BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Analisis sentimen merupakan tugas dasar dalam bidang *Natural Language Processing*. Teknik yang dapat digunakan untuk melakukan analisis sentimen yaitu, menggunakan *machine learning based*, dan *lexicon based*. *Machine learning* klasik sekarang sudah berkembang menjadi *deep learning*. Perkembangan *machine learning* klasik menjadi *deep learning* menjadikan teknik analisis sentimen juga mulai berkembang menggunakan metode *deep learning* (Ain dkk, 2017). Beberapa penelitian analisis sentimen baik yang menggunakan *machine learning* klasik maupun *deep learning* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian analisis sentimen terdahulu

No.	Peneliti	Rekomendasi Metode	Keterangan
1.	(Akbar, Sediyono dkk., 2015)	*Naïve Bayes* berdasarkan ontologi	Tiga lapisan klasifikasi
2.	(Usop, Isnanto, and Kusumaningrum 2017)	*Latent Dirichlet Allocation* (LDA) dengan seleski ciri *Part of Speech* (POS).	Nilai rata-rata akurasi 64% dan seleksi ciri POS memberikan nilai akurasi lebih tinggi sebesar 7,8%
3.	(Tripathy dkk., 2016)	*Support Vector Machine* (SVM)	Akurasi SVM lebih baik dibandingkan *machine learning* klasik lainnya.
4.	(Ghulam et al. 2019)	*Long Short Term Memory* (LSTM)	Akurasi LSTM lebih baik dibandingkan SVM
5.	(Li and Qian 2016)	*Long Short Term Memory* (LSTM)	LSTM memeliki kinerja yang lebih baik dibandingkan *Recurrent Neural network* (RNN)
6.	(Chen and Wang 2019)	*Long Short Term Memory - Convolutional Neural network* (LSTM-CNN)	LSTM-CNN memiliki performa yang lebih baik dibanding LSTM, CNN, dan CNN-LSTM
7.	(Zulfa and Winarko 2017)	*Deep Belief Network* (DBN)	Teks berbahasa Indonesia menggunakan DBN, menunjukkan hasil yang lebih baik dibandingkan *Naïve Bayes*, dan SVM.

Analisis sentimen berdasarkan level kalimat dilakukan untuk mengukur presepsi sebuah produk. Proses klasifikasi yang dilakukan melewati 3 lapisan klasifikasi meliputi klasifikasi buzz/promo untuk memisahkan *tweets* promo dan bukan promo, klasifikasi subyektifitas untuk memisahkan *tweets* bukan promo menjadi *tweets* subyektif dan obyektif, dan klasifikasi sentimen untuk mendapatkan polaritas *tweets* dalam bentuk sentimen positif, negatif, atau netral (Akbar dkk., 2015). Penelitian ini masih menggunakan *machine learning* klasik dan memiliki kekurangan tidak fokus pada satu klasifikasi.

Part of Speech (POS) digunakan sebagai seleksi ciri untuk meningkatkan performa analisis sentimen dari *Latent Dirichlet Allocation* (LDA). Penelitian ini menggunakan data teks dari twitter sebanyak 500 *tweet* dengan 250 berlabel positif dan 250 berlabel negatif. Hasil penelitian ini menunjukkan bawah POS mampu meningkatkan akurasi hingga 7,8% dibandingan metode LDA tanpa menggunakan POS. Akan tetapi pada penelitian ini LDA memiliki rata-rata akurasi 64% yang relatif lebih kecil dibandingkan metode lainnya (Usop dkk., 2017).

Machine learning klasik merupakan salah satu metode yang dapat digunakan untuk melakukan analisis sentimen terhadap suatu teks. Metode *machine learinng* klasik seperti *Naive Bayes* (NB), *Maximum Entropy* (ME), *Support Vector Machine* (SVM) digunakan dalam penelitian analisis sentimen tentang ulasan film pada situs *Internet Movie Review Database* (IMDB). Ketiga metode tersebut diimplementasikan dengan pendekatan *n-gram.* Proses klasifikasi dilakukan untuk menemukan sentimen positif dan negatif dari korpus. Hasil penelitian menunjukkan bahwa metode SVM memiliki nilai akurasi yang lebih baik dibandingkan metode lainnya (Tripathy dkk, 2016).

Perbandingan metode *machine learning* klasik dan *deep learning* dilakukan untuk analisis sentimen pada ulasan *e-markerting* berupa teks Roman Urdu. *Long Short Term Memory* (LSTM) merupakan metode dari *deep learning* yang dibandingkan dengan tiga metode *machine learning* klasik yaitu, *Naïve Bayes* (NB), *Random Forest* (RF), dan *Support Vector Machine* (SVM). Hasil penelitian ini menunjukkan bahwa LSTM memiliki akurasi paling baik sebesar 95% dibandingkan SVM sebesar 92%, RF sebesar 88%, dan NB sebesar 77%. LSTM

juga dikatakan sebagai metode yang efisien untuk analisis sentimen karena tidak membutuhkan seleksi fitur (Ghulam dkk., 2019).

Long Short Term Memory (LSTM) merupakan salah satu pengembangan *Recurrent Neural network* (RNN) untuk mengatasi masalah difusi gradien. Perbandingan metode RNN dan LSTM untuk analisis sentimen teks yang panjang membuktikan bahwa LSTM lebih unggul. Data analisis yang digunakan berupa komentar JD.COM (salah satu *online shop* di Cina), Ctrip Travel dari Cina, dan ulasan film. Jumlah data JD.COM sebanyak 4.000 data latih dan 1.800 data uji, sedangkan jumlah data Ctrip Travel dari Cina sebanyak 6.000 data latih dan 2.000 data uji, serta jumlah data ulasan film sebanyak 12.500 data latih dan 12.500 data uji. Hasil dari penelitian ini menunjukkan bahwa LSTM memiliki kinerja yang lebih baik dari RNN konvensional dalam melakukan klasifikasi teks pada semua sumber data pada penelitian ini (Li dan Qian, 2016).

Long Short Term Memory dengan *Convolutional Neural network* (LSTM-CNN) digunakan untuk analisis sentimen data *twitter* berbahasa Inggris. Data yang digunakan sebanyak 100.000 *tweet* dengan pelabelan positif dan negatif. Hasil penelitian menunjukan akurasi CNN sebesar 75,18%, CNN-LSTM sebesar 76,36%, dan LSTM-CNN sebesar 76,91%. Penelitian ini menunjukkan bahwa model LSTM-CNN memiliki akurasi yang lebih baik dibandingkan CNN dan CNN-LSTM (Chen dan Wang, 2019).

Penelitian analisis sentimen pada teks berbahasa Indonesia sebagian besar menggunakan metode *machine learning* klasik, tapi saat ini sudah ada yang menggunakan *deep learning* dengan metode *Deep Belief Network* (DBN). Data yang digunakan dalam penelitian berupa *tweet* media sosial twitter. Hasil dari penelitian ini memperlihatkan bahwa metode DBN memberikan hasil pengujian klasifikasi lebih baik dengan akurasi 93.31%, presisi 93%, recall 93%, f1-score 93%, dan *support* 2378, sedangkan hasil pengujian sistem menggunakan metode *Naive Bayes* memberikan hasil pengujian klasifikasi dengan akurasi 79.10%, presisi 79%, recall 79%, f1-score 79%, *support* 2378 dan SVM dengan akurasi 92.18%, presisi 92%, recall 92%, f1-score 92%, *support* 2378 (Zulfa dan Winarko, 2017).

LSTM memiliki keunggulan dengan adanya *memory cell* yang dapat memperbaharui informasi dan terbukti berkinerja lebih baik pada analisis sentimen berupa teks dibandingkan RNN, sedangkan CNN memiliki keunggulan dapat memfilter informasi yang dominan. Masalah lainnya yaitu, diperlukan metode *pre-training* agar klasifikasi lebih akurat. *Pre-training Word2Vec* dapat menangkap makna semantik teks dengan baik dan setiap kata yang berhubungan dicirikan dengan vektor yang cenderung mirip (Hitesh dkk., 2019). Melihat keunggulan dari LSTM, CNN dan *Word2Vec*, maka model LSTM-CNN digunakan untuk analisis sentimen teks berbahasa Indonesia dengan model *pre-training* menggunakan *Word2Vec* dimana teks pada penelitian ini berasal dari ulasan objek wisata Pulau Bali di situs tripadvisor.com.

2.2 Dasar Teori

2.2.1 Analisis Sentimen

Natural Language Processing (NLP) mempelajari dan mengembangkan algoritma dan sistem yang memungkinkan komputer memahami dan melakukan tugas yang melibatkan bahasa manusia. NLP disebut juga linguistik komputasional, *computer speech* dan pengolahan bahasa. NLP dapat menganalisis bahasa manusia baik dalam bentuk tertulis maupun suara sehingga didapatkan informasi yang berguna. NLP bertujuan mengatasi masalah komputer dalam memahami bahasa alami manusia yang memiliki peraturan gramatikal dan semantik dengan cara mengubah bahasa alami manusia menjadi representasi data yang dapat dipahami dan dapat diolah oleh komputer (Pustejovsky dan Stubbs, 2012).

Analisis sentimen dikenal pula dengan *opinion mining* yang bertugas menganalisis opini, ekspresi dan perasaan para audiens terhadap suatu entitas. Opini selanjutnya diteliti termasuk positif, negatif ataupun netral. Analisis sentimen mampu menganalisis data teks seperti *tweet*, ulasan dalam media sosial, blog, dan *website* (Pang dan Lee, 2008). Analisis sentimen berupa teks ulasan bertujuan untuk menentukan arah ekspresif ulasan pengguna atau pelanggan. Penelitian analisis sentimen menjadi populer dikarenakan meningkatnya kebutuhan untuk menganalisis informasi tersembunyi dari ulasan pelanggan atau pengguna produk

yang tersebar di media *online* dan situs internet dalam bentuk data yang tidak terstruktur (Luo dkk., 2016).

Analisis sentimen melakukan analisis data menggunakan metode klasifikasi. Klasifikasi memproses data dengan mengelompokkan data kedalam suatu kelas yang telah didefinisikan sebelumnya. Tujuan utama analisis sentimen adalah menemukan polaritas dari suatu data, sehingga proses analisis data dengan bentuk klasifikasi digunakan untuk memprediksi suatu data masuk kedalam kelas positif atau negatif atau netral. Terdapat tiga level dalam analisis sentimen yang menjadi acuan dalam sebuah penelitian, yaitu level dokumen, level kalimat dan level aspek (Liu, 2015).

Level dokumen memiliki tujuan untuk mengklasifikasikan satu keseluruhan dokumen termasuk kelas positif atau kelas negatif. Contoh analisis sentimen opini publik terhadap suatu tokoh, maka hasilnya berupa sentimen positif atau negatif dari tokoh tersebut berdasarkan satu kesatuan komentar. Kelebihan level dokumen adalah mampu menentukan politas secara utuh (Liu, 2015).

Level kalimat menentukan kalimat yang menjadi kalimat sentimen terlebih dahulu. Kalimat sentimen selanjutnya dianalisis apakah kalimat termasuk kelas positif atau kelas negatif. Diasumsikan dalam suatu dokumen terdapat beberapa kalimat, kemudian ditentukan lebih dulu mana yang merupakan kalimat opini yang berarti bukan kalimat fakta, selanjutnya ditentukan kalimat opini tersebut termasuk ke dalam kelas positif atau negatif (Liu, 2015).

Level aspek merupakan level terendah dimana proses analisis dilakukan lebih mendalam. Pada level dokumen dan kalimat penentuan polaritas dokumen tidak mampu menggambarkan target opini. Dalam suatu dokumen ulasan bisa saja terdapat lebih dari satu entitas yang menjadi target opini dimana masing-masing target opini berdiri sendiri dalam penentuan polaritasnya. Hal ini memungkinkan dalam satu dokumen dapat masuk kelas positif dan kelas negatif sekaligus. Misalkan terdapat kalimat berikut “tempat ini sangat indah, tetapi jalan menuju ke sana cukup berbahaya” kalimat ini menunjukkan terdapat dua entitas dari ulasan, yaitu terkait tempat dan akses. Pada kalimat tersebut aspek tempat termasuk kelas positif, sedangkan aspek akses termasuk kelas negatif (Liu, 2015).

2.2.2 Prapengolahan Teks

Data teks merupakan data yang tidak beraturan karena terdapat perulangan kata dan munculnya banyak kata yang tidak berkontribusi pada analisis data. Prapengolahan teks perlu dilakukan untuk membersihkan data teks terlebih dahulu sebelum dilakukan proses analisis sentimen. Prapengolahan teks menghilangkan data yang tidak konsisten, data yang duplikat, dan data yang tidak berpengaruh terhadap polaritas suatu dokumen. Tahapan proses prapengolahan teks antara lain:

1. *Case folding*

Proses mengubah semua karakter huruf pada sebuah kalimat menjadi huruf kecil atau huruf besar disebut *case folding*. *Case folding* yang dilakukan pada penelitian ini yaitu mengubah seluruh *dataset* menjadi huruf kecil. Huruf kapital biasanya terdapat pada setiap awal kalimat seperti "Udara di tempat ini sangat sejuk", menggunakan *case folding* kalimat tersebut berubah menjadi "udara di tempat ini sangat sejuk". Tujuan utama c*ase folding* adalah agar kata "udara" tidak lagi mempunyai dua bentuk yaitu, "Udara", dan "udara", namun hanya memiliki satu bentuk huruf kecil saja (Hidayatullah dan Ma'arif, 2016).

2. Tokenisasi

Tokenisasi merupakan proses untuk memecah dokumen teks menjadi *token.* Tokenisasi memiliki kemampuan untuk memecah dokumen menjadi kata, frasa, simbol atau elemen lain yang memiliki makna. Optimalisasi *token* dapat dilakukan dengan cara menghilangkan karakter-karakter ilegal pada dokumen seperti tanda baca, simbol, angka, html, dan *mention.* Contoh karakter ilegal yang dihilangakan antara lain %, &, >, (, {,], 1-9, *@uluwatu, http://tripadvisor.com* (Symeonidis dkk., 2018).

3. *Stopword Removal*

Stopword Removal merupakan tahap pengambilan kata-kata penting dan membuang kata-kata yang dianggap tidak penting. Cara untuk membuang kata yang tidak penting disebut *stopword removal. Stopword removal* bertujuan untuk

menghilangkan kata-kata yang sering muncul namun tidak memiliki kontribusi dalam proses analisis data. *Stopword removal* berusaha memperkecil dimensi data dan mempercepat waktu komputasi (Symeonidis dkk., 2018). Contoh kata yang tidak penting di bahasa Indonesia seperti kata “dan”, ”yang”, ”di”, ”ke”.

4. *Stemming*

Stemming merupakan proses memetakan variasi kata ke bentuk dasar. Proses *stemming* dilakukan dengan menghapus imbuhan, baik awalan maupun akhiran dari suatu kata untuk mendapatkan kata dasarnya. *Stemming* yang umum digunakan pada teks berbahasa Indonesia menggunakan *library stemmer* Sastrawi yang dikembangkan berdasarkan algoritma Nazief-Adriani (Hidayatullah dan Ma’arif, 2016).

Stemmer Sastrawi merupakan *library stemmer* yang menerapkan algoritma Nazief-Adriani, kemudian dikembangkan oleh algoritma *Confix Stripping* (CS), dan selanjutnya dikembangkan oleh algoritma *Enhanced Confix Stripping* (ECS), lalu dikembangkan lagi menjadi *Modified* ECS (Asian dkk., 2005). Menurut Asian dkk., algoritma yang digunakan *stemmer* Sastrawi dikembangkan berdasarkan morfologi Bahasa Indonesia yang mengelompokkan beberapa imbuhan ke dalam kategori sebagai berikut:

a. *Inflection suffixes* merupakan imbuhan berupa akhiran yang tidak merubah bentuk data dasar. Akhiran ini dibagi menjadi dua yaitu:
   - *Particle* (P) yang merupakan partikel yaitu “-lah”, “-kah”, “-tah”, dan “-pun”.
   - *Possesive Pronoun* (PP) yang merupakan kata ganti kepunyaan yaitu “-ku”, “-mu”, dan “-nya”.
b. *Derivation suffixes* (DS) merupakan imbuhan berupa akhiran yang ditambahkan secara langsung pada kata dasar. Akhiran ini merupakan akhiran asli Bahasa Indonesia yaitu “-i”, “-kan”, dan “-an”.
c. *Derivation prefixes* (DP) merupakan imbuhan berupa awalan yang dapat langsung ditambahkan pada kata dasar maupun pada kata dasar yang telah mendapat penambahan sampai 2 awalan. Kedua awalan itu yaitu:

- Awalan yang bermorfologi (“me-”, “be-”, “te-”, dan “pe-”)
- Awalan yang tidak bermorfologi (“di-“, “ke-“, dan “se-“)

Berdasarkan kategori, bentuk kata berimbuhan dalam Bahasa Indonesia dijelaskan sebagai berikut:

[DP + [DP + [DP +]]] Kata Dasar [[+ DS] [+ PP] [+ P]]

Berdasarkan model imbuhan dan aturan morfologi dalam Bahasa Indonesia, aturan-aturan yang digunakan dalam proses *stemming* Sastrasi sebagai berikut:

a. Beberapa kombinasi imbuhan berupa awalan dan akhiran tidak diperbolehkan, diantaranya “be-i”, “di-an”, “ke-i”, “ke-kan”, “me-an”, “se-i”, “se-kan”, dan “te-an”.
b. Tidak diperbolehkan menggunakan imbuhan yang sama secara berulang.
c. Kata yang terdiri dari satu atau dua huruf maka tidak dilakukan *stemming.*
d. Bentuk asli kata dasar dapat berubah jika diberi awalan tertentu ataupun awalan yang telah diberikan sebelumnya pada kata dasar (bermorfologi).
d. Pembersihan data *(data cleaning),* merupakan proses untuk menghilangkan data yang mengandung *noise* dan data tidak relevan.

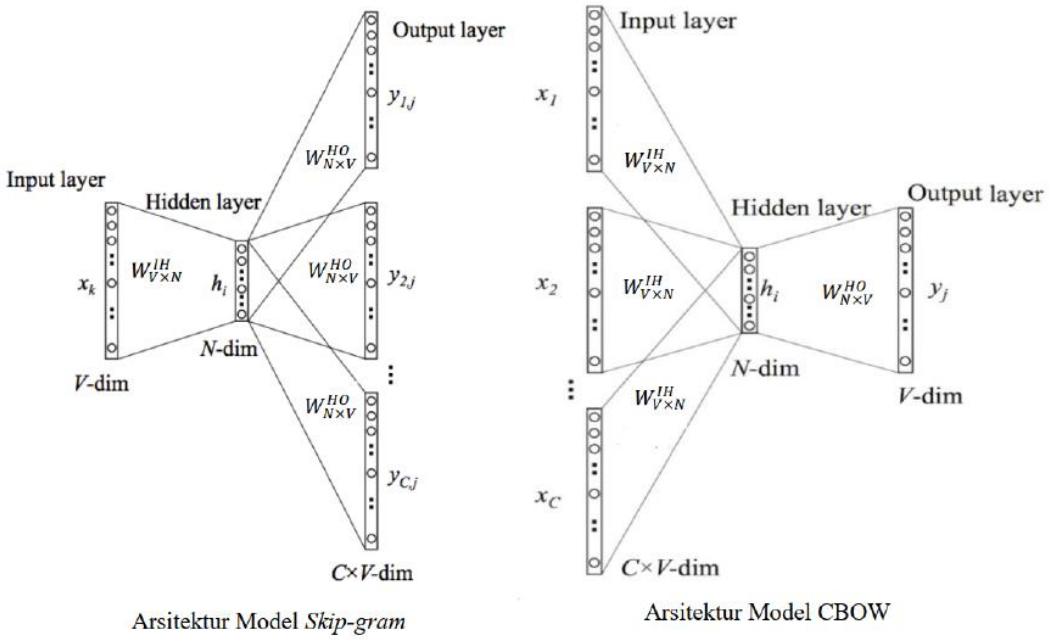
5. *Padding*

Proses pembelajaran yang dilakukan oleh *neural network* memerlukan masukan data dengan panjang yang sama. *Padding* merupakan proses yang dilakukan untuk membuat *input* mempunyai panjang yang sama dengan cara menambahkan kata “<pad>”. *Dataset* pada penelitian ini memiliki panjang teks yang berbeda-beda. Oleh karena itu, perlu dilakukan *padding* agar vektor memiliki panjang yang sama sebelum diproses pada *neural network* (Giménez dkk., 2020).

2.2.3 *Word2Vec*

Model *Word2Vec* merupakan model *word embedding* yang diusulkan oleh Mikolov, Sutskever, Chen, Corrado, dan Dean pada tahun 2013. *Word2Vec* memiliki kemampuan membaca dan mengolah teks dalam ukuran yang besar dan mengubah setiap kata menjadi vektor. Model *Word2Vec* mampu memahami sintaks dan makna semantik kata dari bahasa alami kemudian merepresentasikan setiap kata dengan sebuah vektor. *Word2Vec* mencapai kinerja terbaik dalam NLP dengan mengelompokan kata serupa yang memiliki vektor yang sama (Al-Amin dkk., 2017).

Neural Network digunakan pada model *Word2Vec* untuk menghasilkan *output* berupa ruang vektor dari *input* yang berupa korpus teks. Model *Word2Vec* memiliki dua jenis arsitektur bernama *Continous Bag of Words* (CBOW) dan *Skip-gram*. Arsitektur CBOW memprediksi kata saat ini berdasarkan konteks, sedangkan arsitektur *Skip-gram* memprediksi dalam jangkauan sebelum atau setelah kata sekarang dimana kata sekarang merupakan *input*, sehingga arsitektur *Skip-gram* dianggap sebagai arsitektur yang efisien dalam mempelajari vektor kata yang tidak terstruktur dalam jumlah besar (Nawangsari dkk., 2019). Arisitekur *Skip-gram* dan CBOW dapat dilihat pada Gambar 2.1.



Gambar 2.1 *Skip-gram* dan CBOW (Nawangsari dkk., 2019)

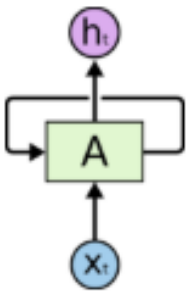
2.2.4 *Deep Learning*

Deep learning merupakan area baru dalam penelitian *machine learning* yang diperkenalkan dengan tujuan menggerakkan *machine learning* lebih dekat dengan salah satu tujuan aslinya yaitu *artificial intelligence*. Keberhasilan algoritma m*achine learning* umumnya bergantung pada representasi data. *Deep learning* mengimplementasikan *deep architecture* yang dapat merepresentasikan data secara efisien dengan unit komputasi yang lebih sedikit untuk fungsi sama dengan cara meningkatkan kinerja pembelajaran, aksesibilitas, dan data latih. Melalui pembelajaran beberapa tingkat representasi dan abstraksi, sistem *deep learning* dapat mempelajari fungsi pemetaan yang kompleks dan membantu untuk pengenalan suara, pengenalan citra, dan pemrosesan bahasa alami (Ain dkk., 2017).

Deep learning merupakan pembelajaran mesin yang mengacu pada *deep neural network*, yang diusulkan pertama kali oleh G.E. Hinton pada tahun. *Neural network* mengadaptasi sistem otak manusia yang tersusun dari beberapa neuron yang membentuk jaringan. *Neural Network* bermanfaat dalam representasi vektor, estimasi representasi kata, klasifikasi kalimat, pemodelan kalimat, presentasi fitur, dan lainnya. *Deep learning* melakukan pelatihan baik *supervised* maupun *unsupervised* (Vateekul dan Koomsubha, 2016). *Deep learning* mencakup banyak jaringan seperti CNN *(Convolutional Neural network)*, RNN *(Recurrent Neural network)* dengan pengembangannya LSTM (Long-Short Term Memory), *Recursive Neural network*, dan DBN *(Deep Belief Networks)* (Zhang dkk., 2016).

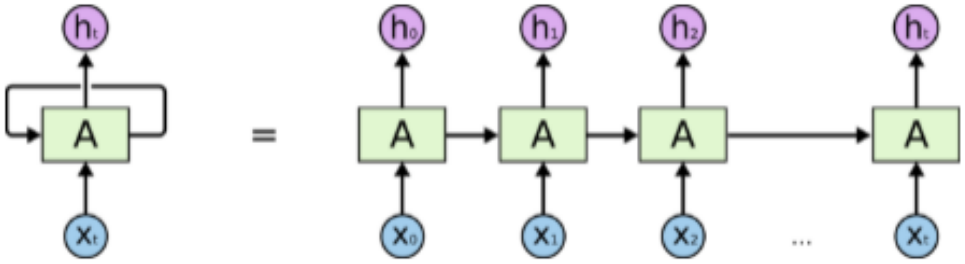
2.2.5 *Recurrent Neural Network (RNN)*

Recurrent neural network (RNN) merupakan salah satu metode dari *deep neural network* yang mampu memproses sequential data. Manusia dalam mengambil keputusan tidak dengan mengolah informasi setiap saat, melainkan selalu memperhitungkan informasi masa lalu. RNN meniru manusia dalam pengambilan keputusan dengan cara menyimpan informasi dari masa lalu dengan melakukan *looping* dalam arsitekturnya seperti terlihat pada Gambar 2.2 yang secara otomatis membuat informasi dari masa lalu tetap tersimpan dan dapat digunakan jika diperlukan (Li dan Qian, 2016).



Gambar 2.2 Proses perulangan informasi RNN (Smagulova dan James, 2019)

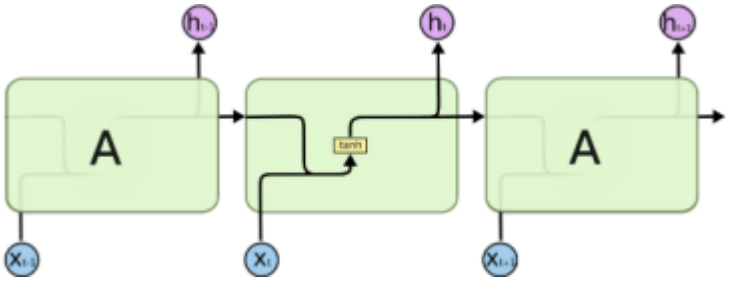
Xt sebagai *input*, ht sebagai *output* dan terdapat perulangan yang memungkinkan informasi dilewatkan dari satu langkah ke langkah berikutnya. *Recurrent Neural network* dapat diasumsikan memiliki multi salinan dari jaringan yang sama. Selanjutnya, masing-masing jaringan mengirimkan pesan kepada jaringan berikutnya seperti terlihat pada gambar 2.3.



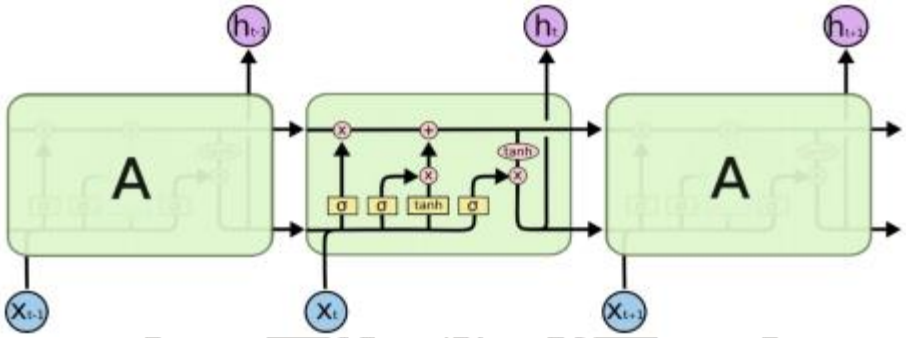
Gambar 2.3 Multi salinan jaringan RNN (Smagulova dan James, 2019)

2.2.6 *Long Short Term Memory*

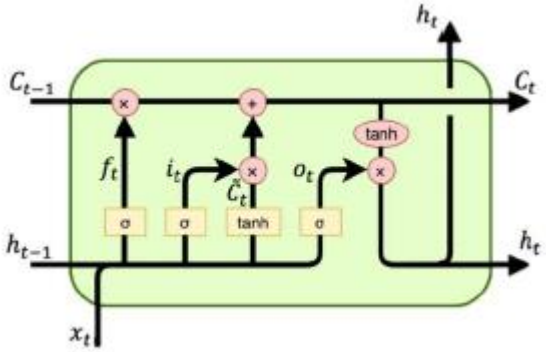
Long Short Term Memory (LSTM) dikenalkan oleh Hotchreiter dan Schmidhuber (1997) untuk mengatasi masalah difusi gradien *Recurrent Neural network* (RNN). LSTM merupakan salah satu variasi dari RNN yang dibuat untuk menghindari masalah mengingat informasi jangka panjang pada RNN. Jaringan perulangan RNN hanya menggunakan satu *layer* sederhana, yaitu *layer* tanh seperti pada gambar 2.4, sedangkan LSTM memiliki empat *layer* pada perulangan modelnya seperti pada Gambar 2.5.



Gambar 2.4 *Layer tanh* pada RNN (Smagulova dan James, 2019)



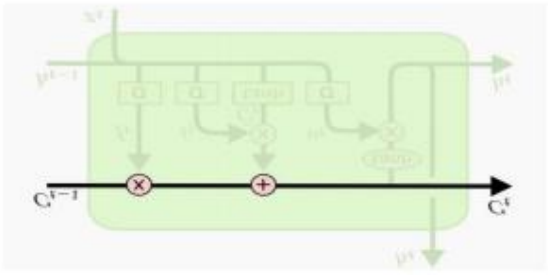
Gambar 2.5 Perulangan empat *layer* pada LSTM (Smagulova dan James, 2019)



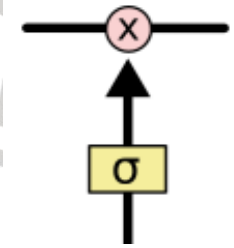
Gambar 2.6 Ilustrasi *Cell* LSTM (Smagulova dan James, 2019)

Cell pada LSTM dapat dilihat pada Gambar 2.6 yang terdiri dari *layer neuron* dilambangkan dengan persegi panjang, operasi *element-wise* dilambangkan dengan lingkaran. Panah hitam melambangkan aliran informasi di dalam *cell* dan antar *cell* maupun keluaran dari *cell* (*output* h). *Cell* LSTM mempunyai 2 hasil keluaran, yang pertama yaitu informasi yang sebenarnya *hidden state* (*ht*) yang diteruskan ke *cell* selanjutnya dan menjadi *input* dari *cell* selanjutnya, yang kedua yaitu *cell state* (Ct). *Cell state* merupakan kunci utama dari LSTM. *Cell state*

merupakan garis horizontal yang menghubungkan semua *output layer* pada LSTM seperti terlihat pada gambar 2.7. LSTM memiliki kemampuan untuk menambah dan menghapus informasi dari *cell state* yang disebut *gates*. *Gates* berfungsi mengatur informasi masuk secara opsional dengan menggunakan *sigmoid layer* yang digambarkan pada Gambar 2.8. Keluaran dari *sigmoid layer* menunjukkan informasi diteruskan atau diberhentikan. Angka 0 untuk diberhentikan dan angka 1 untuk diteruskan.



Gambar 2.7 *Cell state* pada LSTM (Smagulova dan James, 2019)



Gambar 2.8 Sigmoid *layer* pada LSTM (Smagulova dan James, 2019)

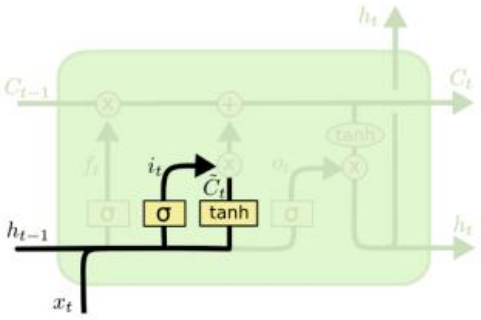
LSTM memiliki tiga jenis *gates* yaitu, *forget gate* (it), *input gate* (ft), dan *output gate* (Ot). *Forget gate* berfungsi untuk memutuskan informasi yang dihapus dari *cell*. *Input gate* berfungsi untuk memutuskan nilai dari *input* untuk diperbaharui pada *state* memori. *Output gate* berfungsi untuk memutuskan apakah yang dihasilkan *output* sesuai dengan *input* dan memori pada *cell* atau tidak.

Proses berjalannya metode LSTM ada empat langkah (Smagulova dan James, 2019):

1. Langkah pertama bertujuan untuk memutuskan informasi yang akan disimpan di *cell state*. Langkah pertama memiliki dua bagian. Bagian pertama *sigmoid layer* yang bernama *input gate layer* untuk memutuskan nilai yang akan diperbaruhi. Bagian kedua *tanh layer* untuk membuat satu kandidat dengan nilai baru ($\widetilde{\boldsymbol{C}}_t$) yang dapat ditambahkan ke *cell state*. Tahap selanjutnya *output* dari *input gate layer* dan *tanh layer* akan digabungkan untuk memperbaruhi *cell state*. Langkah kedua digambarkan pada gambar 2.9, sedangkan persamaan *input gate* diuraikan pada persamaan 2.1 dan persamaan kandidat baru diuraikan pada persamaan 2.2.

$$\widetilde{\boldsymbol{C}}_t = tanh(\boldsymbol{W}_c\boldsymbol{x}(t) + \boldsymbol{U}_c\boldsymbol{h}(t-1)) + \boldsymbol{b}_c \quad (2.1)$$

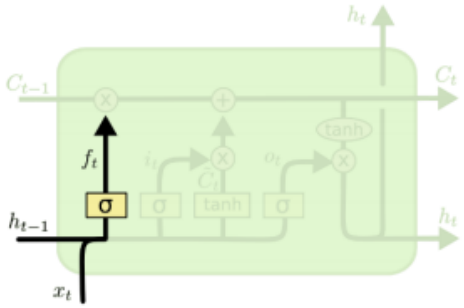
$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{U}_i\boldsymbol{h}(\boldsymbol{t}-\boldsymbol{1}) + \boldsymbol{b}_i \quad (2.2)$$



Gambar 2.9 *Input gate layer* dan tanh *layer* (Smagulova dan James, 2019)

2. Langkah kedua LSTM memutuskan informasi apa yang akan dihapus dari *cell state*. Keputusan ini dibuat oleh *sigmoid layer* yang bernama *forget gate layer*. *Forget gate layer* akan memproses $\boldsymbol{h}(t-1)$ dan $\boldsymbol{x}(t)$ sebagai *input*, dan menghasilkan *output* berupa angka 0 atau 1 pada *cell state* Ct-1 seperti pada Gambar 2.10, adapun persamaan *forget gate* diuraikan pada persamaan 2.3.

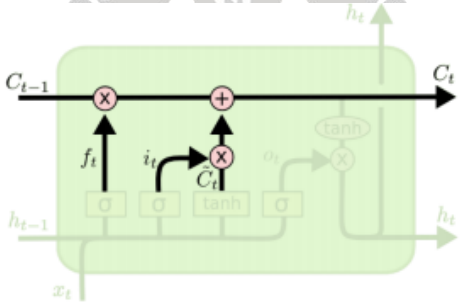
$$f_t = \sigma\left(\boldsymbol{W}_f\boldsymbol{x}(t) + \boldsymbol{U}_f\boldsymbol{h}(t-1) + \boldsymbol{b}_f\right) \quad (2.3)$$



Gambar 2.10 *Forget gate layer* (Smagulova dan James, 2019)

3. Langkah ketiga bertujuan untuk memperbaruhi *cell state* yang lama (Ct-1) menjadi *cell state* baru, Ct seperti pada Gambar 2.11. Perkalian *state* lama dengan $\boldsymbol{f}_t$ bertujuan untuk menghapus informasi yang sudah ditentukan sebelumnya pada langkah *forget gate layer*, selanjutnya ditambahkan dengan $\boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t$ yang merupakan nilai baru dan digunakan untuk memperbaruhi *state*. Persamaan *cell state* diuraikan pada persamaan 2.5.

$$C_t = \boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t + \boldsymbol{f}_t * C_{t-1} \tag{2.5}$$

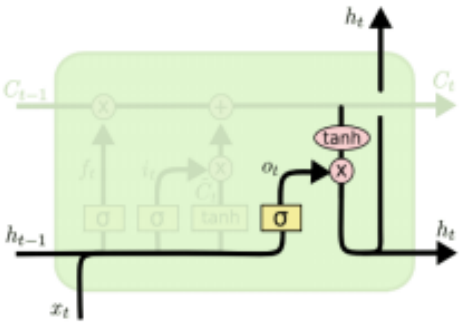


Gambar 2.11 Pembuatan *cell state* baru (Smagulova dan James, 2019)

4. Langkah keempat yang merupakan langkah terakhir dalam metode LSTM bertujuan untuk memutuskan hasil *output* seperti Gambar 2.12. *Output* harus sesuai dengan *cell state* yang telah diproses terlebih dahulu. Pertama, *sigmoid layer* memutuskan bagian dari *cell state* yang menjadi *output*. Kedua, *output* dari *cell state* dimasukkan ke dalam *tanh layer* (untuk mengganti nilai menjadi diantara -1 dan 1) dan dikalikan dengan *sigmoid gate* agar *output* yang dihasilkan sesuai dengan apa yang diputuskan sebelumnya. Persamaan *output gate* diuraikan pada persamaan 2.6 dan 2.7. Seluruh keterangan notasi pada langkah LSTM dijelaskan pada Tabel 2.2.

$$\boldsymbol{O_t} = \sigma(\boldsymbol{W_o x(t)} + \boldsymbol{U_o h(t-1)} + \boldsymbol{b_o} \quad \textbf{(}2.6)$$

$$\boldsymbol{h_t} = \boldsymbol{O_t} * \tanh(\boldsymbol{C_t}) \quad (2.7)$$



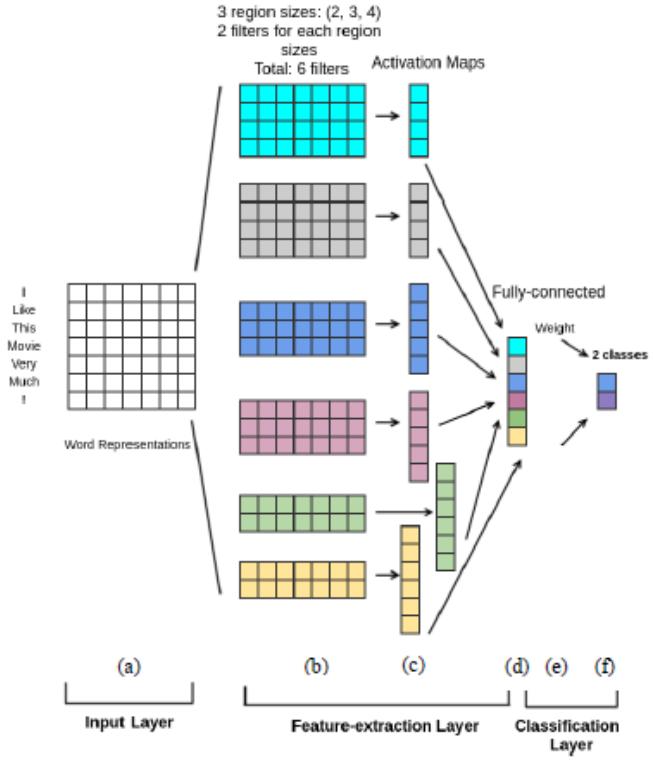
Gambar 2.12 Penentuan *output* (Smagulova dan James, 2019)

Tabel 2.2 Keterangan notasi dan definisi persamaan LSTM

Notasi	Definisi
$\boldsymbol{t}$	*Time step t = 1,2, ...*
$\boldsymbol{x(t)}$ atau $\boldsymbol{x_t}$	Vektor masukan pada saat *time step* ke t
$\boldsymbol{h(t-1)}$	*Hidden state* pada *time step* ke t-1
$\boldsymbol{h_t}$	*Hidden state time step ke t*
$\boldsymbol{C_{t-1}}$	*Cell memory time step* ke t-1
$\boldsymbol{C_t}$	*Cell memory time step* ke t
$\boldsymbol{W_c}$	Bobot pada *candidate layer* untuk *input* kata
$\boldsymbol{W_i}$	Bobot pada *input layer* untuk *input kata*
$\boldsymbol{W_o}$	Bobot pada *output layer* untuk *input kata*
$\boldsymbol{W_f}$	Bobot pada *forget layer* untuk *input kata*
$\boldsymbol{U_c}$	Bobot pada *candidate layer* untuk *hidden state time step* sebelum
$\boldsymbol{U_i}$	Bobot pada *input layer* untuk *hidden state time step* sebelum
$\boldsymbol{U_o}$	Bobot pada *output layer* untuk *hidden state time step* sebelum
$\boldsymbol{U_f}$	Bobot pada *forget layer* untuk *hidden state time step* sebelum
$*$	*Hadamard product*

2.2.7 *Convolutional Neural Network* (CNN)

Convolutional Neural network (CNN) merupakan salah satu metode *deep learning* dari pengembangan *Multi Layer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural network* karena memiliki tingkat jaringan yang dalam. CNN memiliki dua metode yakni klasifikasi menggunakan *feed forward* dan tahap pembelajaran menggunakan *backpropagation*. Cara kerja CNN seperti Gambar 2.13 memiliki kesamaan dengan MLP, namun pada CNN neuron dipresentasikan dalam dua dimensi, sedangkan MLP setiap neuron hanya berukuran satu dimensi (Nasichuddin dkk., 2018).



Gambar 2.13 Arsitektur CNN (Nasichuddin dkk., 2018)

CNN awalnya dirancang untuk pengenalan citra gambar tetapi pada perkembangannya menjadi model serbaguna yang digunakan untuk beragam tugas. CNN memiliki kemampuan mengenali fitur-fitur lokal di dalam bidang multi dimensi. Misalnya pada gambar, CNN akan menemukan fitur tertentu seperti roda atau senyuman, di manapun lokasinya. CNN seperti diuraikan pada Gambar 2.13

bekerja dengan memasukkan data multidimensi (misal; gambar, *embedding* kata) ke lapisan konvolusional yang akan terdiri dari beberapa filter yang akan mempelajari fitur yang berbeda. *Output* biasanya dikumpulkan dan diberikan kepada dimensi yang lebih kecil, kemudian diumpankan ke lapisan yang terhubung (Yenter dan Verma, 2017).

Arsitektur CNN memiliki 3 *layer* yaitu, *input layer*, *feature-extraction layer*, dan *clasification layer* seperti pada Gambar 2.13. Arsitektur CNN dicirikan dengan adanya beberapa parameter, misalnya ukuran wilayah filter, jumlah filter, fungsi aktivasi, dan bentuk regularisasi. Setiap *layer* memiliki lapisan yang berbeda-beda, lapisan utama dari *layer-layer* pada arsitektur CNN yaitu, *convolutional layer*, *pooling layer*, dan *fully connected*. Setiap arsitektur dapat disesuaikan dan dimodifikasi sesuai kebutuhan. Perubahan arsitektur ini dapat mengubah akurasi dari model (Nasichuddin dkk., 2018). Berikut penjelasan dari masing-masing *layer* pada CNN:

1. *Input layer*

   Setiap kata terdapat pada sebuah dokumen dipetakan dalam matriks. *Cell* yang bernilai 1 merepresentaikan posisi kata, sedangkan *cell* tidak berisi kata direpresentasikan dengan nilai 0.

2. *Feature-extraction Layer*

   Lapisan-lapisan yang terdapat dalam *feature* menstranlasikan suatu *input* menjadi *feature* berdasarkan ciri dari *input* yang terbentuk dari angka-angka dalam vektor. *Feature-extraction layer* terdiri dari *convolutional layer* dan *pooling layer*.

   a. *Convolutional layer*

      Convolutional layer menghitung *output* dari neuron yang terhubung ke daerah lokal dalam *input*, masing-masing menghitung produk titik antara bobot mereka dan wilayah kecil yang terhubung ke dalam volume *input*. Hasil dari *convolutional layer* selanjutnya diproses oleh *activation function.* Terdapat fungsi aktivasi yang digunakan ada yaitu, fungsi aktivasi *Tanh* dan fungsi aktivasi *ReLU*.

b. *Pooling layer*

Poling layer merupakan lapisan yang mengurangi dimensi dari *feature map* atau lebih dikenal dengan langkah untuk *downsampling*, sehingga mempercepat komputasi karena parameter yang harus diperbaharui semakin sedikit dan dapat mengatasi *overfitting*. *Pooling* yang biasa digunakan adalah *max pooling* dan *average pooling*. *Max pooling* untuk menentukan nilai maksimum tiap pergeseran filter, sementara *average pooling* menentukan nilai pergeseran filter berdasrkan rata-rata.

3. *Classification Layer*

Lapisan ini berfungsi untuk mengklasifiksikan tiap neuron yang telah diekstraksi pada fiitur sebelumnya. Terdiri dari:

a. Flattern

Membentuk ulang fitur (*reshape feature map*) menjadi sebuah vektor agar bisa digunakan sebagai *input* dari *fully connected layer*.

b. Fully Connected (FC)

Lapisan FC menghitung skor akhir, seperti jaringan saraf biasa dan seperti namanya, setiap neuron dalam lapisan ini terhubung ke semua angka dalam volume.

c. *Output layer*

Terdapat dua jenis fungsi dalam *output layer* yaitu *sigmoid function* dan *softmax function. Sigmoid function* mengambil nilai bilangan real dan mengembalikan nilai *output* dalam rentang 0 hingga 1. *Sigmoid* pada umumnya digunakan untuk aktivasi fungsi *output* pada *binary classification.* Persaman *sigmoid* dijelaskan pada persamaan (2.9).

$$f(x_i) = \frac{1}{1+ e^{(-x_j)}} \quad (2.9)$$

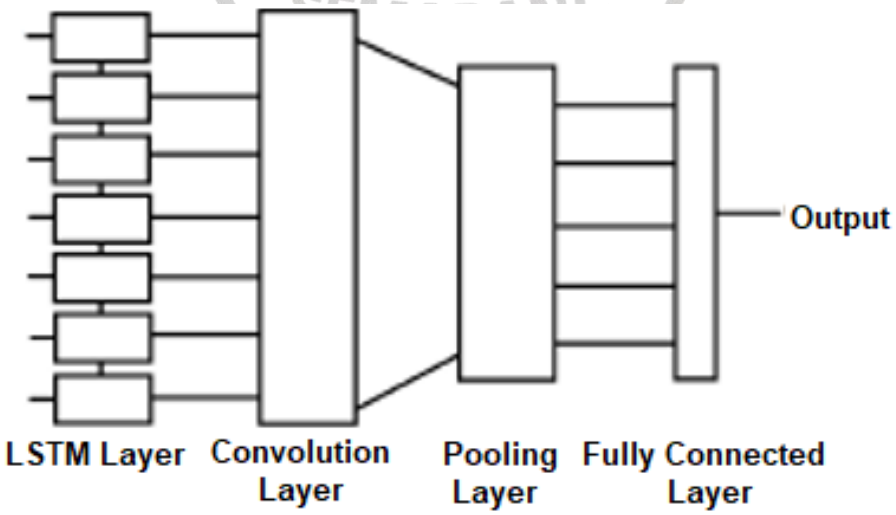
Softmax function menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan untuk menentukan kelas target untuk *input* yang diberikan. Rentang probabilitas *output* memiliki rentang antara 0

hingga 1, dan jumlah semua probabilitas akan sama dengan 1. Fungsi ini cocok untuk digunakan pada *multi classification* karena fungsi ini mengembalikan probabilitas dari masing-masing kelas dan kelas target akan memiliki probabilitas yang tinggi. Persamaan *softmax* dijelaskan pada persamaan (2.10)

$$f(x_i) = \frac{e^{(x_i)}}{\sum_{j=0}^{k} e^{(x_j)}} \tag{2.10}$$

2.2.8 *Long Short Term Memory-Convolutional Neural Network*

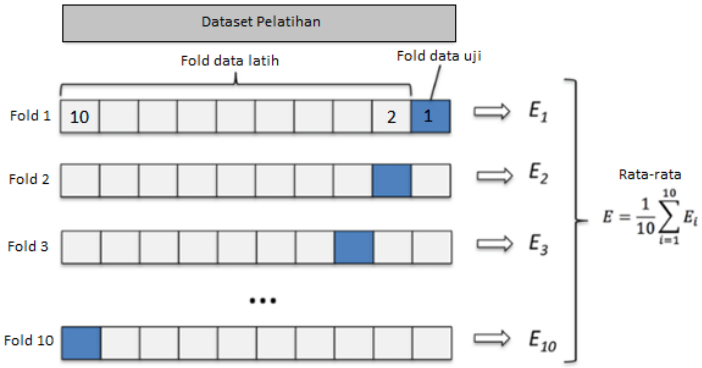
Model LSTM-CNN terdiri dari lapisan awal LSTM yang akan menerima *embedding* kata untuk setiap token. Token kerluarannya akan menyimpan informasi tidak hanya dari token awal, tetapi juga sebelumnya. *Output* dari LSTM dimasukkan ke dalam lapisan konvolusi yang diharapkan dapat mengestrak fitur local dengan baik. *Output* dari LSTM selanjutnya menjadi data *input* dari CNN. *Output* lapisan CNN akan digabungkan ke dimensi yang lebih kecil dan akhirnya dikeluarkan sebagai label positif atau negatif (Chen dan Wang, 2019). Gambar 2.14 menampilkan arsitektur LSTM-CNN dari mulai *input* hingga *output.*



Gambar 2.14 Arsitektur LSTM-CNN (Chen dan Wang, 2019)

2.2.9 *K-fold Cross Validation*

Metode *cross validation* memiliki kelebihan tidak adanya masalah dalam pembagian data karena semua data berperan sebagai data latih dan berperan juga sebagai data uji. Pada metode *K-fold cross validation* data dibagi sebanyak K dengan menghasilkan jumlah data yang sama besar. Data tersebut dibagi perannya sebagai data latih dan data uji yang kemudian diproses secara bergantian sebanyak K kali. Simulasi pada Gambar 2.15 menunjukann *K-fold cross validation* dengan K sama dengan 10.



Gambar 2.15 Simulasi 10-*fold cross validation*

2.2.10 *Confusion Matrix*

Evaluasi dibutuhkan untuk mengetahui performa dari model yang dikembangkan. Teknik evaluasi yang diterapkan pada penelitian ini adalah dengan menghitung *accuracy* menggunakan *confusion matrix*. *Accuracy* digunakan untuk mengukur seberapa baik dan akurat model yang dihasilkan dalam memprediksi label kelas pada *tuple*. *Confusion matrix* memiliki kelebihan dalam menganalisis bagaimana klasifier dapat mengenali *tuple* dari kelas yang berbeda (Han dkk., 2012). Tabel 2.3 menunjukkan *confusion matrix* untuk klasifikasi dua kelas.

Tabel 2.3 Tabel *confusion matrix* pada dua kelas

Parameter		**Prediksi**	
		Negatif(-)	Positif(+)
Aktual	Negatif(-)	TN	FN
	Positif(+)	FP	TP

Keterangan:

- *True Negative* TN jika hasil prediksi negatif dan data aktualnya negatif.
- *True Positive* TP jika hasil prediksi positif dan data aktualnya positif.
- *False Negative* FN jika hasil prediksi negatif dan data aktualnya positif
- *False Positive* FP jika hasil prediksi positif dan data aktualnya negatif

Terdapat beberapa persamaan yang telah ditetapkan pada matriks dua kelas yang memiliki persamaan seperti pada Persamaan 2.9 sampai 2.15.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+FN}}{\boldsymbol{TP+TN+FN+FP}} \quad (2.9)$$

$$\boldsymbol{True\ positif} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (2.10)$$

$$\boldsymbol{False\ positif} = \frac{\boldsymbol{FP}}{\boldsymbol{FP+TN}} \quad (2.11)$$

$$\boldsymbol{True\ negatif} = \frac{\boldsymbol{TN}}{\boldsymbol{TN+FP}} \quad (2.12)$$

$$\boldsymbol{False\ negatif} = \frac{\boldsymbol{FN}}{\boldsymbol{FN+TP}} \quad (2.13)$$

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad (2.14)$$

$$\boldsymbol{Error\ rate} = \frac{\boldsymbol{FP+FN}}{\boldsymbol{TN+FP+FN+TP}} \quad (2.15)$$

Keterangan:

- *Accuracy* AC, proporsi kasus yang diidentifikasi benar dengan jumlah semua kasus.
- *Recall* atau *True positive rate* TP, proporsi kasus positif yang diidentifikasi dengan benar.
- *False positive rate* FP, proporsi kasus negatif yang salah diklasifikasikan sebagai positif.
- *True negative rate* TN, proporsi kasus negatif yang diklasifikasikan dengan benar.
- *False negative rate* FN, proporsi kasus positif yang salah diklasifikasikan sebagai negatif.
- *Precision*, proporsi kasus dengan hasil positif yang benar.
- *Error Rate*, proporsi kasus yang diidentifikasi salah dengan sejumlah semua kasus.