BAB II
TINJAUAN PUSTAKA

2.1. Ekstraksi Kata Kunci

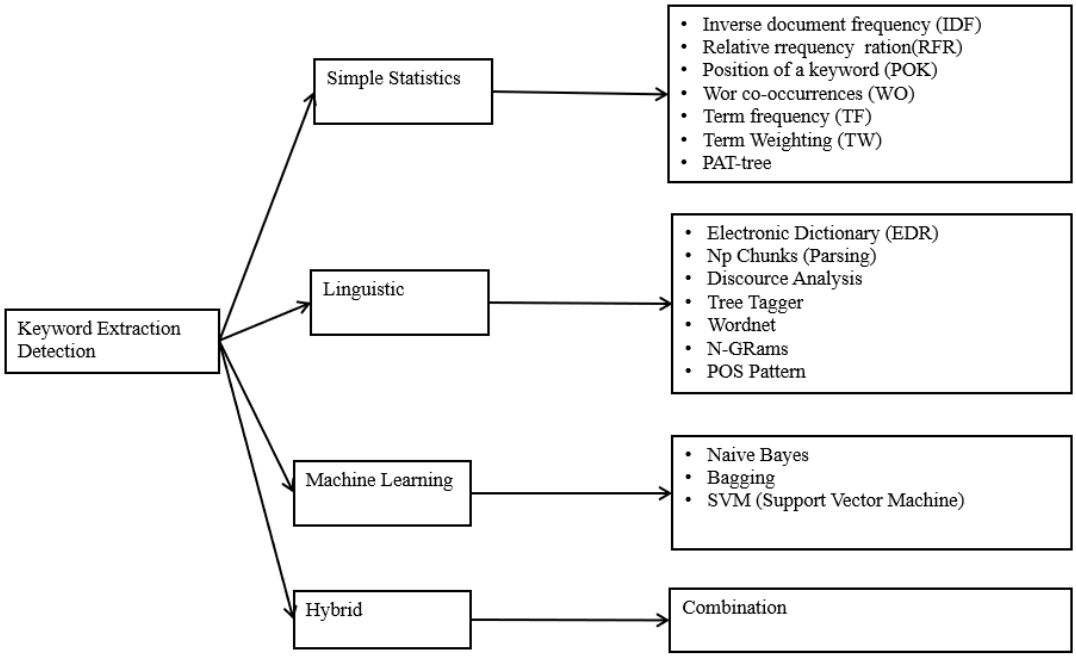
Kata kunci biasanya disebut sebagai *keyphrase* pendek. Istilah *keyphrase* dan *keyword* mengandung satu pengertian sebagai sebuah kata kunci yang penting dalam menjelaskan dan memberi gambaran terhadap isi suatu dokumen[2]. *Keyphrase* merujuk kepada kata kunci yang terdiri dari gabungan kata/*multiword*, contohnya adalah rumah sakit, kereta api, harga diri dan lain-lainnya sedangkan untuk istilah *keyword* sering ditujukan untuk kata kunci yang terdiri dari satu kata saja,contohnya adalah demam, matahari, rumah dan lain – lain [3]. Ekstraksi kata kunci (*keywords extraction*) adalah kumpulan frasa pendek yang menangkap topik utama yang dibahas pada suatu dokumen [4]. Ekstraksi kata kunci otomatis adalah pemilihan frase penting pada sebuah dokumen atau keseluruhan teks secara otomatis[5].

Ekstraksi kata kunci merupakan salah satu tahapan penting dari beberapa aplikasi text mining. Untuk mendapatkan kata kunci yang tepat secara lebih otomatis, berbagai metode ekstraksi kata kunci pun terus dikembangkan dan diuji. Pada artikel ilmiah, ekstraksi kata kunci dibutuhkan untuk memberikan alternatif kata kunci secara lebih sistematis kepada penulis jurnal. Penentuan kata kunci secara manual pada artikel ilmiah tidaklah efektif terutama jika artikel ilmiah yang akan dianalisis kata kuncinya tersebut jumlahnya sangat banyak.

Tahapan keseluruhan metode yang digunakan pada penelitian ini yaitu *preprocessing (text cleaning, tokenizing, case folding, stopword removal, POS tagging, candidates multiword extraction)*, ekstraksi kata kunci dan tahapan terakhir yaitu *postprocessing* untuk pemfilteran kata kunci yang terlalu umum. Secara umum ada 3 metode yang digunakan untuk ekstraksi kata kunci yaitu *statistical approach*,

semantic approach dan *machine learning.* Adapun untuk *machine learning* dibagi menjadi 2 yaitu *supervised learning* dan *unsupervised learning*.

Setelah melakukan penelitian dan dikelompokan menjadi beberapa bagian, maka metode yang umum digunakan menjadi 4 yaitu *simple statistics, linguistic, machine learning* dan *hybrid,* sehingga jika digambarkan dalam sebuah bagan maka akan terlihat seperti pada gambar 2.1 dibawah ini :



Gambar 2.1 Bagan Metode *Keywords Extraction*

Dari gambar 2.1 diatas memperlihatkan bahwa metode yang umum digunakan ada 4 yaitu *simple statistics, linguistic, machine learning* dan *hybrid*. Dari 4 metode tersebut kemudian ada sub – sub tekniknya. Berikut penguraian detailnya :

1. *Simple statistics* yaitu *TF/IDF, Term Weighting,* dan lain – lain.
2. *Linguistic* yaitu *N-Grams, wordnet* dan *POS Pattern* dan lain - lain.
3. *Machine Learning* yaitu *naive bayes, support vector machine, ANN* dan lain – lain.

4. Terakhir *Hybrid* yaitu kombinasi dari ketika metode diatas seperti *simple statistics, linguistic dan machine learning*.

Adapun beberapa penelitian yang telah dilakukan dapat dilihat pada tabel 2.1 yaitu sebagai berikut :

Tabel 2.1 Penelitian Sebelumnya

Nama	**Tahun**	**Metode**	**Perbedaan**
***A Statistical Approach of Keyword Extraction for Efficient Retrieval* [2].**	2013	*Statistic Approach*	Mendeteksi *keywords* dengan statistik kemunculan kata
***Keyword Extraction using Semantic Analysis* [6].**	2013	*Semantic Approach*	Mendeteksi *keywords* dengan melihat hubungan kata
***Keyword Extraction using Auto-associative Neural Networks* [7].**	2017	*Machine Learning Approach*	Mendeteksi *keywords* berdasarkan dari data latih

Pada tabel 2.1 memperlihatkan beberapa metode yang digunakan pada penelitian sebelumnya dalam mendeteksi *keywords*, adapun untuk penjelasan lebih detailnya, yaitu sebagai berikut :

2.1.1 *A Statistical Approach of Keyword Extraction for Efficient Retrieval*

Salah satu teknik dalam ekstraksi kata kunci yaitu teknik statistik, seperti penelitian yang telah dilakukan oleh Shruti Luthra, Dinkar Arora, Kanika Mittal dan

Anusha Chhabra dengan judul *A Statistical Approach of Keyword Extraction for Efficient Retrieval*. Teknik *statistic approach* sudah mampu mendeteksi kata kunci hanya dengan frekuensi kata yang muncul [2]. Penelitian yang telah dilakukan hanya bergantung pada frekuensi kata, sehingga tidak melihat keterkaitan dengan kata selanjutnya ataupun kata sebelumnya.

2.1.2 *Keyword Extraction using Semantic Analysis*

Salah satu teknik dalam ekstraksi kata kunci yaitu teknik semantik, seperti penelitian yang telah dilakukan oleh Mohamed H. Haggag dengan judul *Keyword Extraction using semantic analysis.* Teknik *semantic approach* mampu membuat relasi antar kata sehingga dapat menghasilkan kata kunci yang memilik makna sama [6]. Penelitian yang telah dilakukan kata kunci dapat diekstrak dari relasi antar kata namun belum tentu yang tidak memiliki relasi semantiknya bukanlah sebuah kata kunci.

2.1.3 *Keyword Extraction using Auto-associative Neural Networks*

Salah satu teknik dalam ekstraksi kata kunci yaitu teknik *machine learning*, seperti penelitian yang telah dilakukan oleh Germán Aquino, Waldo Hasperué dan Laura Lanzarini yang berjudul *Keyword Extraction using Auto Associative Neural Network*. Teknik *machine learning* mampu mengesktrak kata kunci berdasarkan data latih [7]. Adapun salah satu kelemahannya yaitu akurasinya menjadi kurang baik jika data latih tidak sesuai topik dan jumlahnya hanya sedikit.

Berdasarkan pada penelitian sebelumnya dengan beberapa metode yang telah digunakan pada ekstraksi kata kunci, maka penulis dalam penelitian berkontribusi menggabungkan dari ketiga metode diatas dan melengkapi kekurangan yang ada. Pada penelitian pertama yang menggunakan metode *statistic approach*, akan dilengkapi dengan metode *semantic approach* yang memiliki kelebihan untuk mendeteksi hubungan kata kemudian digabungkan dengan *machine learning*

approach yang akan memberikan pelatihan dengan menggunakan data latih yang telah disediakan.

Sehingga jika dibentuk dalam sebuah tabel 2.2 *state of the art* akan sepert dibawah ini :

Tabel 2.2 *State Of The Art*

Teknik	***Frequency Keywords***	***Position In text***	***Related Word / Phrase***	***Relevant Topic***
Semantic Approach	✓	✓		
Semantic Relatedness			✓	
Associative Neural Network	✓	✓		✓

Dari tabel 2.2 *State of the art* memperlihatkan bahwa teknik ANN sudah mampu mengekstrak kata kunci dengan metode *frequency text*, *position in text* dan berdasarkan relevansi topik akan tetapi untuk hubungan kata yaitu *related word* belum bisa terpenuhi, maka dari itu membutuhkan teknik *semantic relatedness* untuk menutupi kekurangan dari ANN.

2.2. *Semantic Relatedness*

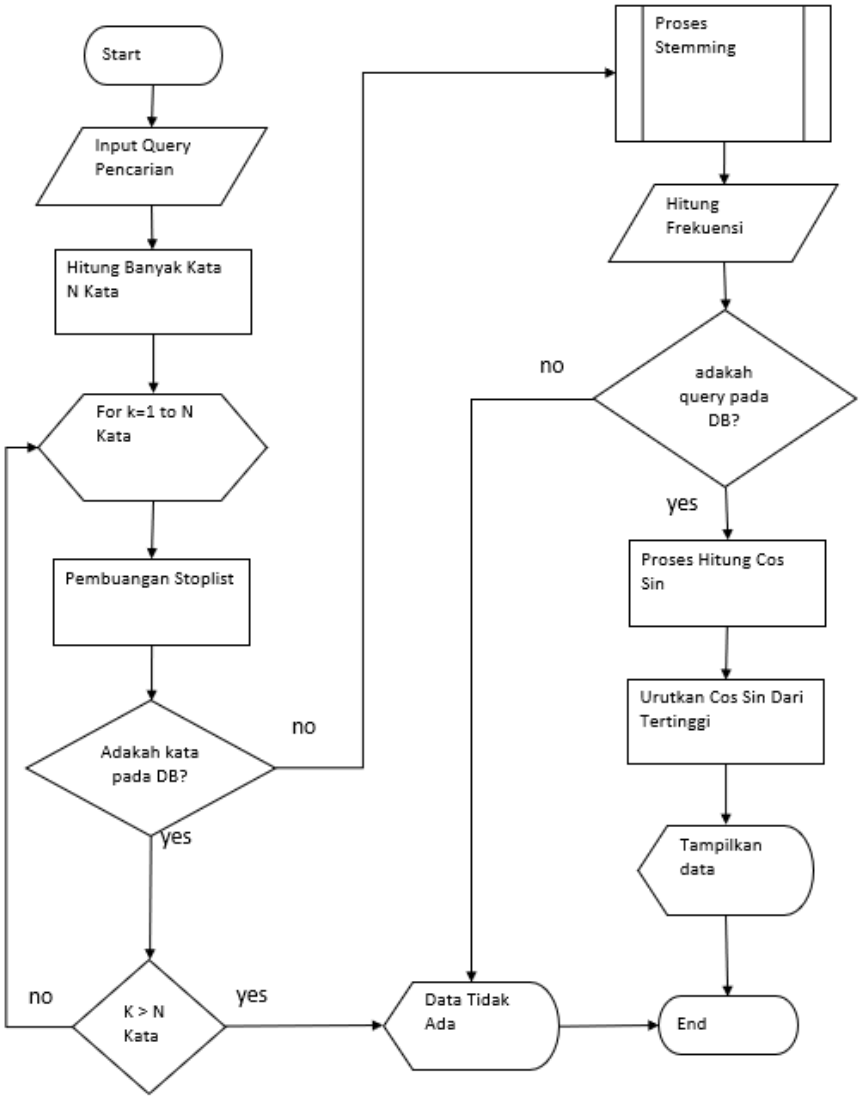
Semantic relatedness (SR) adalah suatu bentuk pengukuran yang secara kuantitatif mengidentifikasi hubungan antara dua kata atau konsep berdasarkan kesamaan atau kedekatan makna mereka[10]. *Semantic Relatedness* adalah relasi suatu matrik yang berdasar pada kesamaan makna dari suatu istilah atau isi dari semantiknya yang biasanya berbeda dengan representasi sintaksisnya dalam suatu kalimat, paragraf ataupun dokumen. Meskipun berbeda entitas akan tetapi masih mungkin memiliki keterkaitan dengan banyak kemungkinan hubungan seperti

hubungan secara langsung contohnya adalah (jari tangan, rumah sakit, dan lain - lain) [8].

Pada teknik *unsupervised learning* untuk mempelajari reprensentasi dari kata, maka salah satu caranya adalah dengan melihat hubungan dan statistik kemunculan kata pada sebuah *corpus* [11]. Dalam penelitian untuk mendapatkan nilai keterkaitan kata atau hubungan kata dan makna adalah dengan menggunakan *glove*. *Glove* adalah salah satu teknik *unsupervised learning* untuk mendapatkan representasi vektor dari setiap kata. Pelatihan yang dilakukan adalah dengan gabungan dari statistik kata - kata global yang umumnya digunakan, sehingga dapat menghasilkan substruktur linier yang cocok pada *word vector space*[11]. Model dari semantik vektor yang mewakili setiap kata dengan memiliki nilai vektor yang benar dapat digunakan untuk *information retrieval* [12], *document classification* [13], *question answering* [14], *named entity recognition* [15] dan *parsing*[16] .

Adapun untuk menghitung keterkaitan semantiknya adalah menggunakan *vector space model*. *Vector space model* adalah suatu model yang digunakan untuk mengukur kemiripan antara suatu dokumen dengan suatu *query*. *Query* dan dokumen dianggap sebagai vektor-vektor pada ruang n-dimensi, dimana t adalah jumlah dari seluruh term yang ada dalam leksikon. Leksikon adalah daftar semua *term* yang ada dalam indeks. Selanjutnya akan dihitung nilai cosinus sudut dari dua vektor, yaitu W dari tiap dokumen dan W dari kata kunci.

Pada algoritma *TF/IDF* terdapat kemungkinan adanya kesamaan bobot sehingga, diperlukan akan sulit untuk diurutkan berdasarkan bobot, maka solusi yang terbaik adalah penggunaan *vector space model*. Adapun gambar 2.2 yaitu *flowchart* dari pencarian menggunakan algortima *vector space model* sebagai berikut:



Gambar 2.2 *Flowchart Vector Space Model*

2.3. ANN

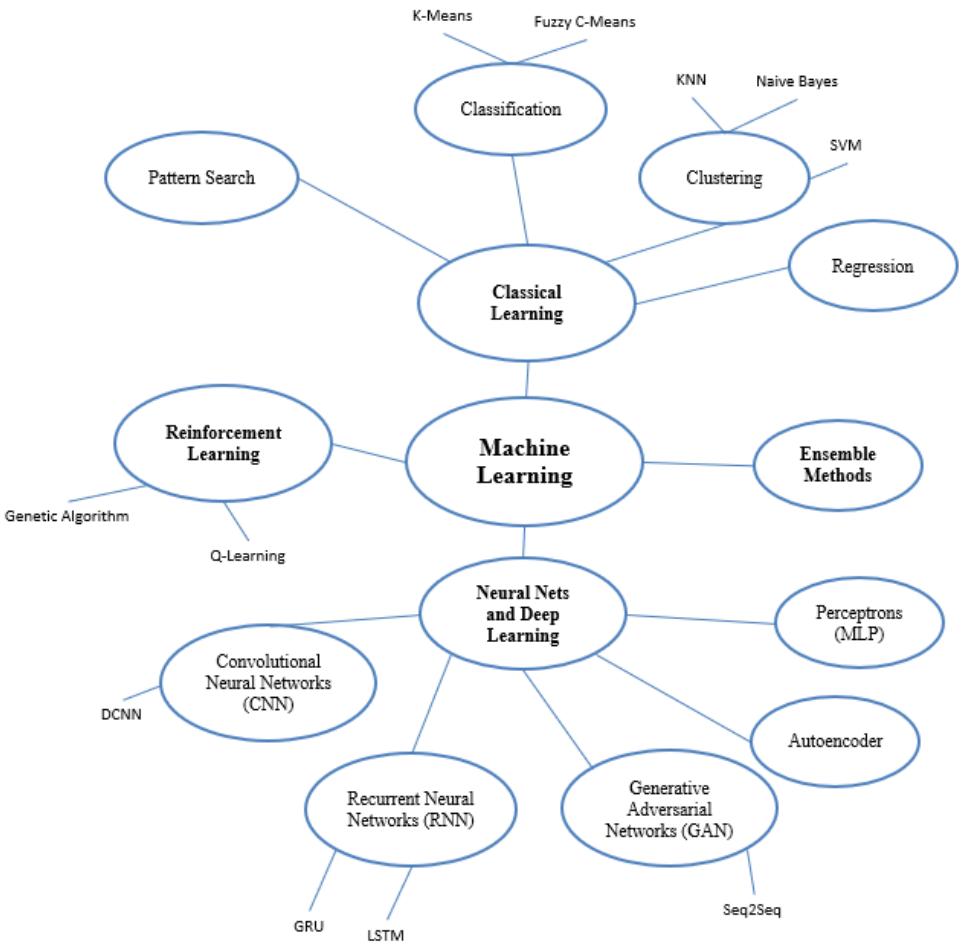
ANN adalah jaringan saraf *feed forward* yang digunakan menghasilkan pelatihan dengan memperkirakan pemetaan identitas antara *input* dengan *output* menggunakan *BPTT* atau model pembelajaran yang serupa. ANN adalah metode *ensemble-based* yang terinspirasi oleh fungsi dan struktur korelasi jaringan saraf di

otak. Metode ini beroperasi dengan mensimulasikan memori jaringan saraf jangka pendek dan jangka Panjang [17].

Fitur dari *ANN* adalah membuat antara *input* memiliki banyak *input* dengan pola yang berbeda menghasilkan *output* yang lebih kecil. Pada aplikasi pemodelan bahasa terdapat dua kriteria yaitu :

1. Kriteria pertama memungkinkan untuk dapat menghitung *score* kalimat yang tidak beraturan berbasiskan seberapa mungkin kalimat tersebut muncul didalam dunia nyata (*real world*). Hal tersebut memungkinkan mampu mengukur kesesuaian *grammatical* dan *semantic*.
2. Kriteria kedua pemodelan Bahasa memungkinkan untuk dapat menghasilkan teks baru.

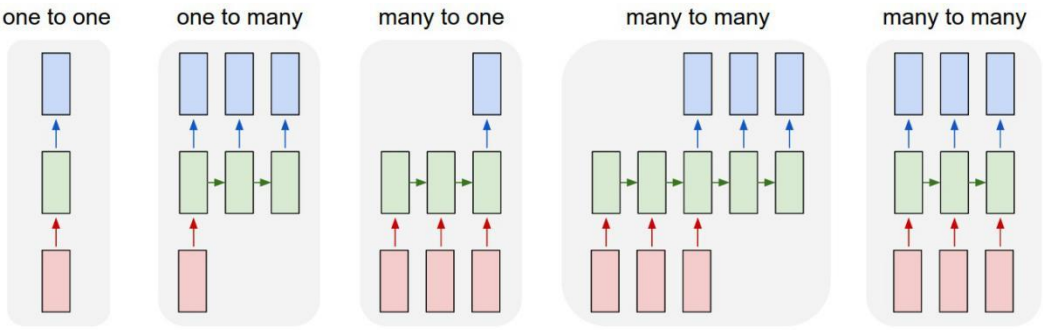
Pada penelitian kali ini hanya menggunakan pada kriteria pertama yaitu untuk menghitung score kalimat yang tidak beraturan. Untuk menghitung suatu kalimat biasanya menggunakan algoritma RNN. RNN juga salah satu jaringan saraf tiruan untuk menangani masalah NLP. RNN berguna untuk mengolah informasi secara bertahap. Jika mengacu pada *neural network* pada umumnya, biasanya semua *input* dan *output* tidak bergantung satu dengan yang lainnya, akan tetapi jika menggunakan RNN bisa mengatasi tugas yang bertumpuk dan dapat melakukan tugas yang sama pada setiap elemen pada sebuah urutan, kemudian memproses *output* yang mengacu pada komputasi sebelumnya. Sehingga dapat disimpulkan bahwa RNN memiliki *memory* untuk menyimpan hasil rekaman informasi sebelumnya. Akan tetapi RNN memiliki kelemahan, tidak mampu menyimpan memori dalam jangka panjang. Sehingga dibutuhkan tipe baru dari RNN yaitu LSTM. Adapun jika digambarkan dalam bentuk *mind map* untuk *Machine Learning* adalah seperti pada gambar 2.2 dibawah ini :



Gambar 2.3 *Mind Map Machine Learning*

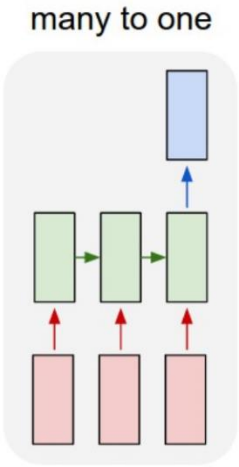
Pada gambar 2.3 diatas merupakan gambaran *simple* dari *machine learning* , terlihat bahwa LSTM adalah turunan dari RNN. LSTM adalah salah satu teknik yang biasa digunakan ketika membutuhkan rekaman memori yang cukup panjang dan ketergantungan jangka panjang (*long term dependency*) serta ketika membutuhkan jumlah *network input training* dan *network output target* yang berbeda. Memori pada LSTM disebut juga *cell.*

Model LSTM memiliki bermacam - macam *network input* dan *output target*, seperti pada gambah dibawah ini :



Gambar 2.4 *Network input* dan *ouput* pada LSTM

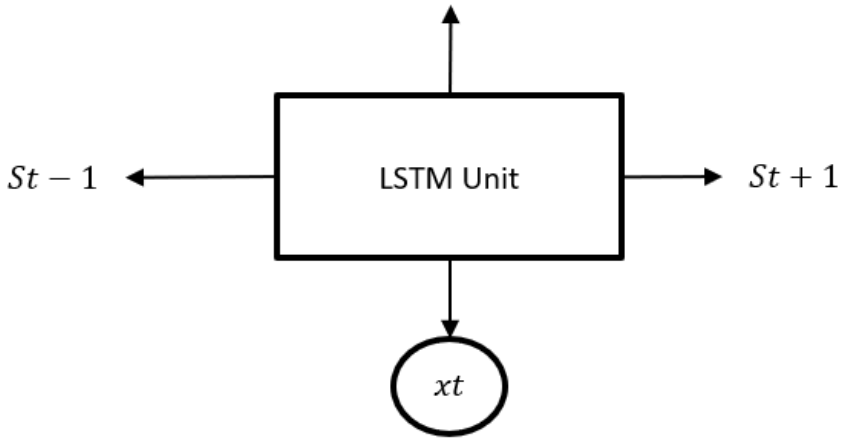
Adapun pada gambar 2.4 *network input* dan *output* yaitu bagaimana gambaran umum dari beberapa pola pada LSTM. Adapun gambaran pola dari *input* dan *ouput* yang dibutuhkan pada penelitian ini adalah sebagai berikut :



Gambar 2.5 *Network Input dan Network Output Model Many To One* pada LSTM

Pada gambar 2.5 yaitu *Network Input dan Network Output Model Many To One* pada *LSTM* diatas menyatakan bahwa network *input* lebih banyak akan tetapi *output* hanya satu. Dalam penelitian ini adalah inputan seperti frekuensi kata, posisi pada paragraf atau kalimat kemudian keterlibatan *semantic relatedness* dan data latih *backpropagation* yang telah diproses dan menghasilkan satu *output* yaitu kata kunci.

Pengklasifikasian yang digunakan dalam penelitian ini adalah jaringan saraf asosiatif atau biasa disebut *autoencoder*. Jaringan saraf bertujuan untuk mempelajari suatu fungsi yang memetakan dari elemen suatu set *input* ke elemen set *output*. Dalam *autoencoder* set keluaran sama dengan set *input*, sehingga jaringan ini berupaya mempelajari fungsi identitas set pelatihan seperti pada gambar 2.5 dibawah ini .

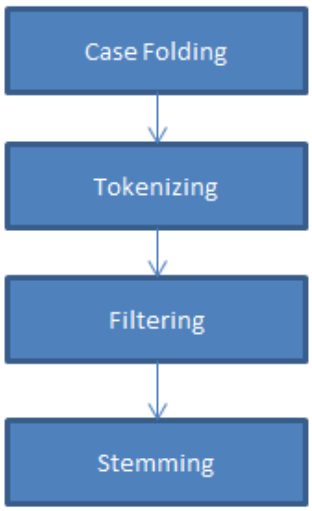


Gambar 2.6 LSTM Unit

Pada gambar 2.6 diatas $St-1$ adalah *previous hiden state*, sedangkan untuk $St+1$ adalah *next hidden state*.

2.4. *Text Processing*

Berdasarkan ketidakteraturan struktur data teks, maka proses dalam *information retrieval* atau*text mining* memerlukan beberapa tahap awal yang pada intinya adalah mempersiapkan agar teks dapat diubah menjadi lebih terstruktur. Salah satu implementasi dari text mining adalah tahap text preprocessing. Tahap *text preprocessing* adalah tahapan dimana aplikasi melakukan seleksi data yang akan diproses pada setiap dokumen. Proses *preprocessing* ini meliputi (1) *case folding*, (2) *tokenizing*, (3) *filtering*, dan (4) *stemming*.



Gambar 2.7 Proses *Preprocessing*

Pada gambar 2.7 yaitu tahapan proses *preprocessing*, adapun tahapan detailnya akan dijelaskan dibawah ini :

2.4.1. *Case Folding*

Case Folding dibutuhkan dalam mengkonversi keseluruhan teks dalam dokumen menjadi suatu bentuk standar (biasanya huruf kecil atau lowercase). Sebagai contoh, user yang ingin mendapatkan informasi "KOMPUTER" dan mengetik "KOMPUTER", "KomPUter", atau "komputer", tetap diberikan hasil retrieval yang sama yakni "komputer". Case folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai dengan 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap delimiter.

2.4.2. *Tokenizing*

Tahap *Tokenizing* adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Contoh pada kalimat *tokenizing* adalah sebagai berikut :

Tabel 2.3 *Tokenizing*

Text Input	***Text Output***
Ibu sedang pergi ke pasar membeli telur dan daging ayam.	Ibu Sedang Pergi Ke Pasar Membeli Telur Dan Daging Ayam

Pada tabel 2.3 menyebutkan menggambarkan bahwa tokenisasi secara garis besar memecah sekumpulan karakter dalam suatu teks ke dalam satuan kata, bagaimana membedakan karakter-karakter tertentu yang dapat diperlakukan sebagai pemisah kata. Sebagai contoh karakter whitespace, seperti enter, tabulasi, spasi dianggap sebagai pemisah kata. Namun untuk karakter petik tunggal (‘), titik (.), semikolon (;), titik dua (:) atau lainnya, dapat memiliki peran yang cukup banyak sebagai pemisah kata.

2.4.3. *Filtering*

Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist / stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh *stopwords* adalah “yang”, “dan”, “di”, “dari” dan seterusnya.

Tabel 2.4 *Filtering*

Text Input	***Text Output***
Ibu sedang pergi ke pasar membeli telur dan daging ayam.	Ibu Pergi Pasar Membeli Telur Daging Ayam

Pada tabel 2.4 menggambarkan bahwa kata sambung akan dihilangkan karena kata-kata seperti “dari”, “yang”, “di”, dan “ke” adalah beberapa contoh kata-kata yang berfrekuensi tinggi dan dapat ditemukan hampir dalam setiap dokumen (disebut sebagai *stopword*). Penghilangan *stopword* ini dapat mengurangi ukuran index dan waktu pemrosesan. Selain itu, juga dapat mengurangi *level noise*.

2.4.4. *Stemming*

Pembuatan indeks dilakukan karena suatu dokumen, akan tetapi tidak dapat dikenali langsung oleh suatu *information retrieval system*. Oleh karena itu, dokumen tersebut harus terlebih dahulu dipetakan ke dalam suatu representasi dengan menggunakan teks yang berada di dalamna.

Tabel 2.5 *Stemming*

Text Input	***Text Output***
Ibu sedang pergi ke pasar membeli telur dan daging ayam.	Ibu Pergi Pasar Beli Telur Daging Ayam

Dari tabel 2.5 teknik *Stemming* diperlukan selain untuk memperkecil jumlah indeks yang berbeda dari suatu dokumen juga diperlukan untuk pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa namun memiliki bentuk atau

form yang berbeda karena mendapatkan imbuhan yang berbeda.Sebagai contoh membeli akan distem ke kata dasarnya yaitu “beli”. Namun, seperti halnya *stopping*, kinerja *stemming* juga bervariasi dan sering tergantung pada *domain* bahasa yang digunakan. Proses *stemming* pada teks berbahasa Indonesia berbeda dengan stemming pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia semua kata imbuhan baik itu sufiks dan prefiks juga dihilangkan.

2.5. Pengukuran Akurasi

Pengukuran akurasi digunakan untuk mengtahui seberapa baik hasil dari penelitian yang dilakukan, berikut ini adalah pengertian dari pengukuran akurasi, metode pengukuran dan pengkategorian hasil akurasi :

2.5.1 Pengertian pengukuran akurasi

Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Pada *information extraction* terdapat beberapa istilah yaitu *precision*, *recall* dan *f-measure*.

- *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem.
- *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi.
- *F-Measure* merupakan salah satu perhitungan evalusasi dalam informasi temu kembali yang mengkombinasikan *recall* dan *precission*. Nilai *recall* dan *precission* pada suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *Recall* dan *Precission* adalah *F-Measure* yang merupakan bobot *harmonic mean* dan *recall* dan *precission*.

2.5.2 Metode Pengukuran

Metode pengukuran untuk *information retrieval* yang biasa digunakan adalah dengan menggunakan *table confusion matrix* untuk melihat nilai *precision*, r*ecall* dan *f1-score* seperti pada tabel 2.6 dibawah ini :

Tabel 2.6 *Confusion Matrix* [20]

Prediksi Kelas	*True Positive* (TP)	*False Negative* (FN)
	False Negative (FN)	*True Negative* (TN)

Tabel 2.6 *Confusion matrix* adalah salah satu metode yang digunakan untuk pengukuran, adapun beberapa penjelasannya, yaitu :

1. P (*Positive*) adalah hasil yang benar.
2. N (*Negative*) adalah hasil yang salah.
3. TP merupakan hasil dari prediksi sistem yang positif dan sesuai dengan target yang positif.
4. TN, merupakan hasil dari predisi sistem yang negatif dan sesuai dengan target yang negative.
5. FP, merupakan hasil dari prediksi sistem yang positif, namun hasil targetnya negatif.
6. FN, merupakan hasil dari prediksi sistem yang negatif, namun hasil targetnya positif.

Rumus *Accuracy*

$$Accuracy = \frac{TP + FN}{P + N} \qquad (1)$$

Rumus *Precision*

$$\text{Pr}ecision = \frac{TP}{TP + FP} \qquad (2)$$

Rumus *Recall*

$$\text{Re}\,call = \frac{TP}{TP + FN} \qquad (3)$$

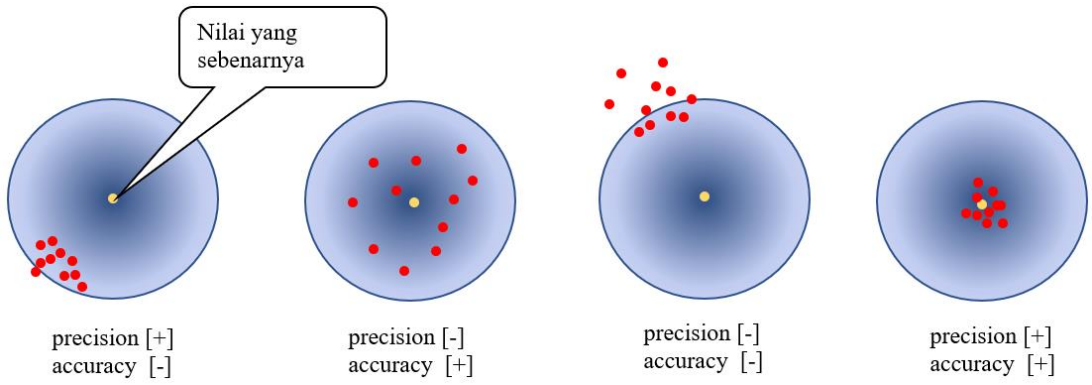
Setelah hasil klasifikasi dapat diukur kebenarannya, maka dilakukan perhitungan kombinasi nilai untuk dijadikan sebagai nilai pengukuran (*F1-score*). *F1-score* dapat dihitung dengan rumus sebagai berikut :

Rumus F1-Score

$$F1 - Score = 2X \frac{\text{Pr}\,ecision.\text{Re}\,call}{\text{Pr}\,ecision + \text{Re}\,call} \qquad (4)$$

2.5.3 Kategori Hasil Pengukuran

Pada saat pengukuran pada *information retrival* ada beberapa hasil yang akan didapatkan. Adapun jika digambarkan dari berbagai kemungkinan yang terjadi dari hasil *precision* dan *accuracy*, akan menjadi 4 kategori yaitu seperti pada gambar 2.7 dibawah ini :



Gambar 2.8 *Precision* dan *Accuracy*

Sumber : Jurnal *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning* [21].

Gambar 2.8 diatas menggambarkan kemungkinan yang akan terjadi pada hasil nilai presisi dan akurasi pada *information retrival*. Dari 4 kategori diatas yaitu :

1. Presisi [+] dan akurasi[-]

   Pada gambar terlihat bahwa titik – titik merah saling berdekatan yang artinya nilai presisinya sudah cukup baik karena tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem sudah cukup baik, akan tetapi nilai dari akurasi kurang baik seperti terlihat pada gambar bahwa titik – titik warna merah masih berjauhan dengan titik warna kuning yang artinya tingkat kedekatan antara nilai prediksi dengan nilai aktual masih jauh.

2. Presisi [-] dan akurasi [+]

   Pada gambar terlihat bahwa titik – titik warna merahnya berjauhan dengan titik titik warna merah lainnya yang artinya nilai presisinya kurang baik karena tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem masih kurang baik, akan tetapi nilai akurasinya sudah cukup baik, seperti terlihat pada gambar bahwa titik – titik warna merah sudah ada yang mendekat pada titik kuning yang berada ditengah artinya tingkat kedekatan antara nilai prediksi dengan nilai aktual sudah ada yang mendekati.

3. Presisi [-] dan akurasi [-]

   Pada gambar terlihat bahwa titik – titik warna merah saling berjauhan dan beberapa titik merah ada yang diluar lingkaran berwarna biru maka artinya adalah memiliki presisi dan akurasi yang yang tidak baik .

4. Presisi [+] dan akurasi [+]

   Pada gambar terlihat bahwa titik warna merah saling berdekatan, baik dengan titik warna merahnya ataupun dengan titik warna kuning (nilai aktualnya atau nilai yang sebenarnya) yang artinya adalah baik presisi maupun akurasinya bernilai baik.

2.5.4 Contoh Cara Pengukuran

Dimisalkan pada ekstraksi kata kunci dimasukan 100 kata kunci yang benar sesuai dengan teks atau kumpulan kalimat yang terdapat pada sebuah website tertentu untuk data latih dan dimasukan lagi 900 kata kunci lain yang didapatkan dari berbagai

website. Misal system dapat menemukan 110 kata kunci dari website tersebut. Ketika dicek manual ternyata kata kunci yang benar pada website tersebut hanya 90 kata kunci yang benar dan 20 kata kunci lainnya merupakan kata kunci yang tidak sesuai dengan isi *website*.

Jika dituliskan dalam bentuk tabel, maka akan terlihat seperti pada tabel 2.7 di bawah ini :

Tabel 2.7 Contoh Cara Pengukuran

		Nilai Sebenarnya	
		True	***False***
Nilai Prediksi	***True***	90	20
	False	10	880

Dari kasus tersebut maka dapat disimpulkan bahwa sistem memiliki *precision* sebesar 82%, *recall* sebesar 90% dan *accuracy* sebesar 97% yang didapatkan dari perhitungan berikut:

Rumus cara perhitungan *precision* :

$$\text{Precision} = \frac{\text{jumlah kata kunci yang benar yang ditemukan oleh sistem}}{\text{jumlah kata kunci yang ditemukan oleh sistem}}$$

$$Precision = \frac{90}{110} = 0.82 = 82\%$$

Rumus cara perhitungan *recall* :

$$\text{Recall} = \frac{\text{jumlah kata kunci yang benar yang ditemukan oleh sistem}}{\text{jumlah kata kunci yang sebenarnya}}$$

$$Recall = \frac{90}{100} = 0.90 = 90\%$$

Rumus cara perhitungan *accuracy* :

$$\text{Accuracy} = \frac{\text{jumlah kata kunci yang ditemukan oleh sistem}}{\text{jumlah kata kunci keseluruhan}}$$

$$Accuracy = \frac{90 + 880}{1000} = 0.97 = 97\%$$

Pada kasus – kasus tertentu cara pengukuran dengan menggunakan *precision* atau *accuracy* saja dikatakan belum cukup dalam mengukur kinerja dari sebuah sistem / metode, karena dapat menimbukan bias yang sangat fatal. Sebagai contoh, misalnya dari pengujian menggunakan 100 *keywords* yang terpilih dan 900 *keywords* lainnya, ternyata sistem hanya dapat menemukan 1 *keywords* saja yang sama dengan *website* yang dimasukan. Adapun setelah dicek secara manual, 1 *keywords* tersebut benar merupakan *keywords* yang sesuai dengan *website* yang dipilih. Sehingga di tuliskan dalam bentuk tabel 2.8 sebagai berikut:

Tabel 2.8 Cara Pengukuran Yang Kurang Tepat

		Nilai Sebenarnya	
		True	***False***
Nilai Prediksi	***True***	1	0
	False	99	900

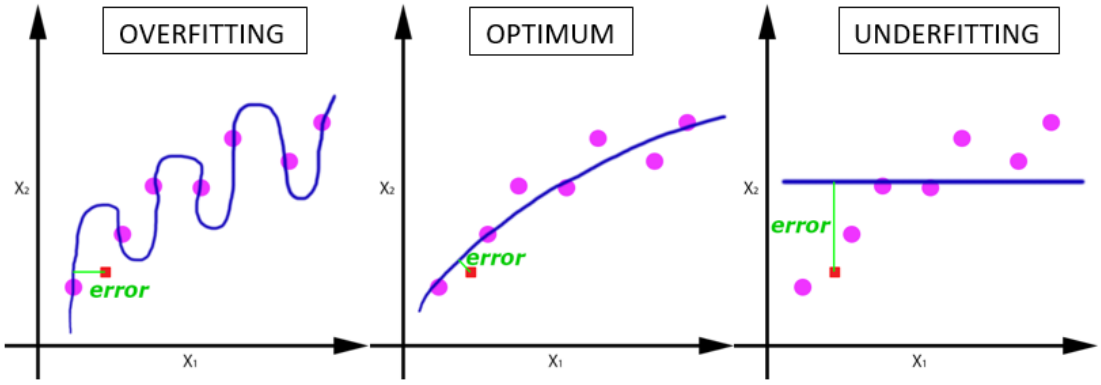
$$Precision = \frac{1}{1 + 0} = \frac{1}{1} = 1 = 100\%$$

$$Recall = \frac{1}{1 + 99} = \frac{1}{100} = 0.01 = 1\%$$

$$Accuracy = \frac{1 + 900}{1 + 900 \ + 0 + \ 99} = \frac{901}{1000} = 0.901 = 90{,}1\%$$

Dari hasil perhitungan nilai *precision* sebesar 100% dan *accuracy* sebesar 90.1%. Sekilas tampak baik, namun nilai *recall* yang didapatkan hanya sebesar 1%. Sehingga akan sangat fatal jika hanya menggunakan pengukuran pada *precision* dan *accuracy* saja. Sehingga dalam mengukur kinerja dari sebuah sistem dalam pengenalan pola atau *information retrieval*, sebaiknya menggunakan minimal dua parameter yaitu *precision* dan *recall* untuk mendeteksi bias seperti pada kasus diatas.

Epoch adalah saat seluruh dataset telah melalui proses *training* pada *Neural Netwok* sampai dikembalikan ke awal untuk sekali putaran, karena satu *Epoch* terlalu besar untuk dimasukkan (*feeding*) kedalam komputer maka dari itu kita perlu membaginya kedalam satuan kecil (*batches*). *Epoch* digunaan saat melewati seluruh dataset melalui jaringan saraf tidak cukup, maka dari itu perlu melewati *dataset* penuh beberapa kali ke jaringan saraf yang sama.



Gambar 2.9 Setiap Pertambahan *Epoch* Mengarah Pada *Underfitting*

Sumber : Jurnal *Object Detection in Infrared Images using Deep Convolutional Neural Networks* [19].

Pada gambar 2.9 menjelaskan bahwa nilai *epoch* yang dimasukan terlalu sedikit maka akan akan terjadi *underfitting* dan sebaliknya jika *epoch* yang dimasukan terlalu besar hasilnya pun sama akan terjadi *overfitting*, maka sebaiknya adalah dengan memasukan nilai *epoch* yang seimbang sehingga akan menghasilkan nilai yang *optimum*.

Batch Size adalah jumlah sampel data yang disebarkan ke *neural network*. Contohnya jika ada 100 dataset dan batch size 5 maka algoritma ini akan menggunakan 5 sampel data pertama dari 100 data yang kita miliki (ke1, ke2, ke3, ke4, dan ke5) lalu disebarkan atau dilakukan pelatihan oleh *Neural Network* sampai selesai kemudian mengambil kembali 5 *sampel* data kedua dari 100 data (ke6, ke7, ke8, ke9, dan ke10), dan begitu seterusnya sampai 5 sampel data ke 20 (100⁄5=20). Sedangkan *iterations* adalah jumlah *batch* yang diperlukan untuk menyelesaikan satu *epoch.*

Contoh ketika memiliki 2000 data latih yang akan digunakan, maka kita dapat membagi *dataset* dari 2000 data tersebut menjadi *batch* dari 500 maka akan dibutuhkan 4 iterasi untuk menyelesaikan 1 *epoch*.