BAB 2
TINJAUAN PUSTAKA

2.1. *Text Mining*

Text mining adalah proses penemuan pola yang sebelumnya tidak terlihat pada dokumen atau sumber tertentu menjadi pola yang diinginkan untuk tujuan tertentu. Text mining sering digunakan untuk analisis imformasi dengan cara mencari kata-kata yang dapat mewakili isi dari dokomen [9].

2.2. Ekstraksi Informasi

Ekstraksi Informasi merujuk pada ekstraksi otomatis dari informasi terstruktur seperti entitas, hubungan antar entitas, dan atribut deskripsi entitas dari sumber yang tidak terstruktur [10]. Dalam sebagian besar kasus, kegiatan ini berkaitan dengan pemrosesan teks bahasa manusia melalui pemrosesan bahasa alami (NLP). Kegiatan terbaru dalam pemrosesan dokumen multimedia seperti anotasi otomatis dan ekstraksi konten dari gambar / audio / video / dokumen dapat dilihat sebagai ekstraksi informasi. Dalam penelitian ini akan melakukan ekstraksi informasi pada dokumen Surat Keputusan.

2.3. *Preprocessing*

Preprocessing adalah proses pengolahan data asli yang sebelumnya tidak terstruktur menjadi terstruktur [11]. Proses ini bertujuan agar data yang akan digunakan dalam proses ekstraksi informasi bersih dari *noise*. Oleh karena itu, dibutuhkan proses yang dapat mengubah data yang sebelumnya tidak terstruktur menjadi data yang terstruktur untuk diproses ke tahap selanjutnya.

2.3.1. *Case Folding*

Case folding merupakan proses penyeragaman bentuk huruf untuk semua teks pada dokumen baik itu menjadi huruf kecil (*lowercase*) atau huruf kapital (*uppercase*). Pada penelitian ini bentuk huruf memiliki bentuk yang beragam sehingga *case folding* dilakukan untuk menghilangkan *noise*.

2.3.2. *Filtering*

Filtering merupakan proses membersihkan simbol-simbol yang tidak diperlukan dalam dokumen seperti tanda baca, angka, dan *emoticon.* Pada penelitian ini *emoticon* seperti :D, :), :(, ^^, :3 akan dihilangkan. Selain itu tanda baca seperti titik (.), koma (,) dan tanda baca lainnya akan dihilangkan

2.3.3. *Tokenizing*

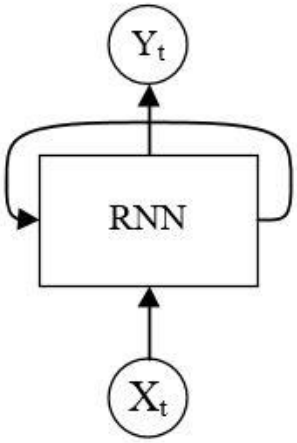
Tokenizing merupakan proses pemisahan setiap kata dalam sebuah kalimat. Proses pemisahan kata dilakukan berdasarkan spasi diantara setiap kata dalam kalimat. Hasil dari proses *tokenization* akan digunakan sebagai masukan dalam tahap transformasi teks ke dalam bentuk angka.

2.3.4. *Labelling*

Tagging dalam teks *pre-processing* disini bisa disebut sebagai aplikasi kondisional yang digunakan untuk memberikan tag kondisional barang atau kata atau frasa tertentu dalam sebuah kalimat tergantung dari kata kunci yang terkandung dalam kalimat yang diidentifikasi oleh *sed command*.

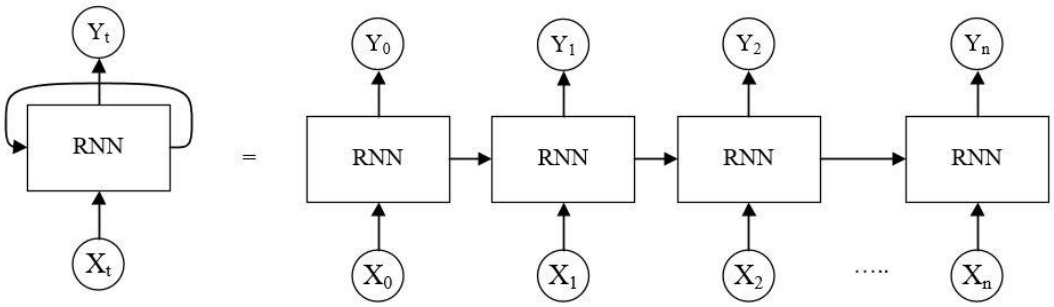
2.4. *Recurrent Neural Network*

Recurrent Neural Network (RNN) adalah jaringan dengan *loop,* yang memungkinkan informasi bertahan dalam jaringan, RNN memiliki koneksi umpan balik ke jaringan itu sendiri yang memungkinkan aktivasi mengalir kembali dalam satu lingkaran mempelajari urutan dan informasi sebelumnya [12]. RNN sangat *powerful* dalam memodelkan data sekuensial, ucapan atau teks dan diterapkan pada data non-sekuensial untuk melatih secara non-sekuensial, RNN bisa digunakan dalam kasus *image, video captioning, word prediction, word translation, image processing, speech recognition, natural language processing, music processing* dan sebagainya. Arsitektur *Recurrent Neural Network* dapat dilihat pada Gambar 2.1



Gambar 2. 1 *Recurrent Neural Network*

Dimana Xt merupakan inputan terhadap t (waktu/urutan inputan berdasarkan waktu/inputan data ke-t) Yt merupakan hasil output. Dari gambar diatas bahwa proses yang dilakukan pada RNN itu secara berulang-ulang sehingga data inputan sebelumnya tersimpan ke memori RNN. *Unfolding Recurrent Neural Network* dapat dilihat pada Gambar 2.2 *Unfolding Recurrent Neural Network*.



Gambar 2. 2 *Unfolding Recurrent Neural Network*

RNN merupakan bagian dari *neural network* sehingga lapisan-lapisan dari RNN dibagi menjadi 3 bagian yaitu :

1. *Input Layer*

   Merupakan lapisan yang menerima input kemudian meneruskan ke neuron lain dalam jaringan.

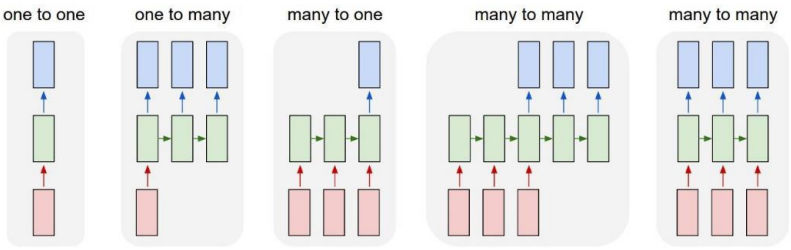
2. *Hidden Layer*

   Merupakan lapisan yang tersembunyi berfungsi untuk meningkatkan kemampuan jaringan dalam memecahkan masalah.

3. Output Layer

Merupakan lapisan yang meghasilkan output dari hasil pemrosesan.

Pada Gambar 2.2 dapat dilihat bahwa RNN akan memproses data input satu persatu secara sekuensial, hidden layer pun akan melempar data menuju ke hidden layer lainnya pada skala waktu selanjutnya. Begitu seterusnya secara sekuensial. Dibawah ini merupakan macam-macam pemrosesan pada RNN ditujukan pada Gambar 2.3 Pemrosesan RNN.



Gambar 2.3 Pemrosesan RNN

Berikut adalah penjelasan dari Gambar 2.3 sebagai berikut :

1. Satu ke Satu

   Pada jenis ini, model memproses satu input dan output yang dihasilkan juga satu.

2. Satu ke banyak

   Pada jenis pemrosesan ini, inputannya hanya satu namun dapat menghasilkan banyak output.

3. Banyak ke Satu

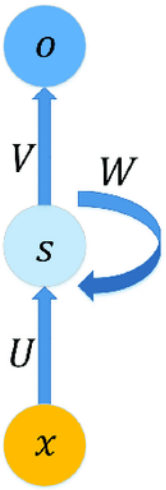
   Pada jenis pemrosesan ini, model memproses banyak input namun output yang dihasilkan hanya satu.

4. Banyak ke banyak

   Pada jenis ini, input dan ouputnya adalah data sekuensial yang jumlahnya banyak. pada jenis ini output yang dihasilkan tidak harus sama dengan jumlah input, dan model ini akan menghasilkan output hanya setelah semua input selesai diproses.

2.4.1. *Elman Recurrent Neural Network*

Elman Recurrent Neural Netwrok (ERNN) merupakan salah satu dari arsitektur *Recurrent Neural Network* (RNN), *Elman Recurrent Neural Network* (ERNN) termasuk ke dalam *Simple Recurrent Neural Network* (SRNN) [12]. ERNN melakukan proses *learning* dengan membuat salinan neuron layer hidden pada layer input disebut sebagai *context input (hidden layer sebelumnya)*, sehingga salinan ini berfungsi sebagai perpanjangan dari input layer. Fungsi dari *context input* ini adalah untuk menyimpan status ataupun keadaaan sebelumnya dari layer hidden, untuk kemudian disampaikan kembali kepada layer hidden. Hubungan antara *context input* dan layer hidden adalah terhubung penuh (*fully connected*) dan diberi bobot 1. Selain ERNN arsitektur yang termasuk *Simple Recurrent Neural Network* adalah *Jordan Recurrent Neural Network*, perbedaannya adalah model Jordan melakukan proses learning dengan membuat salinan layer output pada layer input yang disebut sebagai *state layer*, dengan proses ini, hasil output pada iterasi sebelumnya akan menjadi bagian dari input pada iterasi selanjutnya.



Gambar 2. 4 *Elman Recurrent Neural Network* [13]

Pada Gambar 2.4 tedapat parameter 3 parameter bobot yaitu U, V, dan W. Dimana U adalah bobot yang ada bobot antara *input layer* dan *hidden layer*, V adalah bobot antara *hidden layer* dan *output layer*, dan W adalah input antara *hidden* dan *context layer*.

Untuk proses perhitungan Elman dimulai dari proses perhitungan hidden state hingga output. Adapun rumus untuk menghitung hidden state dapat dilihat pada persamaan (2.1).

$$\boldsymbol{s_t = tanh(W.s_{(t-1)} + U.X_t)} \quad (2.1)$$

Dimana :

S_t : hidden state pada time step t

S_{t-1} : satu hidden state sebelumnya dari time step t

X_t : vektor input

W : bobot yang berada diantara setiap hidden state.

U : bobot yang berada diantara inpur dan hidden state

Kemudian untuk perhitungan nilai *output* dilakukan dengan menggunakan Persamaan (2.2).

$$\boldsymbol{o_t = softmax(V.s_t)} \quad (2.2)$$

Dimana :

O_t : *output* pada *time step t*

V : bobot yang berada diantara *hidden state* dan *output*.

S_t : *hidden state time step t*.

2.4.2. Inisialisasi Bobot

Pada ERNN terdapat 3 bobot: yaitu U, V, dan W. Aturan dimensi dari ketiga bobot tersebut dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Aturan Inisialisasi Bobot Awal

Bobot	Keterangan
$\boldsymbol{U \in \mathbb{R}^{HxI}}$	Adalah matriks berukuran $\boldsymbol{HxI}$ dimana $\boldsymbol{H}$ adalah dimensi hidden *(Hidden_dim)* yang ditentukan manual dan $\boldsymbol{I}$ adalah dimensi vektor data masukan *(Input_dim)*.
$\boldsymbol{V \in \mathbb{R}^{OxH}}$	Adalah matriks berukuran $\boldsymbol{OxH}$ dimana $\boldsymbol{O}$ adalah dimensi vektor label *(Output_dim)* dan $\boldsymbol{H}$ adalah

	dimensi hidden *(Hidden_dim)* yang ditentukan manual.
$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{HxH}}$	Adalah matriks berukuran $\boldsymbol{HxH}$ dimana $\boldsymbol{H}$ adalah dimensi hidden *(Hidden_dim)* yang ditentukan manual.

Berikut adalah teknik inisialisasi yang digunakan dalam penelitian ini [18].

1) Elemen-elemen *U* merupakan nilai acak yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan ini,

$$\boldsymbol{BA}(\boldsymbol{U}) = -\frac{\mathbf{1}}{\sqrt{\boldsymbol{Input\_dim}}} \text{ dan } \boldsymbol{BB}(\boldsymbol{U}) = \frac{\mathbf{1}}{\sqrt{\boldsymbol{Input\_dim}}} \quad (2.3)$$

2) Elemen-elemen $\boldsymbol{V}$ merupakan nilai acak yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan ini,

$$\boldsymbol{BA}(\boldsymbol{V}) = -\frac{\mathbf{1}}{\sqrt{\boldsymbol{Hidden\_dim}}} \text{ dan } \boldsymbol{BB}(\boldsymbol{V}) = \frac{\mathbf{1}}{\sqrt{\boldsymbol{Hidden\_dim}}} \quad (2.4)$$

3) Elemen-elemen $\boldsymbol{W}$ merupakan nilai acak yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan ini,

$$\boldsymbol{BA}(\boldsymbol{W}) = -\frac{\mathbf{1}}{\sqrt{\boldsymbol{Hidden\_dim}}} \text{ dan } \boldsymbol{BB}(\boldsymbol{W}) = \frac{\mathbf{1}}{\sqrt{\boldsymbol{Hidden\_dim}}} \quad (2.5)$$

Keterangan, dimana *Input_dim* adalah ukuran dari dimensi Input dan *Hidden_dim* adalah ukuran dari dimensi Hidden.

2.4.3. *Activation Function* (Fungsi Aktivasi)

Fungsi aktivasi sangat umum digunakan dalam *neural network*. Alasan utama menggunakan fungsi aktivasi adalah agar *neural network* mengenali data yang non-linear, karena output yang dihasilkan dari *neural network* jarang sekali bersifat linear [14]. Fungsi aktivasi juga digunakan untuk mengaktifkan dan menonaktifkan *neuron*, karakteristik *neural network* ditentukan oleh bobot dan input-output fungsi aktivasi yang diterapkan. Terdapat banyak jenis fungsi aktivasi pada *neural network*, namun yang di gunakan dalam penelitian ini fungsi aktivasi *hyperbolic tangent* dan *softmax*.

2.4.3.1. *Tanh* (*Hyporbolic Tangent*)

Fungsi aktivasi *hyperbolic tangent* adalah tipe aktivasi fungsi dalam *deep learning* dan memiliki beberapa varian, fungsi aktivasi *hyperbolic tangent* dikenal juga sebagai fungsi *tanh* yang menghasilkan nilai -1 sampai 1 [14]. Fungsi *tanh* memberikan kinerja pelatihan yang lebih baik untuk *multi layer neural network* dibandingkan fungsi *sigmoid*. Fungsi *tanh* telah banyak digunakan dalam *recurrent neural network* untuk kasus *natural language processing* dan *speech recognition*. Persamaan dari fungsi tanh dapat dilihat pada persamaan (2.6).

$$\boldsymbol{tanh}(\boldsymbol{x}) = \frac{\mathbf{sin}(\boldsymbol{x})}{\mathbf{cos}(\boldsymbol{x})} = \left(\frac{\boldsymbol{e^{x} - e^{-x}}}{\boldsymbol{e^{x} + e^{-x}}}\right) = \left(\frac{\boldsymbol{e^{2x} - 1}}{\boldsymbol{e^{2x} + 1}}\right) \tag{2.6}$$

2.4.3.2. *Sofmax*

Fungsi aktivasi softmax digunakan untuk menghitung probabilitas pada hasil output, yang terjadi di output layer dimana akan diambil nilai probabilitas paling besar sebagai prediksi. Softmax digunakan untuk menghitung probabilitas distribusi dari vektor bilangan real. Fungsi softmax menghasilkan output yang merupakan kisaran nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1 [14]. Persamaan dari fungsi sofmax dapat dilihat pada persamaan (2.7).

$$\boldsymbol{softmax}(\boldsymbol{xi}) = \frac{\mathbf{exp}(\boldsymbol{x_i})}{\sum_j \mathbf{exp}(\boldsymbol{x_j})} \tag{2.7}$$

2.5. *Loss Function*

Loss function bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai yang sebenarnya [14]. Ada beberapa jenis loss function diantaranya *Mean Square Error* (MSE) dan *Cross Entropy* (CE). Namun dalam praktiknya, CE lebih cepat dalam konvergensi dan mendapatkan hasil lebih baik dalam tingkat kesalahan klasifikasi [14]. *Cross entropy* biasa digunakan pada *neural network* dan dalam kasus *multi class classification*. Pada penelitian ini *loss function* yang digunakan untuk menghitung nilai error hasil prediksi menggunakan *Cross Entropy* (CE). Persamaan *Cross Entropy* dapat dilihat pada persamaan (2.8).

$$\boldsymbol{error} = \sum_i \boldsymbol{E_t} \tag{2.8}$$

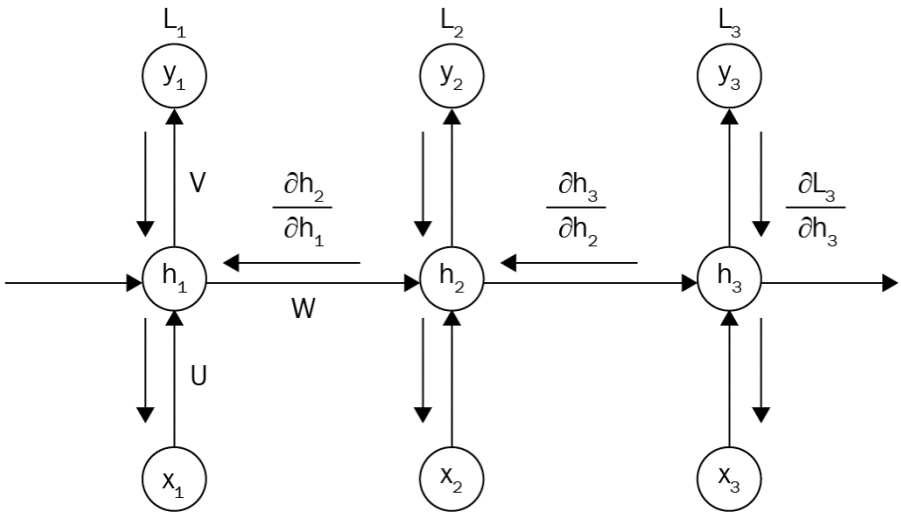
Dimana $\boldsymbol{E_t}$ dapat dilihat pada Persamaan (2.9).

$$\boldsymbol{E_t} = -\sum_i \boldsymbol{y_t} \cdot \mathbf{log}\, \boldsymbol{o_t} \tag{2.9}$$

Pada Persamaan (2.9) terdapat beberapa variabel yaitu, $\boldsymbol{y_t}$ adalah label sebenarnya pada *timestep* ke-t, $\boldsymbol{o_t}$ adalah label prediksi pada timestep ke-t, $\boldsymbol{y_t}$ adalah Vector $\boldsymbol{y_t}$ (label sebenarnya), $\boldsymbol{o_t}$ adalah Vector $\boldsymbol{o_t}$ (*output state* atau dugaan sistem), dan $\boldsymbol{E_t}$ adalah error dari output $\boldsymbol{o_t}$.

2.6. *Backpropagation Through Time* (BPTT)

Backpropagation through time merupakan algoritma yang biasa digunakan untuk memperbaharui bobot pada *Recurrent Neural Network*. Pada *Recurrent Neural Network* nilai error dapat di *backpropagate* lebih jauh daripada neural network biasa.



Gambar 2. 5 ***Backpropagation Through Time***

Prinsip dasar dari BPTT adalah *unfolding* [15]. Secara konseptual, BPTT bekerja dengan membuka gulungan (unfolding) setiap input layer pada setiap timestep (t), kemudian nilai error dihitung dan diakumulasikan untuk setiap timetep (t). Jaringan kemudian diputar kembali untuk memperbaharui bobot. Pada pelatihan RNN ada 3 bobot yang akan diperbaharui dimana U, bobot dari input layer ke hidden layer, W, bobot dari hidden layer sebelumnya ke hidden layer selanjutnya, dan V, bobot dari hidden layer ke output layer.

Pada ERNN dalam melakukan BPTT akan menghitung turunan bobot U, W, dan V. Adapun rumus penurunan dari setiap bobot adalah sebagai berikut.

1. Penurunan Bobot U

   Rumus turunan bobot *W* dapat dilihat pada Persamaan (2.10).

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \tag{2.10}$$

Keterangan, dimana $\frac{\partial E_t}{\partial U}$ adalah turunan dari nilai error ke-t terhadap bobot U, $\boldsymbol{E_t}$ adalah nilai error ke-t, $\boldsymbol{U}$ adalah bobot dari input layer ke hidden layer, $\boldsymbol{X_t}$ adalah input ke-t, dan t adalah timestep atau urutan kata. Persamaan (2.10) dapat diuraikan menjadi Persamaan (2.11).

$$\frac{\partial E_t}{\partial U} = \left(\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial U}\right)\left(\frac{\partial E_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial U}\right)\left(\frac{\partial E_t}{\partial s_{t-2}}\frac{\partial s_{t-2}}{\partial U}\right)\ldots\left(\frac{\partial E_t}{\partial s_1}\frac{\partial s_1}{\partial U}\right) \tag{2.11}$$

Persamaan (2.11) dapat diuraikan menjadi Persaman (2.12).

$$\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial U} = \boldsymbol{\delta_t} \otimes \boldsymbol{X_t} \tag{2.12}$$

Pencarian nilai $\boldsymbol{\delta_t}$ dapat dilihat pada Persamaan (2.13).

$$\boldsymbol{\delta_t} = [\boldsymbol{V^T} \cdot (\boldsymbol{o_t} - \boldsymbol{y_t})] \odot (\mathbf{1} - \boldsymbol{s_t}^2) \tag{2.13}$$

Keterangan dari Persamaan (2.13), dimana V adalah bobot dari hidden layer ke output layer, $\boldsymbol{o_t}$ adalah output state ke-t, $\boldsymbol{y_t}$ adalah label sebenarnya ke-t, t adalah timestep atau urutan kata, dan *T* adalah simbol Transpose.

2. Penurunan Bobot V

   Rumus menghitung turunan bobot V dapat dilihat pada Persamaan (2.14).

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \tag{2.14}$$

   Persamaan (2.14) dapat diuraikan menjadi Persamaan (2.15).

$$\frac{\partial E_t}{\partial V} = \sum_t (\boldsymbol{o_t} - \boldsymbol{y_t}) \otimes \boldsymbol{s_t} \tag{2.15}$$

3. Penurunan Bobot W

   Bobot dari input dalam menghitung turunan bobot W dimana setiap hidden state (s_t) terhubung dengan hidden state sebelumnya (s_{t-1}) dan begitu seterusnya hingga mencapai hidden state pertama s_1 yang terhubung dengan s_0. Maka dalam penurunan bobot W harus memperhatikan *hidden state*. Rumus turunan bobot W ditunjukan oleh Persamaan (2.16).

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{2.16}$$

Persamaan (2.16) dapat diuraikan menjadi Persamaan (2.17).

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial W}\right) + \left(\frac{\partial E_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial W}\right) + \left(\frac{\partial E_t}{\partial s_{t-2}}\frac{\partial s_{t-2}}{\partial W}\right) + \cdots \left(\frac{\partial E_t}{\partial s_1}\frac{\partial s_1}{\partial W}\right) \quad (2.17)$$

Persamaan (2.17) dapat diuraikan menjadi Persaman (2.18).

$$\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial W} = \boldsymbol{\delta_t} \otimes \boldsymbol{s_{t-1}} \quad (2.18)$$

Pada Persamaan (2.18) terdapat beberapa variabel yaitu, $\frac{\partial E}{\partial W}$ adalah Turunan dari nilai error ke-t terhadap bobot W, E_t Nilai error ke-t, W bobot dari hidden layer sebelumnya ke hidden layer sesudahnya, s_t adalah *hidden state* ke-t, t adalah *timestep* atau urutan kata, dan $\boldsymbol{\delta_t}$ didapat dari penurunan. Penuruan $\boldsymbol{\delta_t}$ menggunakan Persamaan (2.13).

2.7. *Stochastic Gradient Descent* (SGD)

Stochastic Gradient Descent adalah salah satu metode dalam dengan meminimalkan nilai error untuk mengukur keakuratan jaringan dalam (2.15) melakukan tugas yang diberikan [16]. SGD bertujuan untuk mengatasi nilai error (loss) yang tinggi dengan cara memperbarui parameter. Disebut "stochastic" karena pada prosesnya sampel dipilih secara acak sebagai kelompok tunggal (seperti dalam penurunan gradien standar) atau dalam urutan, sampel yang dipilih diambil dari dataset pelatihan. Adapun rumus dari SGD ditunjukan persamaan (2.15).

$$\boldsymbol{\theta_{(baru)}} = \boldsymbol{\theta_{(lama)}} - \boldsymbol{\alpha} \times \frac{\boldsymbol{\partial E}}{\boldsymbol{\partial \theta}}$$

Keterangan :

$\boldsymbol{\theta_{(baru)}}$ = Nilai bobot yang baru

$\boldsymbol{\theta_{(lama)}}$ = Nilai bobot yang lama

$\boldsymbol{\alpha}$ = *learning rate*

$\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial \theta}}$ = Nilai turunan antara nilai error dan bobot

2.8. Akurasi

Pada penelitian ini menggunakan perhitungan akurasi untuk mengetahui tingkat akurasi dari klasifikasi ERNN. Untuk menghitung akurasi dari ERNN dilakukan dengan perhitungan rumus berikut.

$$\boldsymbol{Akurasi}\ (\%) = \frac{\boldsymbol{jumlah\ yang\ diklasifikasi\ dengan\ benar}}{\boldsymbol{jumlah\ sampel\ data\ uji}}\ \boldsymbol{x}\ \mathbf{100}\% \qquad (2.16)$$

2.9. *Black Box Testing*

Menurut Pressman [17] *black-box testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program. *Black-Box testing* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau funsgi yang hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam stuktur data atau akses database eksternal
4. Kesalahan perilaku atau kesalahan kinerja
5. Inisiallisasi dan pemurusan kesalahan.

2.10. Pemodelan Sistem

Permodelan sistem merupakan metode yang memungkinkan untuk memodelkan sistem pada aplikasi yang akan dibangun. Adapun permodelan yang digunakan pada penelitian tugas akhir ini.

2.10.1. Diagram Konteks

Menurut Jogiyanto [18] diagram konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara entity luar, masukan dan keluaran dari sistem. Diagram konteks dipresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [18].

2.10.2. *Data Flow Diagram* (DFD)

Data Flow Diagram adalah suatu model logika atau proses yang dibuat untuk menggambarkan dari amana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut [19].

2.11. Bahasa Pemrograman

Bahasa pemrograman atau sering diistilahkan juga dengan bahasa komputer, adalah instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer.

2.11.1. *Python*

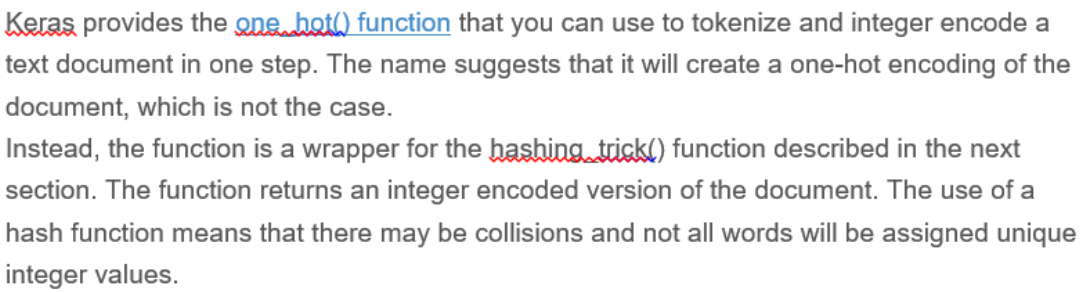
Python adalah bahasa pemrograman yang bersifat *open source*. Bahasa pemrograman ini dioptimalisasikan untuk *software quality*, *developer productivity*, *program portability*, dan *component integration*. Python telah digunakan untuk mengembangkan berbagai macam perangkat lunak, seperti *internet scripting*, *systems programming*, *user interfaces*, *product customization*, *numberic programming*, dan lain-lain [20].

2.12. *Tesseract*

Tesseract merupakan *free engine Optical Character Recognition* (OCR) yang dirilis dibawah lisensi *Apache* dan pengembangannya disponsori oleh *Google*. *Tesseract* saat ini merupakan salah satu *engine* OCR *open source* yang paling akurat dibanding dengan *engine* yang lain. *Tesseract* dapat membaca berbagai format gambar dan mengkonversinya ke teks. Selain gambar, *Tesseract* juga dapat membaca file PDF

2.13. *One Hot* pada *Library Keras*

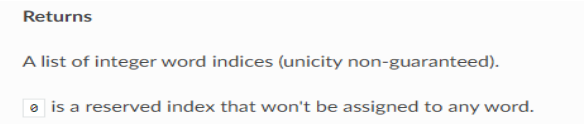
Metode word embedding yang digunakan pada penelitian ini adalah *one hot* menggunakan *library keras* [20]. Kutipan sequence pada library keras dapat dilihat pada Gambar 2.6 dibawah ini :


Keras provides the one_hot() function that you can use to tokenize and integer encode a text document in one step. The name suggests that it will create a one-hot encoding of the document, which is not the case.

Instead, the function is a wrapper for the hashing_trick() function described in the next section. The function returns an integer encoded version of the document. The use of a hash function means that there may be collisions and not all words will be assigned unique integer values.

Gambar 2. 6 *Sequence From Keras Library*

Gambar 2.7 menjelaskan bahwa merubah kata menjadi vector dalam bentuk *sequence* integer yang mana tidak bisa dimulai dari angka 0 sehingga dari kata pertama dimulai dari angka 1.



Gambar 2. 7 *Sequence Rules From Keras Library*

2.14. *Confusion Matrix*

Confusion Matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining. *Confusion matrix* digambarkan dengan tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan [21].

Tabel 2. 2 *Confusion Matrix*

Prediction	Actual	
	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Keterangan :

1. *True Positive* (TP) adalah jumlah data kelas positif yang diklasifikasikan ke dalam nilai positif.
2. *False Positive* (FP) adalah jumlah data pada kelas negatif yang diklasifikasin ke dalam nilai positif.
3. *False Negative* (FN) adalah jumlah data pada kelas positif yang diklasifikasikan ke dalam nilai negatif.
4. *True Negative* (TN) adalah jumlah data kelas negatif yang diklasifikasikan ke dalam kelas negatif.

2.15. *F1 Score*

Perhitungan F_1 score dilakukan per kelas lalu diambil nilai keseluruhan dengan cara diratakan (*averaged*) [21]. F_1 score memiliki beberapa versi; yaitu micro, macro, dan weighted macro F_1 sscore yang persamaannya dapat dilihat pada persamaan dibawah ini.

$$\boldsymbol{weighted\ F_1\ score} = \frac{1}{\sum|\boldsymbol{X_i}|} \times \sum_i |\boldsymbol{X}| \boldsymbol{F_1\ score}(\boldsymbol{X_i}) \quad (2.20)$$

Dengan :

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (2.21)$$

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad (2.22)$$

$$\boldsymbol{F_1\ score}(\boldsymbol{X}) = \mathbf{2} \times \frac{\boldsymbol{Recall \times Precision}}{\boldsymbol{Recall+Precision}} \quad (2.23)$$

Keterangan dari Persamaan (2.21), (2.22), dan (2.23). TP merupakan jumlah total dari hasil prediksi yang sama dengan label sebenarnya, FN merupakan jumlah total dari hasil prediksi yang tidak sama dengan label sebenarnya. FP merupakan jumlah total dari hasil prediksi yang benar tetapi berbeda dengan label sebenarnya, *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi, *Precision* adalah ketepatan antara informasi yang diminta dengan hasil prediksi, dan *X* merupakan kelas atau label.

2.16. Akurasi

Akurasi adalah ukuran dari seberapa baik model mengkorelasikan antara hasil dengan atribut dalam data yang telah disediakan [21].

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}} \quad (2.24)$$

Akurasi untuk menghitung total seluruh data

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Jumlah\ Prediksi\ Benar}}{\boldsymbol{Jumlah\ Label\ Total}} \quad (2.25)$$

2.17. Matriks

Matriks adalah himpunan skalar yang disusun secara empat persegi panjang menurut baris (m) dan kolom (n). Skalar-skalar tersebut disebut elemen matriks. Matriks biasanya menggunakan batas seperti ( ), [ ], atau || ||. Berikut adalah salah satu contoh dari matriks.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{32}} \end{bmatrix}$$

Pada matriks A terlihat bahwa matriks A memiliki dimensi 3 x 2. Selain itu terdapat matriks baris atau matriks kolom dimana salah satu dimensi bernilai 1. Berikut adalah contoh dari matriks baris.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \boldsymbol{a_3} \end{bmatrix}$$

2.17.1. *Transpose* Matriks

Transpose matriks yaitu suatu matriks yang melakukan pertukaran antara dimensi kolom dan barisnya atau sebuah matriks yang memindahkan elemen-elemen pada kolom menjadi elemen baris atau sebaliknya. Berikut adalah contoh dari Transpose matriks.

$$\boldsymbol{A = \begin{bmatrix} 1 & 7 \\ 2 & 8 \\ 3 & 9 \end{bmatrix} \; maka \; A^T = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}}$$

2.17.2. *Outer Product*

Outer product merupakan perkalian perkalian antara dua buah vektor yang akan menghasilkan sebuah matriks. Berikut adalah persamaan dari outer product.

$$\boldsymbol{C = A \otimes B}$$
$$\boldsymbol{= A \cdot B^T}$$
$$\boldsymbol{= \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \cdot [b_1 \quad b_2 \quad b_3]}$$
$$\boldsymbol{= \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \\ a_4 b_1 & a_4 b_2 & a_4 b_3 \end{bmatrix}}$$

2.17.3. *Hadamard Product*

Hadamard Product merupakan perkalian dua matriks berdimensi sama yang akan menghasilkan matriks dengan dimensi yang sama.

$$\boldsymbol{C = A \odot B}$$
$$\boldsymbol{= \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix}}$$
$$\boldsymbol{= \begin{bmatrix} a_{11} b_{11} & a_{21} b_{21} \\ a_{12} b_{12} & a_{22} b_{22} \end{bmatrix}}$$