BAB II
STUDI LITERATUR

2.1 Analisis Sentimen

Analisis sentimen adalah sebuah studi yang menganalisis opini, sentimen, evaluasi, perilaku dan emosi tentang sesuatu hal seperti produk, layanan, organisasi, seseorang, individu, isu, acara dan topik tertentu. Berkembangnya konten yang ada di web seperti ulasan, diskusi forum, komentar dan *post* pada *social network sites* membantu meningkatkan konten yang dapat digunakan bagi pengambilan keputusan baik untuk perusahaan ataupun individu [13].

Tugas dasar dalam analisis sentimen adalah mengelompokkan polaritas dari teks yang ada dalam dokumen, kalimat, atau target/tingkat aspek dan menentukan apakah pendapat yang dikemukakan dalam dokumen, kalimat atau target/aspek bersifat positif, negatif atau netral. Lebih lanjut sentimen analisis dapat menyatakan emosional sedih, gembira, atau marah [13].

2.2 Penyelesaian Analisis Sentimen

Bidang kategorisasi teks telah dimulai sejak lama [14], namun kategorisasi berdasarkan sentimen diperkenalkan baru-baru ini di [10] [11] [15] [16].

Analisis sentimen dapat dilakukan pada tiga tingkatan yaitu, tingkat dokumen, kalimat dan aspek/target [15].

a. Level Dokumen

   Analisis Sentimen jenis ini di lakukan pada level dokumen. Secara garis besar fokus utama dari analisis sentimen jenis ini adalah menganggap seluruh isi dokumen sebagai sebuah sentimen positif atau negatif secara keseluruhan.

b. Level Kalimat

   Analisis Sentimen jenis ini di lakukan pada level kalimat atau isi dari dokumen yang ada, karena mempunyai kemungkinan setiap kalimat atau isi dari dokumen mempunyai nilai sentimen, positif ataupun negatif.

c. Level Aspek/Target

   Analisis Sentimen di tingkat dokumen maupun di tingkat kalimat tidak mampu melakukan analisis menyeluruh terhadap suatu kalimat. Jenis analisis semacam itu membutuhkan analisis yang lebih ketat yang bertujuan untuk menentukan sentimen di tingkat target.

2.3 Jaringan Saraf Tiruan

Jaringan saraf tiruan atau *neural network* merupakan sebuah model yang mengadopsi dari sistem saraf otak manusia. Model jaringan saraf tiruan ditunjukkan dengan kemampuannya dalam emulasi, analisis, prediksi dan asosiasi. Kemampuan yang dimiliki jaringan saraf tiruan dapat digunakan untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh atau input yang dimasukkan dan membuat prediksi tentang kemungkinan output yang akan muncul atau menyimpan karakteristik input yang diberikan kepada jaringan saraf tiruan [17]

2.4 *Recurrent Neural Network*

Recurrent neural network (RNN) adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memproses masukan yang biasanya adalah data sekuensial. Seperti manusia secara umum, manusia tidak membuat keputusan secara tunggal setiap saat, manusia akan selalu memperhitungkan masa lalu dalam membuat sebuah keputusan. Begitu juga dengan RNN, informasi dari masa lalu digunakan dalam proses pembelajarannya. Hal inilah yang membedakan RNN dari *Artificial Neural Network* biasa. RNN masuk dalam kategori *deep learning* karena data diproses melalu banyak lapis (*layer*) [18][19].

2.5 *Backpropagation Through Time*

Backpropagation Through Time (BPTT) adalah algoritme pelatihan yang digunakan untuk memperbarui bobot pada *recurrent neural network,* salah satunya digunakan pada LSTM [20]

BPTT mengacu pada dua hal yaitu, metode matematika yang digunakan untuk menghitung turunan dan aplikasi aturan rantai turunan serta algoritme pelatihan

yang digunakan untuk memperbarui bobot jaringan untuk meminimalkan kesalahan [20].

Tujuan dari algoritme pelatihan BPTT adalah untuk memodifikasi bobot jaringan saraf untuk meminimalkan kesalahan luaran dari jaringan yang dibandingkan dengan beberapa luaran yang diharapkan dalam menanggapi masukan yang sesuai dan memungkinkan jaringan untuk diperbaiki sehubungan dengan kesalahan spesifik yang dibuat [20].

Secara konseptual, BPTT bekerja dengan membuka gulungan semua catatan waktu masukan. Setiap catatan waktu memiliki satu catatan waktu masukan, satu salinan jaringan, dan satu keluaran. *Error* kemudian dihitung dan diakumulasikan untuk setiap catatan waktu. Jaringan digulung kembali dan bobot diperbarui [20].

Secara umum algoritme BPTT dapat dijelaskan sebagai berikut:

1. Sajikan pola *input* pelatihan dan sebarkan melalui jaringan untuk mendapatkan hasil.
2. Bandingkan *output* yang diprediksi dengan *output* yang diharapkan dan hitung kesalahannya.
3. Hitung turunan kesalahan terkait dengan bobot jaringan.
4. Sesuaikan bobot untuk meminimalkan kesalahan.
5. Ulangi langkah di atas.

Nilai *error* dapat dilakukan *propagated backward* dengan Persamaan 2.1.

$$\delta pj\ (t\ -\ 1) =\ Xm\ h\ \sum_{h}^{m} \delta_{ph}(t) u_{hj}\ f'\ (s_{pj}\ (t\ -\ 1)) \tag{2.1}$$

Dimana h adalah indeks untuk *hidden node* pada langkah waktu t, dan j untuk *hidden node* pada langkah waktu t - 1. *Error deltas* dari bobot lapisan yang lebih tinggi kemudian dapat dihitung secara rekursif.

Setelah semua *Error deltas* diperoleh, bobot dilipat kembali dengan menambahkan hingga satu perubahan besar untuk setiap bobot yang tidak dilipat.

Error deltas dari *output* layer di representasikan oleh Persamaan 2.2.

$$e_o(t)\ =\ d(t)\ -\ y(t) \tag{2.2}$$

Keluaran matriks bobot W diperbarui dengan Persamaan 2.3.

$$W(t+1) = W(t) + \eta s(t) e_o(t)^T \quad (2.3)$$

Selanjutnya, gradien dari *error* di *propagated* dari *output* layer ke *hiddent* layer dengan Persamaan 2.4.

$$eh(t) = dh(e_o(t)^T V, t) \quad (2.4)$$

Dimana *error vector* diperoleh menggunakan fungsi dh (·) yang diterapkan dengan Persamaan 2.5.

$$d_{hj}(x,t) = xf'(net_j) \quad (2.5)$$

Bobot V antara *input* layer dan *hiddent* layer diperbarui dengan Persamaan 2.6.

$$V(t+1) = V(t) + \eta x(t) e_h(t)^T \quad (2.6)$$

Bobot *recurrent* U diperbarui dengan Persamaan 2.7.

$$U(t+1) = U(t) + \eta s(t-1) e_h(t)^T \quad (2.7)$$

Di atas menunjukkan notasi matriks-vektor pada algoritme *backpropagation* konvensional untuk *recurrent neural network*. Untuk pelatihan BPTT, *error propagation* dilakukan secara rekursif dengan Persamaan 2.8.

$$eh(t-\tau-1) = d_h(e_h(t-\tau)U, t-\tau-1) \quad (2.8)$$

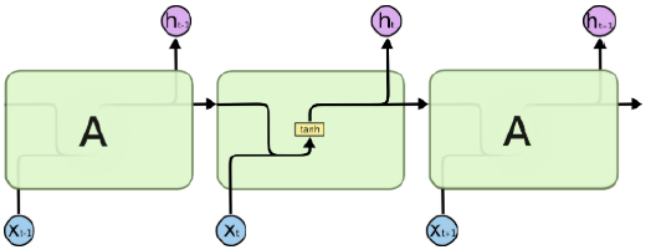
Selanjutnya bobot matriks V dan U diperbarui dengan Persamaan 2.9 dan 2.10.

$$V(t+1) = V(t) + \eta\, X\, T\, z = \sum_{z=0}^{T} x(t-z) e_h(t-z)^T \quad (2.9)$$

$$U(t+1) = U(t) + \eta\, X\, T\, z = \sum_{z=0}^{T} xs(t-z-1) e_h(t-z)^T \quad (2.10)$$

2.6 *Long Short Term Memory*

Long Short Term Memory (LSTM) adalah varian dari *Recurrent Neural Network* (RNN) yang dapat mengatasi masalah *vanishing gradient* [21]. LSTM dapat mengingat informasi jangka panjang. Pada RNN perulangan jaringan hanya menggunakan satu layer sederhana, yaitu layer *tanh* seperti pada Gambar 2.1.



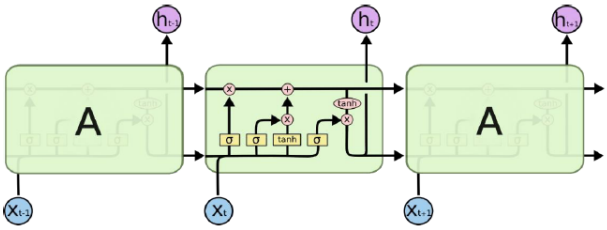
Gambar 2. 1 Layer tanh pada RNN [20].

Persamaan *tanh* diuraikan sebagai pada Persamaan 2.1 :

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.1)$$

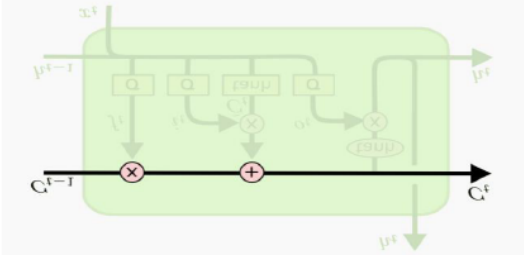
Dimana σ adalah fungsi aktivasi sigmoid dan x adalah data input.

Sedangkan, LSTM memiliki empat layer pada perulangan modelnya seperti pada Gambar 2.2.



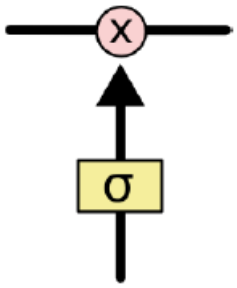
Gambar 2. 2 Perulangan dengan empat *layer* pada LSTM [20].

Kunci utama pada LSTM adalah *cell state. Cell state* adalah garis horizontal yang menghubungkan semua *output* layer pada LSTM seperti terlihat pada Gambar 2.3.



Gambar 2. 3 *Cell state* pada LSTM [20].

LSTM memiliki kemampuan untuk menambah dan menghapus informasi dari *cell state*. Kemampuan ini disebut dengan *gates*. *Gates* sebagai pengatur apakah informasi akan diteruskan atau diberhentikan. *Gates* terdiri dari *sigmoid* layer dan *pointwise multiplication operation* seperti yang terlihat pada Gambar 2.4.



Gambar 2. 4 *Sigmoid layer* pada LSTM [20].

Output dari *sigmoid* layer adalah angka 1 atau 0 yang menunjukkan apakah informasi tersebut akan diteruskan atau diberhentikan. Angka 0 menunjukkan bahwa tidak ada informasi yang akan diteruskan, sedangkan angka 1 menunjukkan bahwa semua informasi akan diteruskan. Persamaan *sigmoid* diuraikan pada Persamaan 2.2 :

$$\sigma(x) = 1/(1 + \varepsilon^{-x}) \quad (2.2)$$

Dimana X adalah data input dan ε adalah konstanta matematika (2,71828).

LSTM memiliki 3 jenis *gates* diantaranya adalah *forget gate, input gate*, dan *output gate*. *Forget gate* adalah *gate* yang memutuskan informasi mana yang akan dihapus dari *cell*. *Input gate* adalah *gate* yang memutuskan nilai dari *input* untuk diperbarui pada *state* memori. *Output gate* adalah *gate* yang memutuskan apa yang akan dihasilkan *output* sesuai dengan *input* dan memori pada *cell*. Langkah – langkah panduan jalannya metode LSTM :

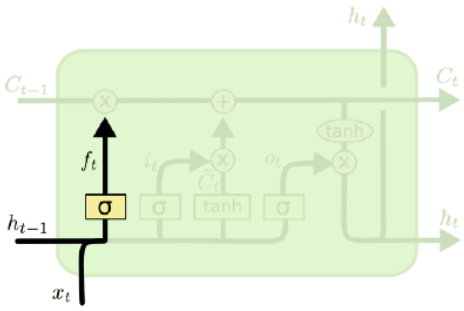
Langkah pertama adalah LSTM memutuskan informasi apa yang akan dihapus dari *cell state*. Keputusan ini dibuat oleh *sigmoid layer* yang bernama *forget gate layer*. *Forget gate layer* akan memproses h_{t-1} dan X_t sebagai *input*, dan menghasilkan *output* berupa angka 0 atau 1 pada *cell state* C_{t-1}seperti yang terlihat pada Gambar 2.5.

Persamaan *forget gate* diuraikan dalam Persamaan 2.3:

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f \quad (2.3)$$

Dimana f_t adalah *forget gate*, σ adalah fungsi *sigmoid*, W_f menunjukkan nilai *weight* untuk *forget gate*, h_{t-1} menunjukkan nilai *output* sebelum orde ke t, X_t

menunjukkan nilai *input* pada orde ke t dan b_f menunjukkan nilai bias pada *forget gate*.



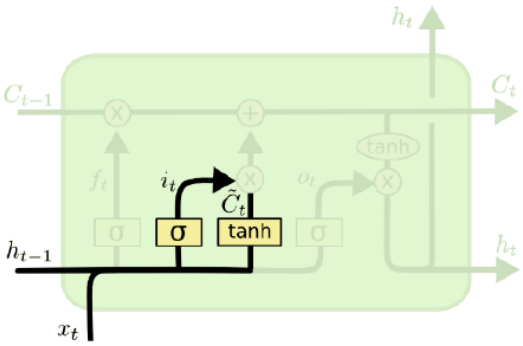
Gambar 2. 5 Langkah pertama metode LSTM "*Forget gate layer*" [20].

Nilai *weight* diuraikan pada Persamaan 2.4 :

$$W = \left(-\frac{1}{\sqrt{d}} , \frac{1}{\sqrt{d}} \right) \tag{2.4}$$

Dimana W menunjukkan *weight* dan d menunjukkan jumlah data.

Langkah kedua adalah memutuskan informasi apa yang akan disimpan di *cell state*. Untuk langkah ini terdapat dua bagian. Bagian pertama, *sigmoid* layer yang bernama *input gate layer* yang memutuskan nilai mana yang akan diperbarui. Selanjutnya, *tanh layer* membuat satu kandidat dengan nilai baru, $C^{-}t$, yang dapat ditambahkan ke *cell state*. Tahap selanjutnya adalah *output* dari *input gate layer* dan *tanh layer* akan digabungkan untuk memperbarui *cell state*. Langkah kedua digambarkan pada Gambar 2.6.



Gambar 2. 6 Langkah kedua metode LSTM "*Input gate layer & tanh layer*" [20].

Persamaan input gate diuraikan pada Persamaan 2.5:

$$i_t = \sigma(W_i . [h_{t-1} , X_t] + b_i) \tag{2.5}$$

Dimana i_t adalah *input gate*, σ adalah fungsi *sigmoid* , W_i menunjukkan nilai *weight* untuk *input gate*, h_{t-1} menunjukkan nilai *output* sebelum orde ke t, X_t menunjukkan nilai *input* pada orde ke t, b_i menunjukkan dan menunjukkan nilai bias pada *input gate.*

Persamaan kandidat baru diuraikan pada Persamaan 2.6 :

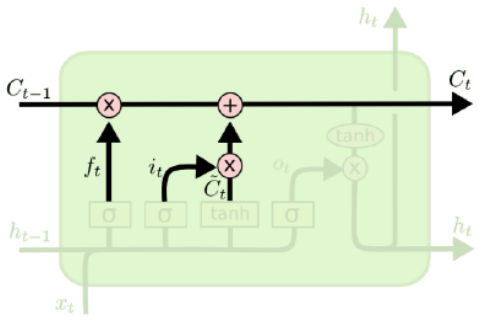
$$C_{\bar{t}} = \tanh(W_c \,.\, [\, h_{t-1}, x_t] + \, b_c) \tag{2.6}$$

Dimana $C_{\bar{t}}$ adalah nilai baru yang dapat ditambahkan ke *cell state*, $tanh$ adalah fungsi *tanh*, W_c menunjukkan nilai *weight* untuk *cell state*, h_{t-1} menunjukkan nilai *output* sebelum orde ke t, x_t menunjukkan nilai *input* pada orde ke t dan b_c menunjukkan nilai bias untuk *cell state*.

Langkah ketiga adalah memperbarui *cell state* yang lama, C_{t-1} menjadi *cell state* baru, C_t seperti digambarkan pada gambar 2.7. Dengan mengalikan *state* lama dengan f_t, untuk menghapus informasi yang sudah ditentukan sebelumnya pada langkah *forget gate* layer. Selanjutnya, ditambahkan dengan $i_t * C_{\bar{t}}$, yang merupakan nilai baru dan digunakan untuk memperbarui *state*. Persamaan *cell state* diuraikan pada Persamaan 2.7 :

$$C_t = \, f_t * \, C_{t-1} + \, i_t * \, C_{\bar{t}} \tag{2.7}$$

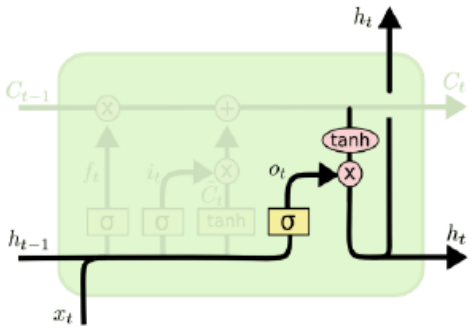
Dimana C_t adalah *Cell state*, f_t adalah *forget gate*, C_{t-1} adalah *Cell state* sebelum orde ke t, i_t adalah *input gate* dan $C_{\bar{t}}$ menunjukkan nilai baru yang dapat ditambahkan ke *cell state* pada Gambar 2.7.



Gambar 2. 7 Langkah ketiga metode LSTM "Membuat *cell state* baru" [20].

Langkah keempat adalah langkah terakhir dalam metode LSTM yang bertujuan untuk memutuskan hasil *output* digambarkan pada Gambar 2.8. *Output* harus sesuai

dengan *cell state* yang telah diproses terlebih dahulu. Pertama *sigmoid* layer memutuskan bagian dari *cell state* yang menjadi *output*. Selanjutnya, *output* dari *cell state* dimasukkan *kedalam tanh* layer (untuk mengganti nilai menjadi diantara -1 dan 1) dan dikalikan dengan *sigmoid gate*, agar *output* yang dihasilkan sesuai dengan apa yang kita putuskan sebelumnya.



Gambar 2. 8 Langkah keempat metode LSTM "menentukan *output*" [20].

Persamaan *output* gate diuraikan pada Persamaan 2.8 :

$$o_t = \sigma\,(\,W_o\,.\,[\,h_{t-1}, X_t] + \,b_o\,) \tag{2.8}$$

Dimana o_t adalah *output gate*, σ adalah fungsi sigmoid, W_o menunjukkan nilai *weight* untuk *output gate*, h_{t-1} menunjukkan nilai *output* sebelum orde ke t, X_t menunjukkan nilai *input* pada orde ke t dan b_o menunjukkan nilai bias pada *output gate*.

Persamaan nilai *output* orde t diuraikan pada Persamaan 2.9 :

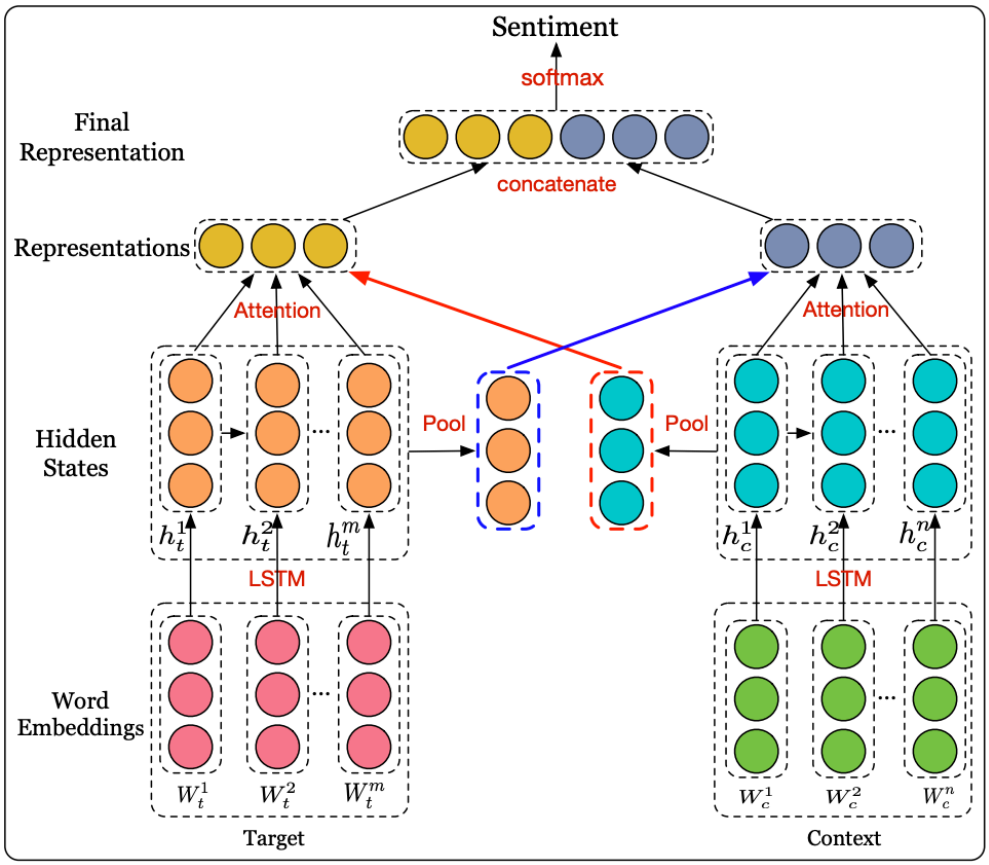
$$h_t = o_t \tanh(C_t) \tag{2.9}$$

Dimana h_t menunjukkan nilai *output* orde t, o_t adalah *output gate*, tanh adalah fungsi *tanh* dan C_t adalah *Cell state.*

2.7 Interactive Attention Network

LSTM standar tidak dapat mendeteksi bagian penting dari klasifikasi sentimen pada level target. Untuk mengasatasi masalah tersebut, Ma dkk. [9] mengusulkan model *Interactive Attention Network* (IAN) yang terdiri dari dua bagian arsitektur yang digunakan untuk memodelkan target dan konteks secara interaktif. Dengan *word*

embedding sebagai masukan, lalu LSTM untuk memperoleh *hidden states* dari tiap-tiap kata pada level kata untuk masing-masing target dan konteks. Ma dkk [9] menggunakan nilai rata-rata dari *hidden states* target dan *hidden states* konteks untuk men-s*upervise* pembangkitan dari vektor *attention* yang selanjutnya digunakan pada *attention mechanism* untuk menangkap informasi penting dalam konteks dan target. Dengan desain IAN, target dan konteks dapat mempengaruhi pembangkitan representasi dari target dan konteks secara interaktif. Pada akhirnya, representasi dari konteks dan target digabungkan sebagai representasi akhir yang diumpankan ke fungsi *softmax* untuk klasifikasi sentimen level target.



Gambar 2. 9 Arsitektur IAN.

Secara khusus, Ma dkk. [9] menformalkan notasi sebagai berikut, suatu konteks terdiri dari n-kata $[w_c^1, w_c^2, w_c^2, \ldots, w_c^n]$ dan target terdiri dari m-kata $[w_t^1, w_t^2, w_t^3, \ldots, w_t^n]$. Dimana w adalah kata spesifik. Untuk merepresentasikan sebuah kata, Ma dkk. [9] menyimpan tiap kata menjadi sebuah *vektor real-value low-dimentional*, atau yang disebut juga *word embedding* [22]. Selanjutnya

didapatkan $w^k \in R^d$ dari M^{vxd}, dimana k adalah indeks kata pada konteks atau target, d adalah dimensi *embedding* dan v adalah ukuran dari *vocabulary*.

Selanjutnya, Ma dkk. [9] menggunakan LSTM *network* untuk melakukan *learning* dari *hidden word semantic*, karena kata-kata dalam kalimat memiliki ketergantungan yang kuat satu sama lain, dan LSTM cukup bagus untuk melakukan *learning* pada *long-term depedencies* dan bisa menghindari *vanishing gradient problem* dan *expansion problem* [9]. Secara formal, nilai masukan *word embedding* w^k, *previous cell state* c^{k-1} dan *previous hidden state* h^{k-1} , *current cell state* c^k dan *current hidden state* h^k di jaringan LSTM diperbarui dengan Persamaan 2.10, 2.11, 2.12, 2.13, 2.14 dan 2.15

$$i^k = \sigma(W_i^w \,.\, w^k + W_i^h \,.\, h^{k-1} + b_i) \tag{2.10}$$

$$f^k = \sigma(W_f^w \,.\, w^k + W_f^h \,.\, h^{k-1} + b_f) \tag{2.11}$$

$$o^k = \sigma(W_o^w \,.\, w^k + W_o^h \,.\, h^{k-1} + b_o) \tag{2.12}$$

$$\hat{c}^k = tanh(W_c^w \,.\, w^k + W_c^h \,.\, h^{k-1} + b_c) \tag{2.13}$$

$$c^k = f^k \odot c^{k-1} + i^k \odot \hat{c}^k \tag{2.14}$$

$$h^k = o^k \odot tanh(c^k) \tag{2.15}$$

Dimana secara berturut-turut **i**, **f** dan **o** adalah *input gate*, *forget gate* dan *output gate* di LSTM. σ adalah fungsi *sigmoid*, W adalah matriks bobot dan b adalah bias. Simbol $\cdot$ adalah perkalian matriks dan $\odot$ adalah perkalian *element-wise*. Selanjutnya, didapatkan *hidden states* $[h_c^1, h_c^2, h_c^3, \ldots, h_c^n]$ sebagai representasi final dari konteks. Untuk mendapatkan model target yang lebih baik, Ma dkk. [9] juga menggunakan LSTM untuk memperoleh hidden states dari target $[h_t^1, h_t^2, h_t^3, \ldots, h_t^n]$.

Selanjutnya, didapatkan representasi awal dari target dan konteks yaitu c_{avg} dan t_{avg} dengan menghitung rata-rata dari hidden state yang ditunjukan dengan Persamaan 2.16 dan 2.17.

$$c_{avg} = \sum_{i=1}^{n} h_c^i / n \tag{2.16}$$

$$t_{avg} = \sum_{i=1}^{n} h_t^i / m \tag{2.17}$$

Dengan representasi awal dari target dan konteks sebagai input, Ma dkk. [9] mengadopsi *attention mechanism* untuk memilih informasi penting yang berkontribusi untuk menetapkan polaritas sentimen. Seperti yang dinyatakan pada Gambar 2.9, Ma dkk. [9] mempertimbangkan pengaruh pada konteks dari target dan pengaruh pada target dari konteks, yang dapat memberikan lebih banyak petunjuk untuk memperhatikan fitur sentimen terkait.

Ma dkk. [9] mengambil sepasang konteks dan target untuk menggambarkan proses attention, seperti yang ditunjukkan pada Gambar 2.9. Dengan representasi konteks $[h_c^1, h_c^2, h_c^3, \ldots, h_c^n]$, *attention mechanism* akan membangkitkan vektor attention α_i menggunakan representasi target t_{avg} untuk konteks yang ditujunjukan pada Persamaan 2.18. Dimana γ adalah fungsi *score* yang mengalkulasi representasi konteks h_c^i yang penting.

$$\alpha_i = \frac{exp(\gamma(h_c^i, t_{avg}))}{\sum_{j=1}^{n} exp(\gamma(h_c^j, t_{avg}))} \tag{2.18}$$

Fungsi *score* γ didefinisikan dengan Persamaan 2.19. Dimana W_a adalah matriks bobot, b_a adalah bias, $tanh$ adalah fungsi non-linear dan ${t_{avg}}^T$ adalah transpos dari t_{avg}.

$$\gamma(h_c^i, t_{avg}) = tanh(h_c^i \cdot W_a \cdot {t_{avg}}^T + b_a) \tag{2.19}$$

Demikian pula, untuk target, Ma dkk. [9] mengalkulasi vektor attention β_i menggunakan representasi konteks C_{avg} dengan Persamaan 2.20. Dimana γ adalah fungsi *score* yang ditunjukan pada Persamaan 2.19.

$$\beta_i = \frac{exp(\gamma(h_t^i, C_{avg}))}{\sum_{j=1}^{n} exp(\gamma(h_t^j, C_{avg}))} \tag{2.20}$$

Setelah menghitung bobot dari *attention word*, untuk mendapatkan representasi target t_r dan konteks c_r berdasarkan vektor attention α_i dan β_i dilakukan perhitungan yang ditunjukan Persamaan 2.21 dan 2.22.

$$c_r = \sum_{i=1}^{n} \alpha_i h_c^i \tag{2.21}$$

$$t_r = \sum_{i=1}^{n} \beta_i h_t^i \tag{2.22}$$

Tahap akhir adalah menggabungkan representasi target t_r dan representasi konteks c_r sebagai vektor **d** untuk *classifer*. Disini, Ma dkk. [9] menggunakan *non-linear layer* untuk memproyeksikan **d** ke ruang C yang ditargetkan dengan perhitungan yang ditujukan pada Persamaan 2.23.

$$x = tanh(W_l \cdot d + b_l) \tag{2.23}$$

Dimana W_l adalah bobot matriks dan b_l adalah bias. Probabilitas dari pelabelan dokumen dengan polaritas sentimen $i(i \in [1, C])$ dihitung dengan Persamaan 2.24. Label dengan probabilitas tertinggi akan dipilih menjadi hasil akhir.

$$y_i = \frac{exp(x_i)}{\sum_{i=1}^{C} exp(x_i)} \tag{2.24}$$

2.8 Penelitian Terkait

Penelitian tentang klasifikasi sentimen berbasis target telah banyak dilakukan, seperti yang dilakukan oleh Dong dkk. [10] yang mengusulkan *Adaptive Recursive Neural Network* (AdaRNN) untuk dataset *twitter*. AdaRNN secara adaptif mempropagasi sentimen pada tiap kata untuk ditargetkan tergantung pada konteks dan hubungan sintaksisnya. Dong dkk. [10] memodelkan propagasi sentimen yang adaptif sebagai distribusi atas fungsi komposisi. Studi eksperimental yang dilakukan Dong dkk. [10] menggambarkan bahwa AdaRNN meningkatkan motode *baseline*. Dong dkk. [10] mengonversi *dependency tree* untuk suatu target, selanjutnya AdaRNN belajar bagaimana mempropagasi sentimen kata ke *node* target secara adaptif. AdaRNN memungkinkan propagasi sentimen menjadi peka terhadap kategori linguistik dan sematik dengan menggunakan komposisi yang berbeda. Hasil dari percobaan Dong dkk. [10] mendapatkan akurasi sebesar 66.3 dan *macro f-1* sebesar 65.9.

Pada peneltian yang dilakukan Tang dkk. [11], mengadopsi dua LSTM untuk memodelkan konteks sebelah kiri target dan konteks sebelah kanan target secara berturut-turut.Representasi dari bagian sebelah kiri dan kanan target digabungkan untuk memprediksi polaritas sentimen dari target. Tang dkk. [11] melakukan percobaan dengan dataset *twitter* dan mendapatkan nilai akurasi sebesar 70.2 dan *macro f-1* sebesar 69.0.

Pada penelitian yang dilakukan Peng dkk. [23], mengusulkan kerangka kerja baru berdasarkan jaringan saraf tiruan untuk mengidentifikasi sentimen target opini didalam komentar atau ulasan. Kerangka kerja Peng dkk. [23] mengadopsi mekanisme *multi-attention* untuk menangkap fitur sentimen yang dipisahkan oleh jarak yang jauh, sehingga lebih kuat terhadap informasi yang tidak relevan. Hasil dari multiple attentions adalah *non-linear* yang dikombinasikan dengan RNN yang memperkuat kekuatan ekspresif dari model untuk menangani lebih banyak komplikasi. Pada dataset *twitter* penelitian yang dilakukan Peng dkk. [23] mendapatkan nilai akurasi sebesar 69.3 dan *macro f-1* sebesar 67.3.

Pada penelitian Liu dkk. [24], menemukan bahwa klasifikasi sentiment berbasis aspek sebagian besar hanya fokus pada mengindentifikasi kata-kata sentimen tanpa mempertimbangkan relevansi kata-kata tersebut sehubungan dengan aspek yang diberikan dalam kalimat. Oleh karena itu, biasanya tidak cukup untuk berurusan dengan kalimat multi-aspek dan struktur kalimat yang rumit secara sintaksis. Untuk mengatasi masalah tersebut Liu dkk. [24], mengusulkan model klasifikasi sentimen berbasis *content attention*, dengan dua *attention mechanism*: *sentence-level content attention mechanism* yang mampu menangkap informasi penting tentang aspek yang diberikan dari perspektif global, dan *context attention mechanism* yang bertanggung jawab secara simultan untuk memperhitungkan urutan kata-kata dan korelasinya, dengan menanamkan ke dalam serangkaian *memory* yang disesuiakan. Pada dataset *twitter* penelitian ini mendapatkan nilai akurasi sebesar 71.53.

Berdasarkan penelitian pada dataset *twitter* yang telah dilakukan dengan berbagai macam metode, pada penelitian ini dilakukan eksperimen dengan metode *Interactive Attention Network* (IAN) dalam masalah klasifikasi sentimen level target pada dataset *twitter.*

Rangkuman penelitian terdahulu yang terkait dengan dataset *twitter* dengan beragam metode dapat dilihat dalam Tabel 2.1.

Tabel 2. 1 Penelitian Terkait

Peneliti	Tahun	Judul	Hasil
Dong dkk. [10]	2014	Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification	Akurasi yang didapat pada daset *twitter* sebesar 66.3 dan *macro f-1* sebesar 65.9.
Tang dkk. [11]	2016	Effective LSTMs for Target-Dependent Sentiment Classification	Percobaan dengan dataset *twitter* mendapatkan nilai akurasi sebesar 70.2 dan *macro f-1* sebesar 69.0.
Peng dkk. [23]	2017	Recurrent Attention Network on Memory for Aspect Sentiment Analysis	Percobaan dengan dataset *twitter* mendapatkan nilai akurasi sebesar 69.3 dan *macro f-1* sebesar 67.3.
Liu dkk. [24]	2018	Content attention model for aspect based sentiment analysis	Pada dataset *twitter* penelitian ini mendapatkan nilai akurasi sebesar 71.53
Muhamad Enrinal Zulhimar	2020	Interactive Attention Network Pada Dataset Twitter	Percobaan ini mendapat nilai akurasi sebesar 70.80 dan *macro f-1* sebesar 68.60.