, representasi dari kata-kata ini harus serupa dengan kata-kata terkait yang muncul di set pelatihan untuk memungkinkan model menggeneralisasi kejadian yang tidak terlihat dengan lebih baik. Oleh karena itu, kesamaan antara vektor kata yang diperiksa oleh algoritme tanpa pengawasan diharapkan dapat menangkap aspek kesamaan yang sama yang membantu melakukan tugas jaringan yang dimaksud.

Algoritma penyematan kata *unsupervised* yang umum termasuk *word2vec* [32], *GloVe* [33], dan algoritma penyisipan *Collobert dan Weston* [34]. Model ini terinspirasi oleh jaringan saraf dan didasarkan pada pelatihan gradien stokastik. Namun, mereka terkait erat dengan keluarga algoritme lain yang tumbuh di komunitas NLP dan IR, yang didasarkan pada faktorisasi matriks.

Pilihan masalah tambahan (apa yang diprediksi berdasarkan konteks apa) dapat mempengaruhi vektor yang dihasilkan lebih dari metode pembelajaran yang digunakan untuk pelatihan. Oleh karena itu, kami hanya akan membaca sepintas detail metode pelatihan, dengan fokus pada berbagai masalah tambahan yang tersedia. Beberapa paket perangkat lunak tersedia untuk menurunkan vektor kata. Misalnya, *word2vec* dan *Gensim*, yang mengimplementasikan model *word2vec* dengan konteks berbasis jendela kata, *word2vec*, versi modifikasi dari *word2vec* yang memungkinkan penggunaan konteks arbitrer, dan model *Grove*. Banyak vektor kata terlatih juga tersedia untuk diunduh di Internet.

Ini berada di luar cakupan tutorial ini, tetapi perlu dicatat bahwa penyematan kata yang berasal dari algoritme pelatihan *unsupervised* memiliki

aplikasi yang luas di NLP selain digunakan untuk menyematkan lapisan penyisipan kata dari inisialisasi model jaringan saraf.

2.5 *Word Embeddings: GloVe*

GloVe, seperti namanya, menyarankan vektor global adalah teknik pembelajaran tanpa pengawasan/*unsupervised learning* yang menghasilkan representasi vektor kata. Representasi yang dihasilkan mengungkapkan segmen linier yang menarik dari ruang vektor kata, dengan pelatihan berdasarkan korpus statistik kejadian bersama kata-ke-kata global. *GloVe* memiliki keuntungan bahwa, tidak seperti *Word2vec*, tidak hanya bergantung pada statistik lokal (informasi konteks lokal kata) untuk menghasilkan vektor kata, tetapi juga menggabungkan statistik di seluruh dunia (*word co-occurrence*).

Teknik yang ada untuk mempelajari representasi ruang vektor dari kata-kata telah efektif dalam menarik keteraturan semantik dan sintaksis berbutir halus menggunakan aritmatika vektor, tetapi asal usul keteraturan ini tetap menjadi misteri. Akibatnya, model regresi *log-linear global* baru telah dikembangkan yang menggabungkan manfaat dari dua keluarga model utama: faktorisasi matriks global dan pendekatan jendela konteks lokal [33].

Karena rasio P_{ik}/ P*jk* ditentukan oleh tiga kata, i, j, dan k, model yang paling umum adalah sebagai berikut:

$$F(w_i , w_j, w_k) = P_{ik} / P_{jk} \quad (2.1)$$

Di mana wER^d, *i*, dan *j* adalah kata-kata dalam korpus. Rasio probabilitas kemunculan bersama mereka dengan berbagai kata penyelidikan, *k*, dapat digunakan untuk menyelidiki hubungan antara istilah-istilah ini. Sedangkan P_{ik}/P*jk* adalah perbandingannya.

Semantara F mungkin merupakan fungsi canggih yang ditentukan oleh jaringan saraf ini akan menyembunyikan struktur linier yang sedang dicoba untuk digambarkan. Produk titik antara dua entitas dapat dipertimbangkan:

$$F((w_i - w_j)^T w_k) = P_{ik} / P_{jk} \quad (2.2)$$

Perbedaan antara kata dan kata konteks dalam matriks *co-occurrence* kata-kata bersifat subyektif, dan dengan demikian pertukuran diperbolehkan antara dua

peran. F dapat memiliki sifat homomorfisme yang memastikan bahwa pembagian F(A)/F(B) dapat digunakan untuk measlikan pengurangan F(A-B):

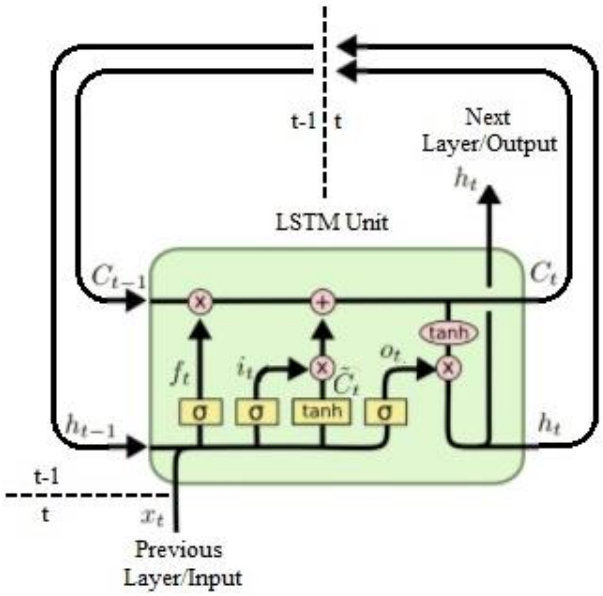
$$F(w^T{}_i w_k) = P_{ik} = X_{ik} / X_i \quad (2.3)$$

Setelah proses tokenisasi, parameter lain di *scikit-learn* dapat digunakan untuk menyempurnakan input model. Untuk menyempurnakan input model, konsep fitur maks telah digunakan di sini. Daripada menggunakan semua kata, kompleksitas dan ukuran mmodel dapat dikurangi dengan menggunakan jumlah maksimum kata. Model *GloVe* dilatih pada entri bukan nol dari matriks *co-occurrence* kata-kata global, yang melacak seberapa sering kata muncul bersama dalam korpus. Mengumpulkan statistik untuk matriks ini memerlukan satu putaran di seluruh korpus. Jalur ini bisa mahal secara komputasi untuk perusahaan besar, tetapi ini adalah investasi satu kali. Karena jumlah entri matriks bukan nol seringkali jauh lebih sedikit daripada jumlah total kata dalam korpus, iterasi pelatihan selanjutnya jauh lebih cepat.

2.5 Jaringan *Long Short Term Memory*

RNN sulit untuk dilatih secara efektif karena masalah *vanishing gradients* [35]. Sinyal kesalahan (gradien) pada langkah selanjutnya dalam urutan berkurang dengan cepat proses propagasi balik dan tidak mencapai sinyal input sebelumnya, sehingga menyulitkan RNN untuk menangkap dependensi jarak jauh. Arsitektur *Long Short Term Memory* (LSTM) [36] dirancang untuk memecahkan masalah gradien yang hilang.

Model *Long Short Term Memory* (LSTM) adalah arsitektur RNN yang dikembangkan untuk menghindari ketergantungan jangka panjang pada RNN [37]. Model LSTM memiliki keunggulan menyelesaikan masalah data yang bersifat berurutan [38]. Komponen utama lapisan LSTM disebut sel memori. Sel memori terdiri dari empat elemen utama, yaitu *input gate*, *neuron* dengan koneksi berulang, *forget gate*, dan *output gate*. Masukkan yang disediakan untuk LSTM mengontrol operasi yang akan dilakukan oleh *gate* di sel memori [39], seperti *write* (*input gate*), *read* (*output gate*), *reset* (*forget gate*).



Gambar 2.16 Arsitektur LSTM

Arsitektur LSTM memungkinkan untuk menambah atau menghapus informasi dari *cell state*. Hal ini disebut sebagai *gates* yang merupakan pengatur apakah informasi akan diteruskan atau diberhentikan. *Gates* terdiri dari lapisan *sigmoid* dan *pointwise multiplication operation*, yang mana *output* dari lapisan *sigmoid* adalah angka 1 dan 0 yang menunjukkan apakah informasi tersebut akan diteruskan atau diberhentikan. Lebih lanjut bahwa jika bernilai 0 menunjukkan tidak ada informasi yang akan diteruskan, sebaliknya jika bernilai 1 maka informasi akan diteruskan. Menurut [40], dalam arsitektur LSTM terdapat tiga *gate* utama yang mengatur alur informasi, yaitu sebagai berikut:

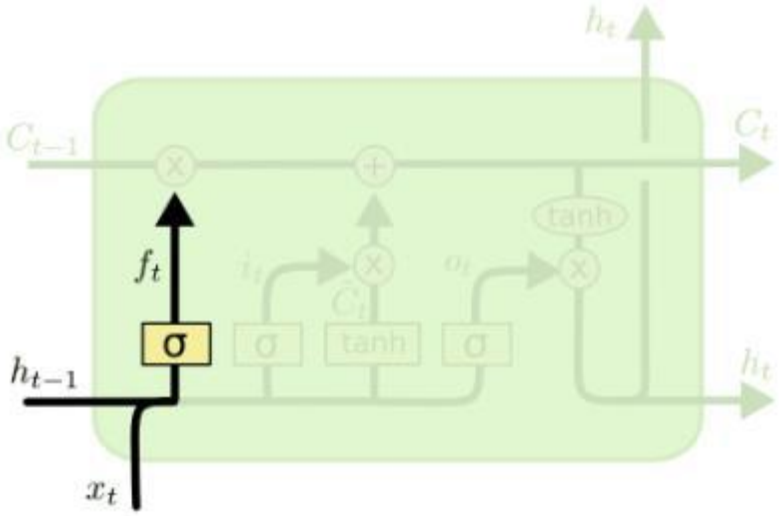
1. *Forget gate*, merupakan *gate* yang memutuskan informasi mana yang akan dihapus dari *cell.*
2. *Input gate*, merupakan *gate* yang memutuskan nilai dari *input* untuk di perbarui pada *state* memori.
3. *Output gate,* merupakan *gate* yang memutuskan apa yang akan dihasilkan sesuai dengan *input* dan memori pada *cell.*

Pada umumnya dalam *Neural Network* terdapat bobot yang digunakan sebagai paramter dalam pemrosesannya, begitu juga jenis arsitektur *Recurrent Neural Network*. Namun dalam arsitektur *Recurrent Neural Network* memiliki bobot berulang atau *recurrent weight* (U) di setiap jaringannya, hal tersebut

dikarenakan dalam pemrosesannya dilakukan secara berulang. Menurut [41], berikut ini adalah langkah-langkah perulangan pada arsitektur LSTM:

1. Memutuskan informasi yang akan dibuang

   Langkah pertama adalah memutuskan informasi apa yang akan dibuang dari *state cell*. Keputusan ini dibuat oleh lapisan *sigmoid*, yaitu *forget gate layer*. Pada lapisan ini (*forget gate*), akan memproses h_{t-1} dan x_t sebagai *input*, dan menghasilkan output berupa angka 0 dan 1 pada *cell state* C_{t-1}. Angka mendekati 1 menunjukkan informasi akan disimpan, sedangkan angka 0 menunjukkan informasi akan dibuang.



Gambar 2.17 Langkah Pada *Forget Gate Layer*

Persamaan *forget gate* dinotasikan pada persamaan 2.4, sebagai berikut:

$$ft = \sigma(W_f.x_t+U_f.h_{t-1}+b_f)\text{.....(2.4)}$$

Keterangan:

f_t = *forget gate*

σ = fungsi *sigmoid*

W_f = nilai bobot untuk *forget gate*

U_f = nilai bobot berulang untuk *forget gate*

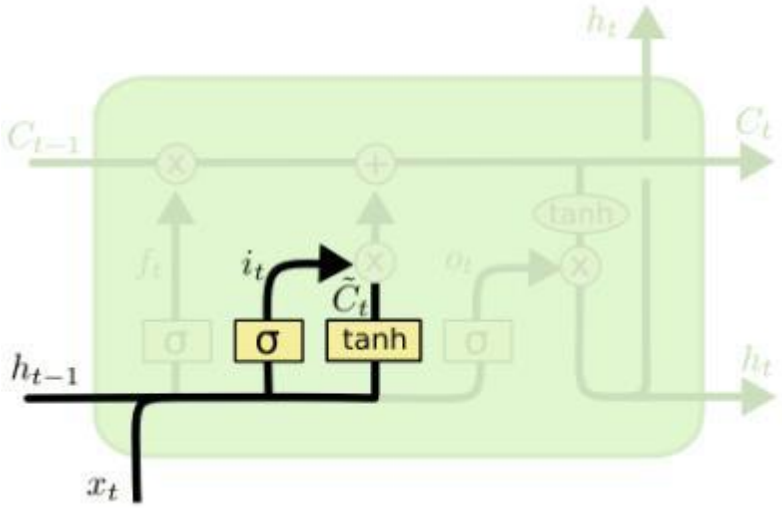
h_{t-1} = nilai *output* sebelum orde ke t

x_t = nilai *input* pada orde ke t

b_f = nilai bias pada *forget gate*

2. Simpan informasi ke dalam *cell state*

   Langkah kedua adalah memutuskan informasi apa yang harus disimpan di *cell state* Ct. Terdapat dua bagian dalam tahapan ini, bagian pertama lapisan *sigmoid*, yaitu *input gate layer* memutuskan nilai mana yang akan diperbarui, bagian kedua yaitu *tanh layer* membuat kandidat dengan nilai baru, $\tilde{C}t$, yang dapat ditambahkan ke *cell state*. Selanjutnya menggabungkan *ouput* dari *input gate layer* dan *tanh layer* untuk memperbarui *cell state*.



Gambar 2.18 Langkah Pada *Input Gate Layer* Dan *Tanh Layer*

Persamaan kandidat baru dinotasikan pada persamaan 2.5 berikut:

$$i_t = \sigma(W_i.x_t+U_i.h_{t-1}+b_i) \quad \text{.....(2.5)}$$

Keterangan:

i_t = *input gate*

σ = fungsi *sigmoid*

W_i = nilai bobot untuk *input gate*

U_i = nilai bobot berulang untuk *input gate*

h_{t-1} = nilai *output* sebelum orde ke t

x_t = nilai *input* pada order ke t

b_i = nilai bias pada *input gate*

Persamaan kandidat baru dinotasikan pada persamaan 2.6 berikut:

$$\tilde{C}t = tanh(Wc.xt+Uc.ht-1+bc) \quad \text{.....(2.6)}$$

Keterangan:

$\tilde{C}_t$ = nilai baru yang dapat ditambahkan ke *cell state*

$tanh$ = fungsi *tanh*

W_c = nilai bobot untuk *cell state*

U_c = nilai bobot berulang untuk *cell state*

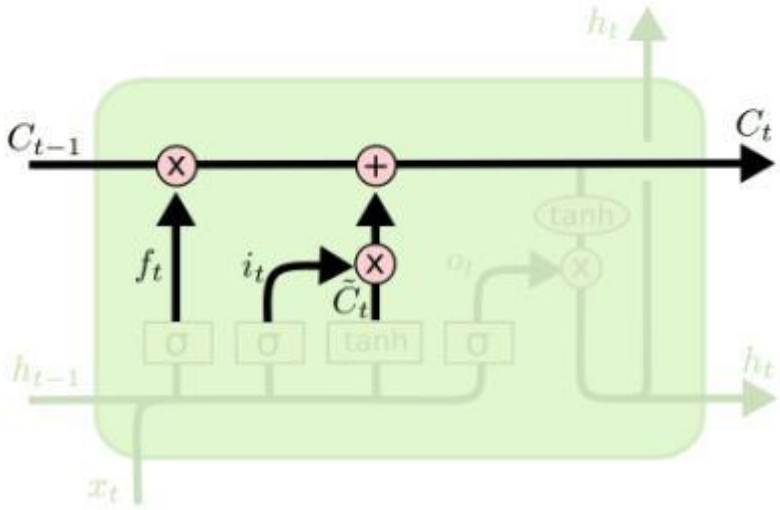
h_{t-1} = nilai *output* sebelum order ke t

x_t = nilai *input* pada orde ke t

b_c = nilai bias untuk *cell state*

3. Memperbarui *cell state*

Langkah ketiga adalah memperbarui *cell state* yang lama, C_{t-1}, menjadi *cell state* baru, C_t, dengan mengkalikan *state* lama dengan f_t. Kemudian ditambahkan dengan nilai kandidat baru, $i_t * C_t$, yang digunakan untuk memperbarui *state*.



Gambar 2.19 Langkah Memperbarui *Cell State*

Persamaan *cell state* dinotasikan pada persamaan 2.7, sebagai berikut:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{.....(2.7)}$$

Keterangan:

C_t = *cell state*

f_t = *forget state*

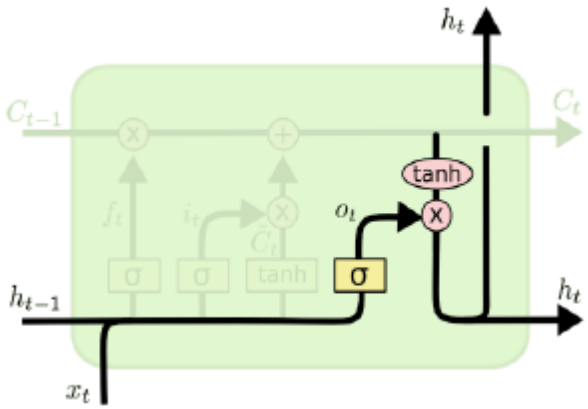
C_{t-1} = *cell state* sebelum orde ke t

i_t = *input gate*

$\tilde{C}_t$ = nilai baru yang dapat ditambahkan ke *cell state*

4. Konfirmasi hasil *output* h_t

   Langkah terakhir adalah mengkonfirmasi hasil *output* pada *output gate*. Hasil *output* harus sesuai dengan *cell state* yang telah diproses. Pertama lapisan *sigmoid* menentukan bagian dari *cell state* yang menjadi hasil *output*. Kemudian, *output* dari *cell state* dimasukkan ke dalam lapisan *tanh* (untuk mengganti nilai menjadi antara -1 dan 1) dan dikalikan dengan *sigmoid gate*, agar *output* yang dihasilkan sesuai dengan apa yang ditentukan sebelumnya.



Gambar 2.20 Langkah Menentukan *Output*

Persamaan *output gate* dinotasikan pada persamaan 2.8, sebagai berikut:

$$o_t = \sigma(W_o.x_t+U_o.h_{t-1}+b_o) \quad \text{.....(2.8)}$$

Keterangan:

o_t = *output gate*

σ = fungsi *sigmoid*

W_o = nilai bobot untuk *output gate*

U_o = nilai bobot berulang untuk *output gate*

h_{t-1} = nilai *output* sebelum orde ke t

x_t = nilai *input* pada orde ke t

b_o = nilai bias pada *output gate*

Sedangkan persamaan nilai *output* orde t diuraikan pada persamaan 2.9, sebagai berikut:

$$h_t = o_t * \tanh(c_t) \quad \text{.....(2.9)}$$

Keterangan:

h_t = nilai *output* orde t

o_t = *output gate*

$tanh$ = fungsi *tanh*

C_t = *cell state*

2.6 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi yang digunakan dalam jaringan saraf untuk menghitung jumlah *input* dan *bias* yang terbobot [42]. Hal ini memanipulasi data yang disajikan melalui beberapa pemrosesan gradien, biasanya *Gradient Descent* dan kemudian menghasilkan *output* untuk jaringan saraf, yang berisi parameter dalam data. Fungsi aktivasi dapat berupa linier atau non-linier tergantung pada fungsi yang merepresentasikannya, dan digunakan untuk mengontrol *output* dari jaringan saraf luar.

2.6.1 *Sigmoid Function*

Sigmoid merupakan fungsi aktivasi non linier yang digunakan dalam jaringan saraf. Fungsi *Sigmoid* mengubah *range* nilai *input x* menjadi nilai antara 0 dan 1 [43]. Fungsi *Sigmoid* ditunjukkan oleh persamaan 2.10. $S(x)$ akan menghasilkan sebuah kurva dalam rentang 0-1 pada sumbu *y*. Jika *x* adalah bilangan positif sangat besar, maka nilai $e-x$ memiliki nilai mendekati 0 yang menghasilkan *output* mendekati 1. Sedangkan jika *x* adalah bilangan negatif sangat besar, maka nilai $e-x$ memiliki nilai yang besar dan menghasilkan *output* mendekati 0.

$$f(x) = 1/(1+e^{-x}) \quad \text{.....(2.10)}$$

Keterangan :

$f_{sigmoid}(x)$ = fungsi *sigmoid*

e = epsilon

2.6.2 *Hyperbolic Tangent Function* (Tanh)

Fungsi aktivasi *tanh* merupakan salah satu fungsi aktivasi yang digunakan untuk kasus multi klasifikasi. Kelemahan fungsi aktivasi *tanh* adalah tidak menyelesaikan *vanishing gradient* yang umum terjadi pada *sigmoid* [42]. Fungsi aktivasi *tanh* ditunjukkan oleh persamaan 2.11. Jika x merupakan bilangan negatif sangat besar, maka nilai *tanh* akan mendekati -1, ketika nilai x merupakan bilangan positif sangat besar, maka nilai *tanh* mendekati 1.

$$f(x) = (e^{x}-e^{-x}/e^{x}+e^{-x}) \quad (2.11)$$

Keterangan:

$f_{tanh}(x)$ = fungsi *tanh* dari x

e = epsilon

2.6.3 *Rectified Linear Unit* (ReLu)

Fungsi ReLU merupakan salah fungsi aktivasi yang telah terbukti menjadi fungsi yang sering digunakan dalam *Deep Learning*. Hal ini dikarenakan fungsi ReLU menawarkan kinerja dan generalisasi yang lebih baik dalam *Deep Learning* dibandingkan dengan fungsi aktivasi *sigmoid* dan *tanh* [42]. Aktivasi ReLU akan memberikan keluaran 0 ketika $x<0$ dan merepresentasikan fungsi linier ketika $x \geq 0$. Fungsi aktivasi ReLU melakukan operasi ambang batas untuk setiap elemen *input* di mana nilai kurang dari nol diatur ke nol sehingga ReLU dinotasikan sebagai persamaan berikut:

$$f(x) = \max(0,x) = \{xi, if\ xi \geq 0;\ 0, if\ xi < 0 \quad (2.12)$$

Range dari fungsi aktivasi ReLU adalah [0, ∞], semua *input* bernilai negatif pada fungsi aktivasi ReLU akan menjadi nilai 0. Fungsi ini memperbaiki nilai *input* yang kurang dari nol sehingga diubah menjadi nol dan menghilangkan masalah gradien (*vanishing gradient*) yang diamati pada jenis aktivasi sebelumnya.

2.6.4 *Softmax Function*

Fungi aktivasi *Softmax* adalah jenis dari fungsi aktivasi yang digunakan dalam komputasi saraf. Ini digunakan untuk menghitung distribusi probabilitas dan vektor bilangan *real*. Fungsi *Softmax* biasanya digunakan sebagai *output* dari model klasifikasi untuk merepresentasikan distribusi kemungkinan terhadap sejumlah

kelas [42]. Fungsi *Softmax* menghasilkan *output* yang kisaran nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1. Fungsi *softmax* dihitung menggunakan persamaan berikut:

$f(xi) = exp\ (xi)\ /\ \Sigma j\ exp\ (xj)$ …….……...……(2.13)

Keterangan :

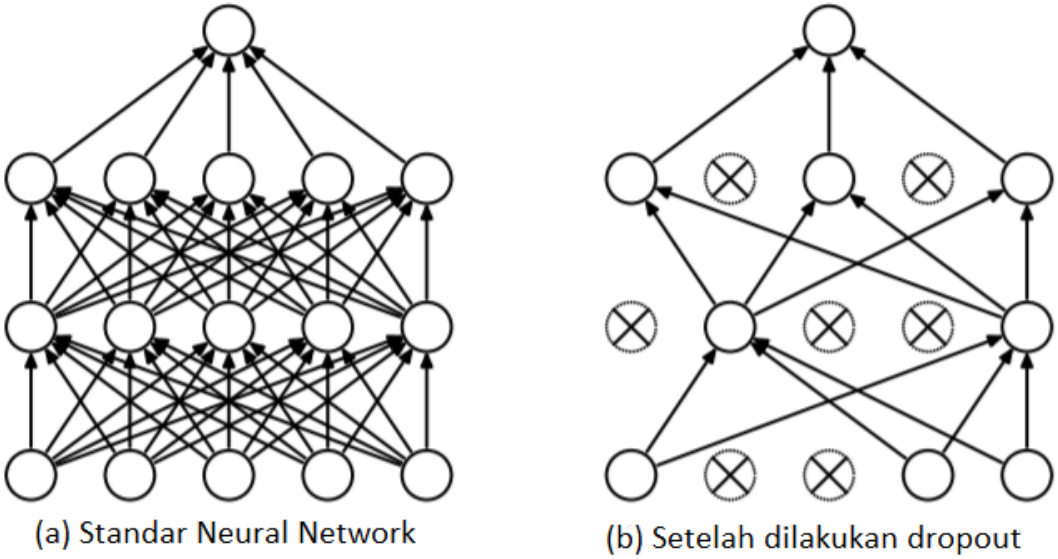
x = nilai *input* lapisan sebelumnya

i,j = indeks unit dan lapisan

Fungsi *softmax* digunakan dalam model multi kelas di mana fungsi ini mengembalikan probabilitas setiap kelas. Fungsi *softmax* sebagian besar muncul di hampir semua lapisan *output* dari arsitektur *Deep Learning*. Perbedaan utama antara fungsi aktivasi *Sigmoid* dan *Softmax* adalah *Sigmoid* digunakan dalam klasifikasi biner sedangkan *Softmax* digunakan untuk tugas klasifikasi multivarian [42].

2.7 *Dropout*

Dropout merupakan teknik regularisasi model *Neural Network* untuk mengurangi *overfitting* pada *dataset*. *Dropout* adalah teknik di mana *neuron* yang dipilih secara acak diabaikan selama pelatihan (*dropout* secara acak). Ini berarti bahwa kontribusinya terhadap aktivasi *neuron* untuk sementara dihapus pada *forward pass* dan setiap pembaruan bobot tidak diterapkan pada *neuron* pada *backward pass*. Istilah *dropout* merujuk pada menggugurkan unit (yang tersembunyi ataupun yang terlihat) dalam sebuah jaringan syaraf tiruan. Menggugurkan unit merupakan upaya menghilangkan sementara unit tersebut dari jaringan syaraf tiruan dengan semua koneksi yang masuk dan keluar menuju unit tersebut [44]. Seperti yang ditunjukkan pada gambar 2.21.



Gambar 2.21 (a) Standar *Neural Network* Dengan 2 *Hidden Layer*, (b) Contoh Setelah Dilakukan *Dropout* Pada *Network* [44]

Dalam kasus yang sederhana, setiap unit dipertahankan dengan probabilitas tetap p, tidak tergantung pada unit lain, di mana p dapat dipilih menggunakan set validasi atau hanya dapat ditetapkan pada 0,5, yang mendekati optimal untuk berbagai jaringan. Namun, untuk unit *input*, probabilitas retensi optimal biasanya lebih dekat ke 1 dari pada ke 0,5 [44].

2.8 Optimasi Adam

Optimasi Adam merupakan metode optimisasi berbasis stokastik orde pertama dari fungsi objektif stokastik untuk memperbarui bobot jaringan yang berulang berdasarkan data pelatihan [45]. Optimasi Adam bertujuan untuk menghitung *adaptive learning rate* untuk setiap parameter. *Learning rate* menentukan seberapa banyak perubahan bobot pada jaringan. Berikut ini merupakan persamaan untuk optimasi Adam [46]:

1. Menghitung gradien gt pada waktu t

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \quad \text{(2.14)}$$

2. Memperbarui bias momen vektor pertama

$$m_t = \beta_1 m_{t-1} + (1-\beta 1) g_t \quad \text{(2.15)}$$

3. Memperbarui bias momen vektor kedua

$$v_t = \beta_2 v_{t-1} + (1-\beta 2) g_t^2 \text{…..……(2.16)}$$

4. Menghitung bias momen vektor pertama

$$\hat{m_t} = m_t \,/\, 1-\beta_1{}^{t} \text{…..…….……..(2.17)}$$

5. Menghitung momen vektor kedua

$$\hat{v_t} = v \,/\, 1-\beta_2{}^{t} \text{….…….(2.18)}$$

6. Memperbarui parameter dengan menggunakan *learning rate*

$$\theta_{t+1} = \theta_t - n \,/\, \sqrt{\hat{v_t}} + \varepsilon\, \hat{m_t} \text{…….....(2.19)}$$

2.9 *Cross Entropy Loss Function*

Loss function merupakan metode untuk mengevaluasi seberapa baik algoritma memodelkan suatu data [47]. *Loss function* memiliki kurva yang bertujuan untuk memberi tahu cara mengubah parameter untuk membuat model lebih akurat.

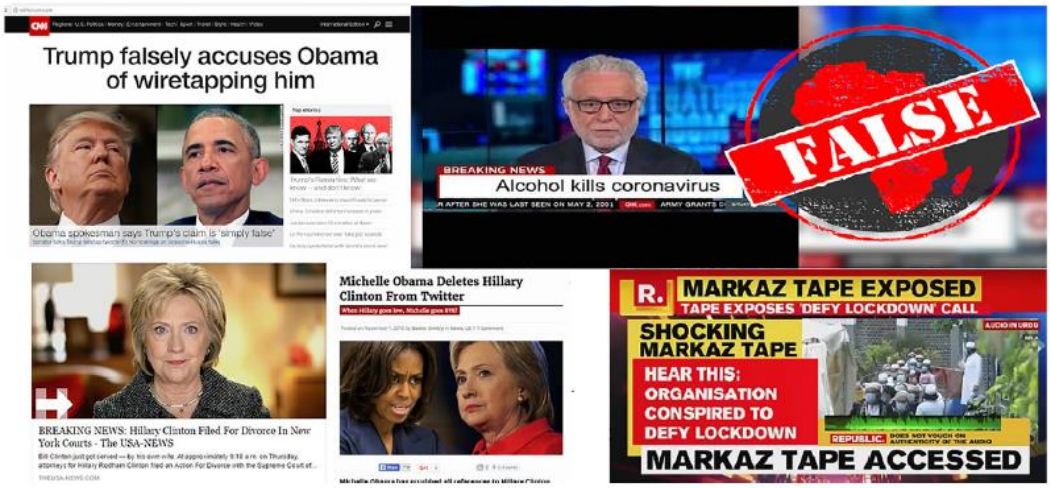
Cara kerja *loss function* adalah membandingkan hasil prediksi dari *output layer* dengan target. *Loss function* yang populer saat ini adalah *cross entropy*. Metode *cross entropy* mengukur kinerja model klasifikasi yang *output* nya merupakan nilai probabilitas antara 0 dan 1. *Cross entropy* dinotasikan dengan persamaan 2.20.

$$H(p,q) = -\Sigma_x p(x) \log (q(x)) \text{……(2.20)}$$

2.10 Berita Palsu

Menjelang pemilihan umum Nigeria 2019, bahaya sosial, ekonomi, dan politik bagi masyarakat Nigeria hadir. Penyebaran informasi palsu selama waktu pemilihan adalah penyebab ancaman ini bagi penduduk Nigeria. Penelitian ini menunjukkan bahwa ketika tindakan serius diambil, berita palsu dan gagasan *post-truth* terkait dapat terus menjadi ancaman bagi pemerintahan Nigeria. Strategi keterlibatan tiga lapis yang mencakup pemerintah, legislator, dan masyarakat umum harus dikembangkan dan diikuti secara ketat untuk memerangi konsekuensi berita palsu dan fenomena *post-truth* secara efektif [48]. Media yang berhubungan dengan politik secara signifikan dipengaruhi oleh *post-truth* berbeda satu sama lain [49]. Sebagai alat untuk berbagi ide dan komentar, *Twitter* semakin banyak

digunakan di sektor politik. Salah satu kandidat pemilihan Presiden Amerika Serikat 2016, Donald Trump, menggunakan *Twitter* untuk berinteraksi dengan publik dan terus menggunakannya bahkan setelah menang. Penggunaan *Twitter* oleh Trump menuai kritik atas sambutannya tentang berbagai topik, termasuk Hillary Clinton, ukuran kerumunan pada pelantikannya, kebijakan pemerintahan Obama sebelumnya, imigrasi, dan urusan global. Seringnya Trump mengejek media di Twitter dengan menggunakan terminologi seperti "berita palsu" dan "media yang tidak jujur" telah menjadi salah satu fitur platform yang paling menonjol. Ungkapan-ungkapan ini berkontribusi pada kepercayaan publik yang semakin besar terhadap berita media. Mengingat kebenaran obyektif, orang mungkin berpendapat bahwa Trump, bersama dengan media, harus disalahkan karena menyebarkan informasi palsu [50].



Gambar 2.22 Contoh Berita Palsu Yang Tersebar Di Media Sosial Facebook [8]

Dan diantara sekian banyak situs media sosial saat ini, *WhatsApp* telah menjerat masyarakat dalam penyebaran informasi palsu. Untuk mengilustrasikan efek dari virus corona, sejumlah klip yang dimaksudkan untuk menggambarkan pandangan dari China seperti gambar korban yang tergeletak di jalanan beredar di *WhatsApp*. Organisasi pemeriksa fakta telah mengungkapkan beberapa dari film ini adalah rekaman lama, latihan palsu, atau bahkan adegan film. [51].