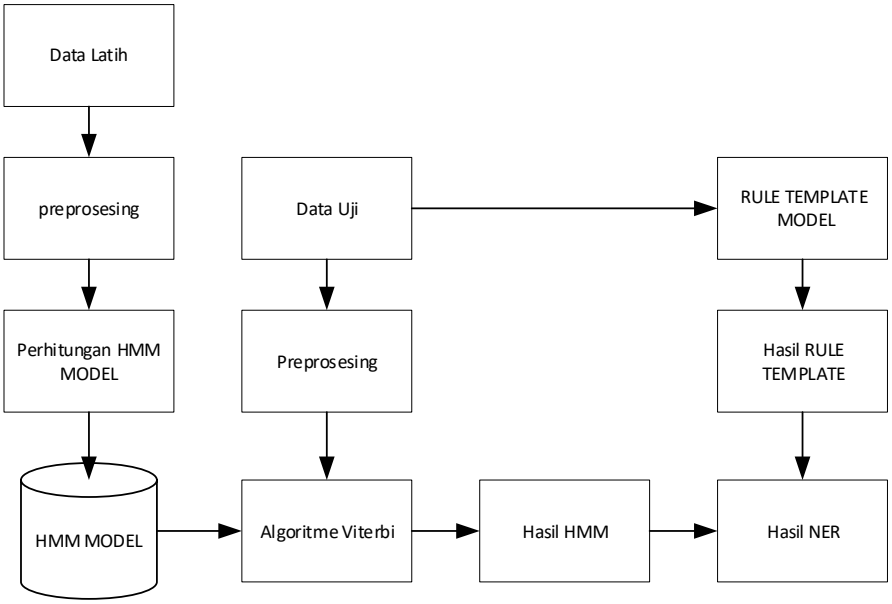
BAB 1 PERANCANGAN

Pada bab ini akan dibahas mengenai perancangan yang terdiri dari pembahasan mengenai deskripsi umum sistem, perhitungan manual, alur perancangan sistem, perancangan antarmuka serta perancangan pengujian.

1.1 Diskripsi Umum Sistem

Penelitian penerapan *named entity recognition* pada *e-commerce* menggunakan *rule template* dan *hidden markov model* dibangun dengan tujuan untuk melakukan ektraksi informasi, sehingga diharapkan dapat mengenali nama entitas pada teks percakapan atau pertanyaan mengenai produk terutama produk ponsel.



Gambar 1.1 Deskripsi umum sistem

Gambar 4.1 adalah gambaran umum tahapan untuk melakukan pengenalan nama entitas. Langkah awal dari penerapan *named entity recognition* ini ialah dengan melakukan pemberian entitas pada data yang akan digunakan sebagai data latih. Langkah selanjutnya adalah melakukan *pre-processing* pada data latih maupun data uji. Kemudian dilakukan tahapan-tahapan pada metode *Rule Template, Hidden Markov Model* hingga perancangan *Additive smoothing.* Hasil akhir sistem ini yaitu merupakan pemberian entitas pada percakapan atau pertanyaan mengenai produk ponsel, entitas tersebut antara lain yaitu *MEREK, TIPE, HARGA, SPEK, N_SPEK* dan *N_TAG*. Tabel 4.1 menunjukan beberapa contoh pengenalan entitas pada penelitian ini.

Tabel 1.1 Contoh Entitas yang akan dikenali

ENTITAS	CONTOH
MEREK	Samsung, Apple, Xiaomi, Sony, Nokia
TIPE	Galaxy, iphone, N70, S7, 5S

Tabel 1.2 Contoh Entitas yang akan dikenali (lanjut)

ENTITAS	CONTOH
HARGA	3 juta, 500, 300 ribu, 2jutaan

SPEK	Memori, cpu, ram, kamera, layar
N_SPEK	Merah, 3G, 4G, 32GB, 64GB, 15MP
N_TAG	Kata yang tidak dikenali atau tidak memiliki entitas

1.2 Perhitungan Manual

1.2.1 Rule Template

Berikut penerapan *named entity recognition* untuk mengenali fitur produk ponsel pada *e-commerce* menggunakan *rule template.* Diasumsikan memiliki data sebagai berikut Tabel 4.3.

Tabel 1.3 Data Uji Rule Template

Dokumen	Teks Pertanyaan
Uji	harga untuk hp samsung s7 yang memori internal 64gb berapa mas

Langkah awal yang dilakukan adalah membaca setiap kata pada data uji yang akan dikenali. Langkah selanjutnya ialah membuat *rule* dengan menggunakan *Regular Expression* (RE). Setelah pembuatan *rule* menggunakan RE selesai dibuat selanjutnya data uji akan diseleksi apakah kata pada data uji tersebut termasuk kata yang sesuai atau termasuk dengan *rule*, jika kata tersebut sesuai maka kata tersebut akan memiliki entitas sesuai dengan entitasnya. Selain menggunakan RE *rule template* ini juga menggunakan pengenalan secara *pattern-based* dimana membuat dataset yang berisi kumpulan data atau kata yang sesuai dengan entitas yang akan dikenali kemudian mencocokkan data uji dengan dataset yang telah dibuat. Sebagai contoh pengenalan yang digunakan pada *rule template* sebagai berikut Tabel 4.4.

Tabel 1.4 Contoh pengenalan Rule Template

RE	Pattern	Data Uji
<TIPE>: ([aA-zZ]{1,2}[0-9]{1,4}[aA-zZ+]{0,4})	<MEREK>: samsung, nokia, LG	harga untuk hp samsung<MEREK> S7<TIPE> yang memori internal 64gb berapa mas

1.2.2 Hidden Markov Model

Berikut penerapan *named entity recognition* untuk mengenali fitur produk ponsel pada *e-commerce* menggunakan *Hidden Markov Model.* Diasumsikan memiliki data sebagai berikut Tabel 4.5.

Tabel 1.5 Data dokumen pertanyaan produk

Dokumen	Teks Pertanyaan
Latih	mas<N_TAG> saya<N_TAG> mencari<N_TAG> hp<N_TAG> samsung<MEREK> galaxy<TIPE> yang<N_TAG> harganya<HARGA> sekitar<N_TAG> 5juta<HARGA> ada<N_TAG>

Latih	<START> untuk<N_TAG> hp<N_TAG> samsung<MEREK> S7<TIPE> warna<SPEK> apa<N_TAG> saja<N_TAG> yang<N_TAG> tersedia<N_TAG> <END>
Latih	untuk<N_TAG> samsung<MEREK> S7<TIPE> berapa<N_TAG> memori<SPEK> internal<SPEK> yang<N_TAG> tersedia<N_TAG>
Latih	untuk<N_TAG> samsung<MEREK> S7<TIPE> yang<N_TAG> memori<SPEK> internal<SPEK> 64GB<N_SPEK> apakah<N_TAG> tersedia<N_TAG>
Latih	harga<HARGA> untuk<N_TAG> hp<N_TAG> samsung<MEREK> galaxy<TIPE> S7<TIPE> yang<N_TAG> 64GB<N_SPEK> berapa<N_TAG>
Latih	kalau<N_TAG> harga<HARGA> untuk<N_TAG> samsung<MEREK> S7<TIPE> yang<N_TAG> 32GB<N_SPEK> berapa<N_TAG>
Latih	harga<HARGA> samsung<MEREK> S7<TIPE> yang<N_TAG> memori<SPEK> 64GB<N_SPEK> apa<N_TAG> lebih<N_TAG> mahal<N_TAG> dari<N_TAG> pada<N_TAG> 32GB<N_SPEK>
Latih	samsung<MEREK> S7<TIPE> ini<N_TAG> sudah<N_TAG> mendukung<N_TAG> OS<SPEK> nougat<N_SPEK> belum<N_TAG>
Latih	hp<N_TAG> samsung<MEREK> S7<TIPE> ini<N_TAG> sudah<N_TAG> mendukung<N_TAG> jaringan<SPEK> 4G<N_SPEK> belum<N_TAG>
Latih	mas<N_TAG> saya<N_TAG> beli<N_TAG> samsung<MEREK> S7<TIPE> yang<N_TAG> memori<SPEK> 64GB<N_SPEK> ada<N_TAG>
Uji	harga untuk hp samsung S7 yang memori internal 64gb berapa mas

Langkah awal yang dilakukan adalah menghitung nilai *start probability*. Nilai *start probability* didapatkan dari hasil perhitungan kemunculan masing-masing *entitas* pada awal dokumen dibagi dengan banyaknya dokumen latih. Berdasarkan Tabel 4.5 didapatkan N_TAG sebanyak 7 kali, MEREK 1 kali dan HARGA 2 kali, maka nilai *start probability* dari N_TAG bernilai 7/10 = 0.7. Keseluruhan, hasil perhitungan *start probability* ditunjukan pada Tabel 4.6.

Tabel 1.6 Hasil perhitungan *start probability* data latih

Entitas	Nilai
MEREK	1/10 = 0.1
TIPE	0/10 = 0

Tabel 1.7 Hasil perhitungan *start probability* data latih (lanjut)

Entitas	Nilai
HARGA	2/10 = 0.2
SPEK	0/10 = 0
N_SPEK	0/10 = 0
N_TAG	7/10 = 0.7

Setelah dihitung hasil dari *start probability* langkah selanjutnya yaitu menghitung nilai *transition probability* dengan menggunakan rumus pada Persamaan 2.1. Menghitung nilai *transition probability* dengan cara menghitung frekuensi kemunculan antar entitas dan transisi antar entitas. Berdasarkan data latih didapatkan:

Tabel 1.8 Frekuensi entitas pada data latih

Frekuensi Entitas	Nilai
MEREK	10
TIPE	11
HARGA	5
SPEK	9
N_SPEK	8
N_TAG	49

Tabel 1.9 Frekuensi transisi entitas pada data latih

Frekuensi Transisi	MEREK	TIPE	HARGA	SPEK	N_SPEK	N_TAG
MEREK	0	10	0	0	0	0
TIPE	0	1	0	1	0	9
HARGA	1	0	0	0	0	4
SPEK	0	0	0	2	5	2
N_SPEK	0	0	0	0	0	7
N_TAG	8	0	3	6	3	20

Misal untuk menghitung nilai *transition probability* dari entitas MEREK ke TIPE dengan menggunakan rumus pada Persamaan 2.1 didapatkan:

$P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$, maka $P(t_{TIPE}|t_{MEREK}) = \frac{C(t_{MEREK},t_{TIPE})}{C(t_{MEREK})} = \frac{10}{10} = 1$

perhitungan tersebut juga digunakan untuk mencari nilai *transition probability* MEREK ke HARGA, MEREK ke SPEK, MEREK ke N_SPEK dan seterusnya. Tabel 4.10 menunjukan keseluruhan hasil perhitungan *transition probability* yang didapatkan dari data latih.

Tabel 1.10 Hasil perhitungan *transition probability* data latih

Transition	MEREK	TIPE	HARGA	SPEK	N_SPEK	N_TAG
MEREK	0	1	0	0	0	0
TIPE	0	0.09	0	0.09	0	0.8
HARGA	0.2	0	0	0	0	0.8
SPEK	0	0	0	0.2	0.5	0.2
N_SPEK	0	0	0	0	0	0.8
N_TAG	0.1	0	0.06	0.1	0.06	0.4

Setelah mendapatkan hasil perhitungan *transition probability* langkah selanjutnya yaitu menghitung nilai *emission probability* dengan menggunakan rumus pada Persamaan 2.2. Menghitung nilai *emission probability* dengan cara menghitung frekuensi kemunculan suatu kata pada suatu entitas tertentu terhadap frekuensi kemunculan entitas tersebut. Misal untuk menghitung nilai *emission probability* dari kata 'samsung' pada entitas MEREK dengan menggunakan rumus pada Persamaan 2.2 didapatkan:

$P(w_i|t_i) = \frac{C(t_i,w_i)}{C(t_i)}$, maka $P(w_{samsung}|t_{MEREK}) = \frac{C(t_{MEREK},w_{samsung})}{C(t_{MEREK})} = \frac{10}{10} = 1$

perhitungan tersebut juga digunakan untuk mencari nilai *emission probability* 'samsung' pada entitas TIPE, 'samsung' pada entitas HARGA dan seterusnya. Tabel 4.11 menunjukan keseluruhan hasil perhitungan *emission probability* yang didapatkan dari data latih.

Tabel 1.11 Hasil perhitungan *emission probability* data latih

Emission	MEREK	TIPE	HARGA	SPEK	N_SPEK	N_TAG
mas	0	0	0	0	0	0.04
saya	0	0	0	0	0	0.04
mencari	0	0	0	0	0	0.02
hp	0	0	0	0	0	0.08
samsung	1	0	0	0	0	0
galaxy	0	0.18	0	0	0	0
yang	0	0	0	0	0	0.16
harganya	0	0	0.2	0	0	0
sekitar	0	0	0	0	0	0
5juta	0	0	0.2	0	0	0
ada	0	0	0	0	0	0.02

Tabel 1.12 Hasil perhitungan *emission probability* data latih (lanjutan)

Emission	MEREK	TIPE	HARGA	SPEK	N_SPEK	N_TAG
untuk	0	0	0	0	0	0.1
s7	0	0.81	0	0	0	0
warna	0	0	0	0.11	0	0
apa	0	0	0	0	0	0.04
saja	0	0	0	0	0	0.02
tersedia	0	0	0	0	0	0.06
berapa	0	0	0	0	0	0.06
memori	0	0	0	0.44	0	0
internal	0	0	0	0.22	0	0

64gb	0	0	0	0	0.5	0
apakah	0	0	0	0	0	0.02
harga	0	0	0.6	0	0	0
kalau	0	0	0	0	0	0.02
32gb	0	0	0	0	0.25	0
Lebih	0	0	0	0	0	0.02
mahal	0	0	0	0	0	0.02
dari	0	0	0	0	0	0.02
pada	0	0	0	0	0	0.02
ini	0	0	0	0	0	0.04
sudah	0	0	0	0	0	0.04
mendukung	0	0	0	0	0	0.04
os	0	0	0	0.11	0	0
nougat	0	0	0	0	0.12	0
belum	0	0	0	0	0	0.04
jaringan	0	0	0	0.11	0	0
4g	0	0	0	0	0.12	0
yasudah	0	0	0	0	0	0.02
beli	0	0	0	0	0	0.02
aja	0	0	0	0	0	0.02

Setelah perhitungan *transition probability* dan *emission probability* didapatkan, maka langkah selanjutnya adalah melakukan proses *decoding*. Proses *decoding* dalam penelitian ini menggunakan algoritme Viterbi, dimana proses ini memiliki tujuan untuk mencari jalur atau urutan terbaik dari data uji berdasarkan data latih. Perhitungan Viterbi menggunakan rumus pada Persamaan 2.3. Misal, untuk mencari nilai Viterbi pada data uji dengan kata pertama yaitu kata 'harga' dengan menggunakan Persamaan 2.3:

$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) a_{ij} b_j(o_t)$, maka nilai Viterbi kata 'harga' pada entitas MEREK didapatkan dari perhitungan nilai *start probability* * nilai *emission probability* sehingga data uji kata 'harga' pada entitas MEREK adalah 0.1 * 0 = 0. Perhitungan ini dilakukan seterusnya hingga ditemukan entitas yang memiliki nilai terbesar atau optimum. Setelah menghitung kata pertama pada data uji selanjutnya porses Viterbi menghitung kata berikutnya yaitu 'untuk', perhitungan ini didapatkan dari perkalian *transition probability* dengan *emission probability,* yang mana untuk *transition probability* adalah hasil perhitungan *transition probability* pada data latih yang memiliki nilai optimum pada perhitungan Viterbi pada kata sebelumnya.

Proses Viterbi ini dilakukan seterusnya pada semua kata pada data uji. Tabel 4.13 menunjukan keseluruhan hasil perhitungan algoritme Viterbi pada data uji dengan data latih yang tersedia.

Tabel 1.13 Hasil perhitungan Viterbi

Viterbi	MEREK	TIPE	HARGA	SPEK	N_SPEK	N_TAG
harga	0	0	0.12	0	0	0
untuk	0	0	0	0	0	0.08
hp	0	0	0	0	0	0.03
samsung	0.16	0	0	0	0	0
s7	0	0.81	0	0	0	0
yang	0	0	0	0	0	0.13
memori	0	0	0	0.05	0	0
internal	0	0	0	0.04	0	0
64gb	0	0	0	0	0.27	0
berapa	0	0	0	0	0	0.05
mas	0	0	0	0	0	0.01

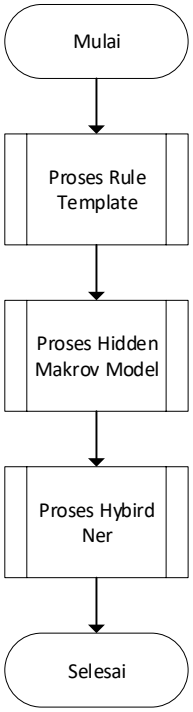
Setelah melakukan perhitungan pada data uji berdasarkan algoritme Viterbi selanjutnya adalah memberikan entitas pada data uji, pemberian entitas ini berdasarkan nilai maksimal atau optimum pada perhitungan Viterbi tersebut. Hasil pemberian entitas ditunjukan pada Tabel 4.14 berdasarkan hasil perhitungan Viterbi pada Tabel 4.13.

Tabel 1.14 Hasil pemberian entitas pada data uji

Kata	Entitas
harga	HARGA
untuk	N_TAG
hp	N_TAG
samsung	MEREK
s7	TIPE
yang	N_TAG
memori	SPEK
internal	SPEK
64gb	N_SPEK
berapa	N_TAG
mas	N_TAG

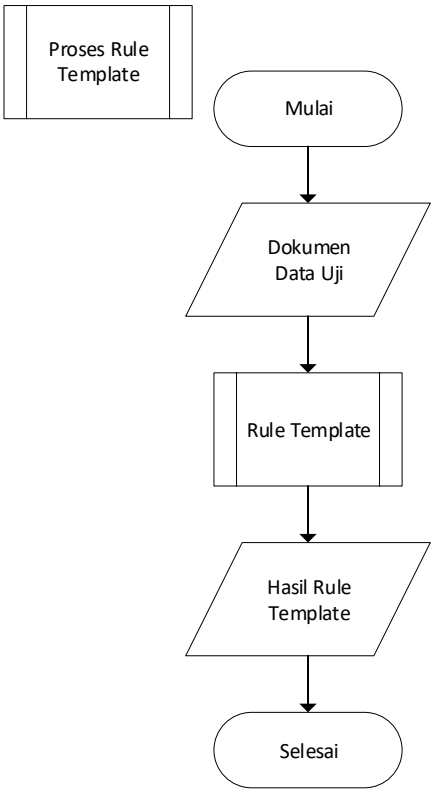
1.3 Perancangan Sistem

Sub bab ini menjelaskan tentang proses penerapan *named entity recognition* (NER) untuk mengenali fitur produk pada *e-commerce* menggunakan *Rule Template* dan *Hidden Markov Model*.



Gambar 1.2 Diagram alir Pengenalan NER

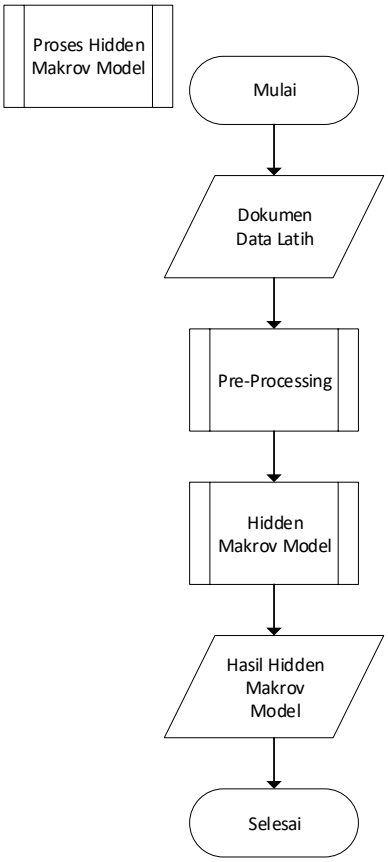
Proses pengenalan NER untuk identifikasi fitur produk dibagi menjadi empat rancangan sistem. Rancangan pertama ialah perancangan sistem menerapkan metode *Rule Template,* rancangan kedua ialah perancangan sistem menerapkan metode *Hidden Markov Model* dengan algoritme Viterbi*,* rancangan ketiga ialah perancangan sistem *Hidden Markov Model* dengan algoritme Viterbi dan penambahan teknik *additive smoothing* pada perhitungan HMM dan rancangan keempat ialah rancangan *hybrid* atau penggabungan hasil dari 2 metode. Perancangan *Rule Template* ditunjukkan pada Gambar 4.3.



Gambar 1.3 Diagram alir Rule Template

Berdasarkan Gambar 4.3, dokumen data uji akan di proses menggunakan metode *Rule Template*. Proses alur pada *Rule Template* ini menggunakan *Regular Expression* untuk pembuatan *rule*. Setelah data uji diproses oleh sistem dengan *Regular Expression* akan menghasilkan data uji yang sudah memiliki entitas sesuai dengan metode *Rule Template* yang sudah dibuat. Pada perancangan berikutnya adalah Perancangan *Hidden Markov Model.* Perancangan *Hidden Markov Model* ditunjukkan pada Gambar 4.4.

Pada perancangan *Hidden Markov Model* berdasarkan Gambar 4.4. Dokumen yang digunakan sebagai data latih akan dilakukan *pre-processing.* Setelah dokumen latih telah dilakukan *pre-processing* maka data telah siap untuk dibentuk pemodelan menggunakan *Hidden Markov Model*. Di dalam proses *Hidden Markov Model* terdapat proses Viterbi dalam penentuan urutan atau pemberian entitas pada data uji dan proses *Hidden Markov Model* ini terdapat 2 model, model pertama tanpa penambahan *Additive smoothing* dan model kedua dengan penambahan *Additive smoothing*. Keluaran dari proses ini adalah data uji yang sudah memiliki entitas.



Gambar 1.4 Diagram alir Hidden Markov Model

1.3.1 Rule Template

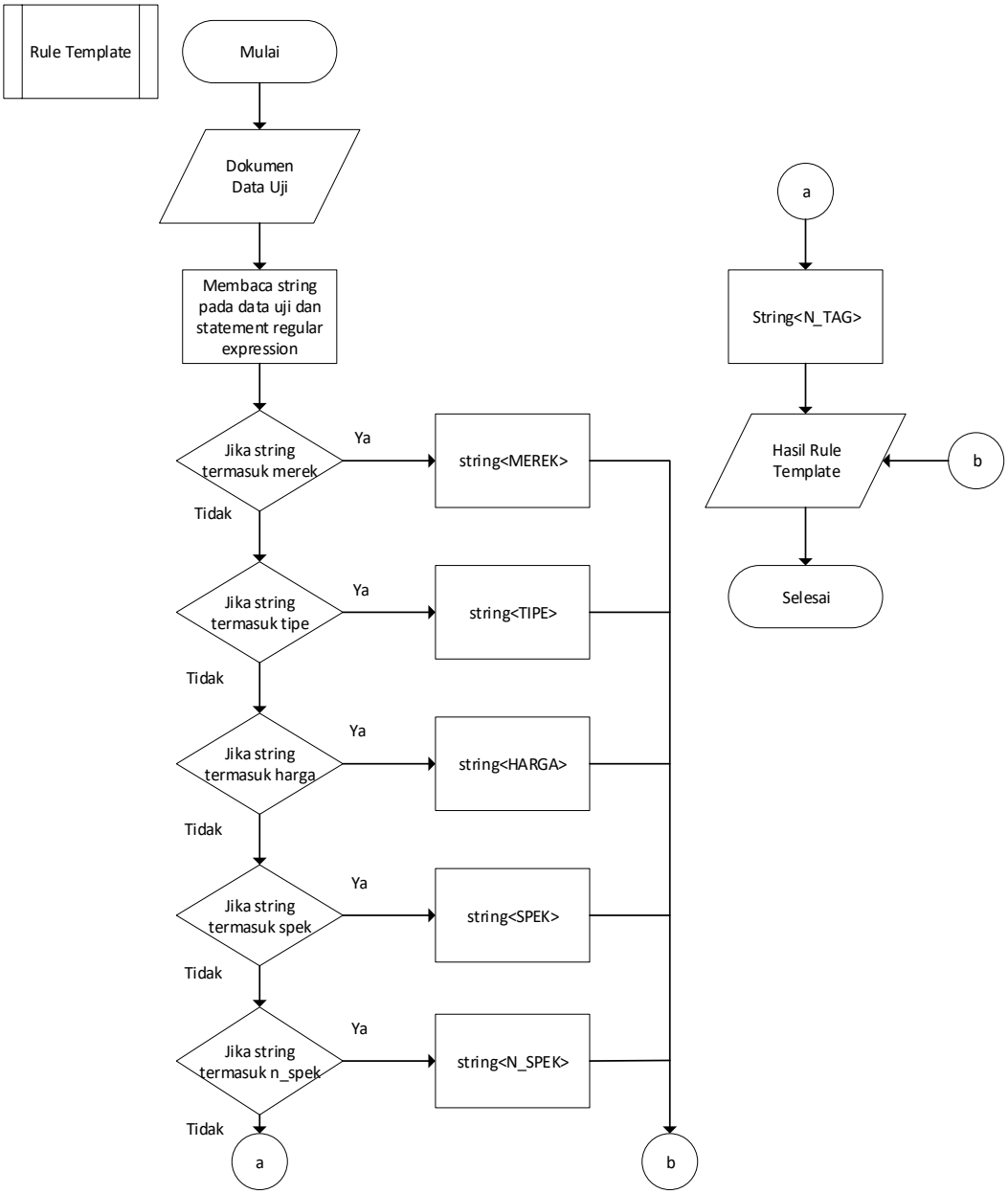
Tahap ini merupakan tahap awal dari metode *Rule Template* dalam memberikan entitas pada data uji. Data uji ini diolah dengan *Regular Expression* (RE) sehingga mendapatkan hasil berupa data uji yang sudah memiliki entitas sesuai *rule* yang telah dibuat. Proses *Rule template* ini ditunjukan pada Gambar 4.5.

Berdasarkan Gambar 4.5 tahapan diagram alir proses *rule template* sebagai berikut:

1. Memasukan dokumen uji.
2. Melakukan proses *regular expression.*

   Pada proses ini membaca data uji dan melakukan proses RE yang sudah dibuat sesuai dengan *rule* setiap entitas.
3. Melakukan proses pencocokan kata pada data uji.

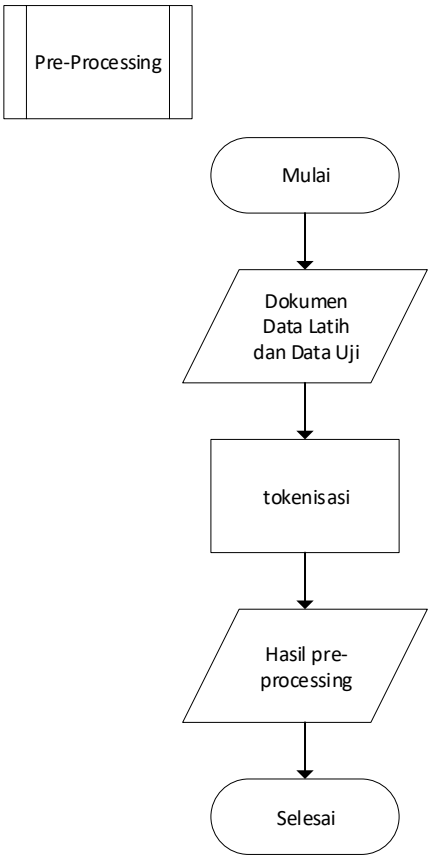
   Pada proses ini kata yang sudah melakukan proses RE akan dicocokan ke dalam data merek, tipe, harga, spek, dan n_spek jika kata tersebut cocok maka kata tersebut termasuk entitas sesuai entitasnya, jika tidak maka kata tersebut termasuk entitas n_tag.
4. Menghasilkan data uji yang memiliki entitas.



Gambar 1.5 Diagram alir proses Rule Template

1.3.2 Pre-processing

Tahap ini merupakan tahapan awal yang pada intinya adalah mempersiapkan agar data latih dan data uji dapat diubah menjadi data yang lebih baik saat diproses ke dalam model pemrograman. Proses *pre-processing* ini ditunjukan pada Gambar 4.6.



Gambar 1.6 Diagram alir proses Pre-processing

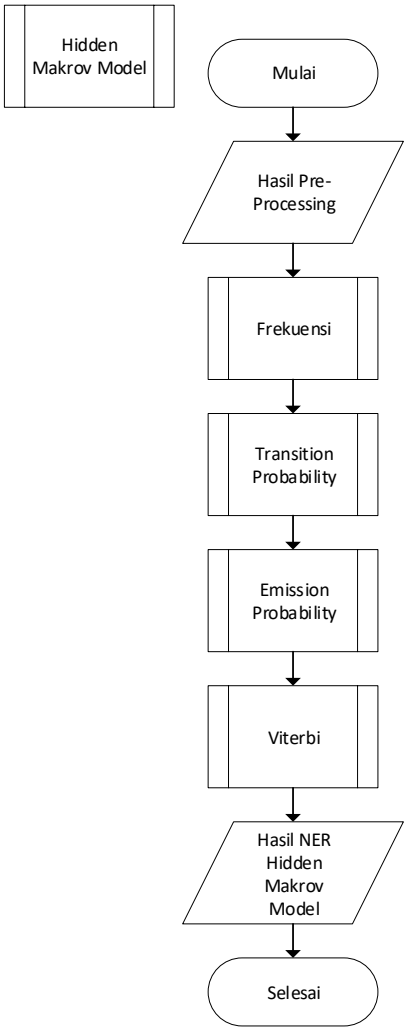
Berdasarkan Gambar 4.6 tahapan diagram alir proses *Pre-processing* sebagai berikut:

1. Memasukan dokumen data latih dan data uji.
2. Melakukan tokenisasi.

   Tokenisasi dimana proses tokenisasi adalah proses pemotongan *string* berdasarkan tiap kata pada data latih dan data uji. tokenisasi ini dapat berupa angka maupun huruf seperti kata ‘hp samsung s7’ menjadi ‘hp’ ‘samsung’ ‘s7’.
3. Menghasilkan kumpulan kata tunggal.

1.3.3 Hidden Markov Model

Proses *Hidden Markov Model (HMM)* terdiri dari beberapa proses dalam mengolah data latih. Data Latih yang sudah dilakukan pre-processing akan diolah dengan beberapa proses. Proses HMM terdiri dari proses frekuensi, proses *transition probability*, proses *emission probability* dan proses Viterbi. Proses HMM ini ditunjukan pada Gambar 4.7.



Gambar 1.7 Diagram alir proses Hidden Markov Model

Berdasarkan Gambar 4.7 tahapan diagram alir proses *Hidden Markov Model* sebagai berikut:

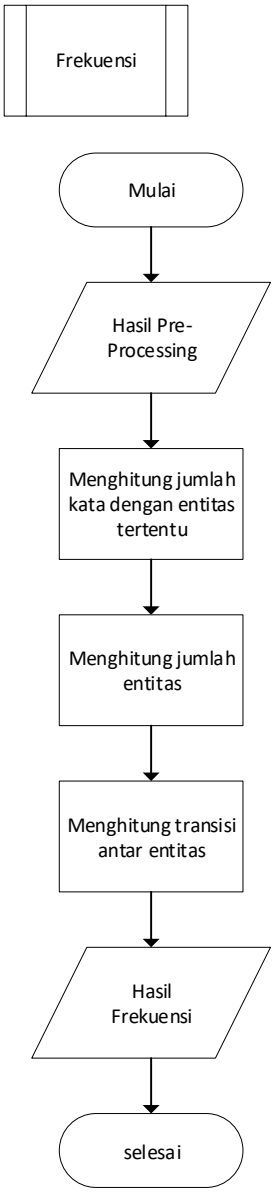
1. Memasukan hasil *pre-processing* data latih.
2. Melakukan proses frekuensi.
   Proses frekuensi dimana proses untuk menghitung data pada data dokumen latih, seperti menghitung banyaknya entitas, kata, transisi dan dilakukan penambahan entitas START pada awal kalimat dan END pada akhir kalimat pada data dokumen latih untuk dapat membantu dalam memudahkan perhitungan.
3. Melakukan proses *transition probability.*
   Proses *transition probability* merupakan proses perhitungan nilai yang didapatkan dari perpindahan *state* antar entitas. Pada proses ini proses *start probability* digabung dengan proses *transition probability* untuk membantu mempermudah perhitungan.
4. Melakukan proses *emission probability.*
   Proses *emission probability* merupakan proses perhitungan nilai suatu kata pada data latih yang memiliki entitas tertentu. perhitungan *emission probability* ini didapatkan dari frekuensi suatu kata dengan entitas dibagi dengan frekuensi kemunculan entitas tersebut.
5. Melakukan proses Viterbi.

Proses Viterbi merupakan proses perhitungan perkalian *transition probability* dengan *emission probability* yang didapatkan dari perhitungan data latih. Perhitungan ini dilakukan guna mencari nilai entitas yang optimal pada data dokumen uji.

6. Menghasilkan data dokumen uji yang sudah memiliki entitas.

1.3.3.1 Frekuensi

Langkah awal dalam melakukan permodelan HMM yaitu denga menghitung frekuensi. Proses ini menghitung frekuensi kemunculan pada data latih. Proses frekensi ini ditunjukan pada Gambar 4.8.



Gambar 1.8 Diagram alir proses frekuensi

Berdasarkan Gambar 4.8 beberapa tahapan diagram alir proses frekuensi sebagai berikut:

1. Memasukan hasil *pre-processing.*
2. Melakukan proses menghitung jumlah kata dengan entitas tertentu.

Proses ini adalah proses menghitung jumlah frekuensi kemunculan tiap kata pada data dokumen latih yang sudah memiliki entitas tertentu, seperti contoh ‘samsung<MEREK> A7<TIPE> dan<N_TAG> samsung<MEREK> A8<TIPE>’ sehingga mendapatkan frekuensi ‘samsung<MEREK>’ 2, ‘A7<TIPE>’ 1, ‘A8<TIPE>’ 1, dan ‘dan<N_TAG>’ 1.

3. Melakukan proses menghitung jumlah entitas.

   Proses ini adalah proses menghitung jumlah frekuensi kemunculan tiap entitas pada data dokumen latih.

4. Melakukan proses menghitung transisi antar entitas.

   Proses ini adalah proses menghitung jumlah transisi antar entitas pada data dokumen latih, seperti ‘samsung<MEREK> S7<TIPE> dan<N_TAG> apple<MEREK> iphone<TIPE>’ sehingga mendapatkan ‘MEREK ke TIPE 2 TIPE ke N_TAG 1 ‘.

5. Menghasilkan data frekuensi.

1.3.3.2 Transition probability

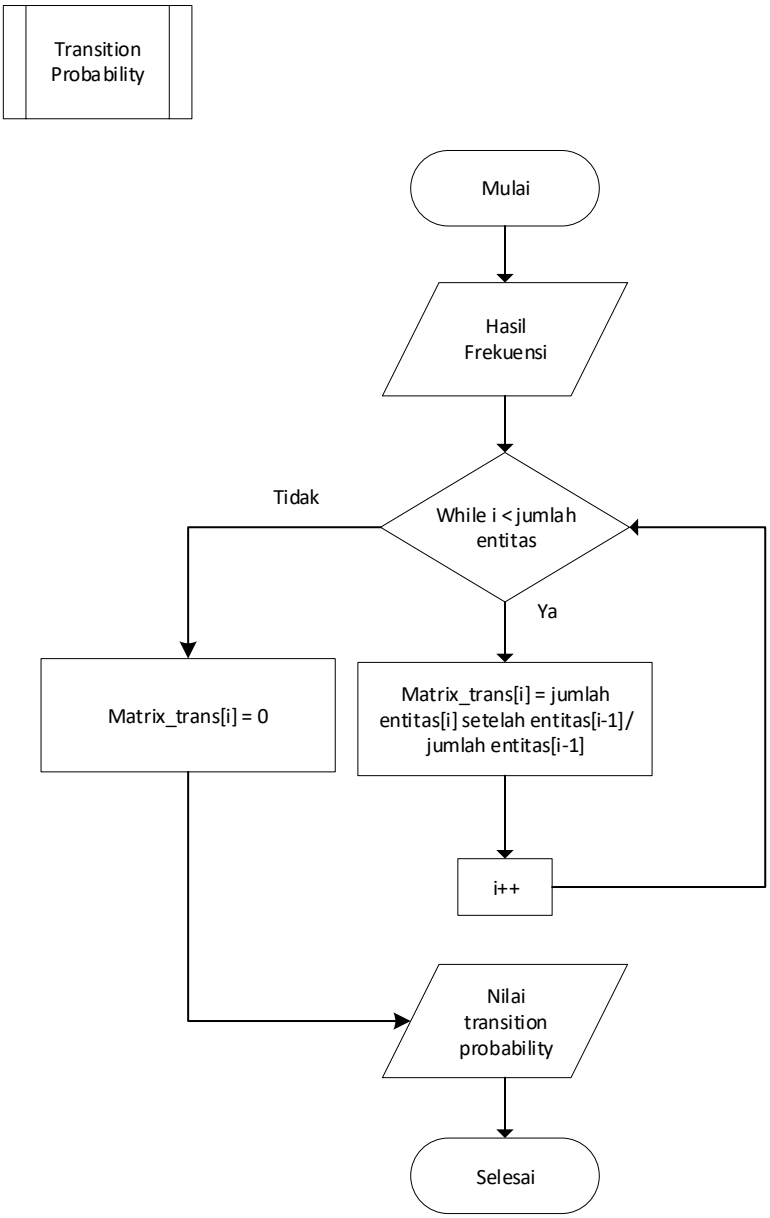
Proses *transition probability* ini dilakukan untuk menghitung nilai probabilitas *transisi state.* Proses *transition probability* ini ditunjukan pada Gambar 4.9.

Berdasarkan Gambar 4.9 beberapa tahapan diagram alir proses *transition probability* sebagai berikut:

1. Memasukan hasil frekuensi.

   Data frekuensi ini adalah data frekuensi entitas dan data transisi antar entitas.

2. Perulangan dimulai berdasarkan jumlah entitas.
3. Melakukan perhitungan *transition probability* dimana menghitung jumlah kemunculan entitas i setelah entitas i-1 dibagi dengan jumlah kemunculan entitas i-1.
4. Menghasilkan perhitungan *transition probability.*



Gambar 1.9 Diagram alir proses *transition probability*

1.3.3.3 Transition probability dengan additive smoothing

Sama seperti proses *transition probability* sebelumnya, proses ini menambahkan teknik smoothing dalam perhitungan. Proses *transition probability* dengan *additive smoothing* ini ditunjukan pada Gambar 4.10.

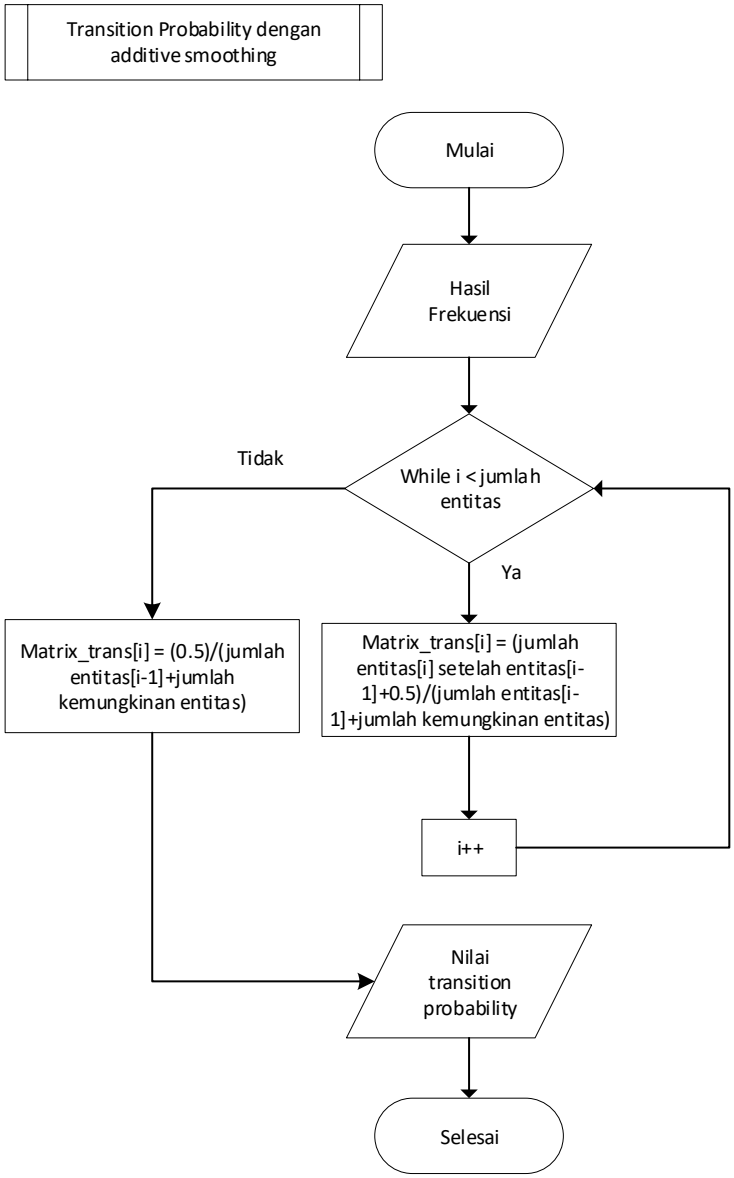
Berdasarkan Gambar 4.10 beberapa tahapan diagram alir proses *transition probability* dengan *additive smoothing* sebagai berikut:

1. Memasukan hasil *frekuensi.*

   Data frekuensi ini adalah data frekuensi entitas dan data transisi antar entitas.

2. Perulangan dimulai berdasarkan jumlah entitas.
3. Melakukan perhitungan *transition probability* dimana menghitung jumlah kemunculan entitas i setelah entitas i-1 + 0.5 dibagi dengan jumlah kemunculan entitas i-1 + jumlah kemungkinan entitas.

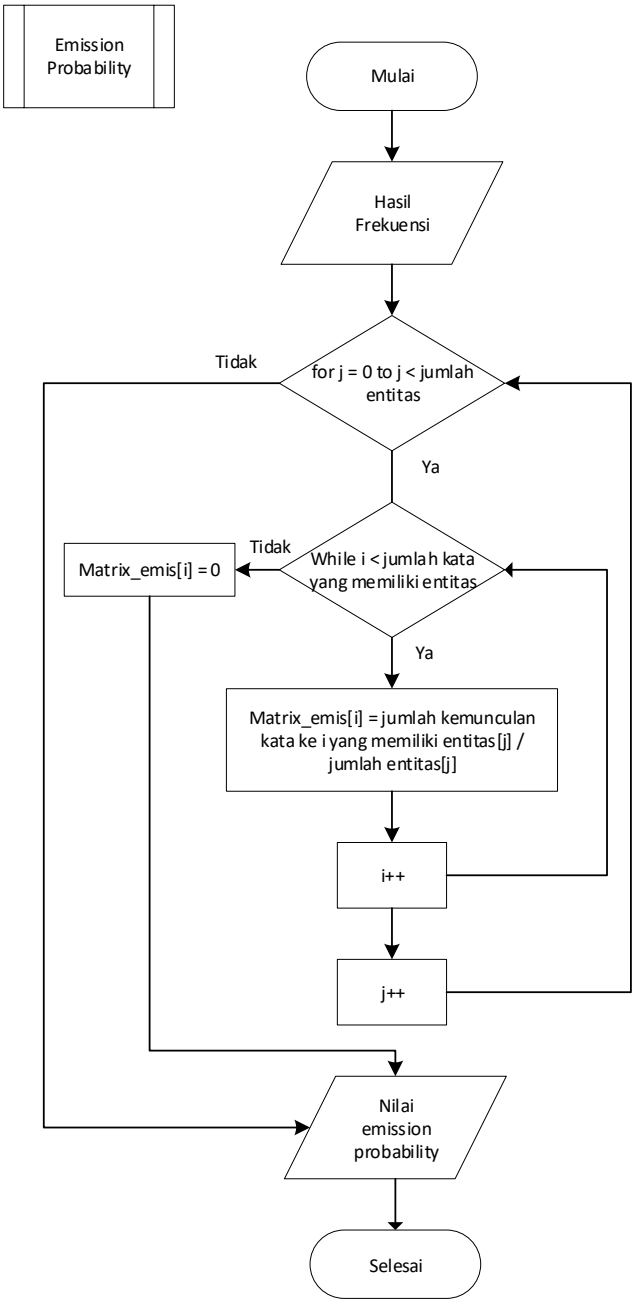
4. Menghasilkan perhitungan *transition probability.*



Gambar 1.10 Diagram alir proses *transition probability* dengan *smoothing*

1.3.3.4 Emission probability

Proses *emission probability* ini dilakukan untuk menghitung nilai probabilitas kata pada data dokumen latih dengan entitas tertentu. Proses *emission probability* ini ditunjukan pada Gambar 4.11.



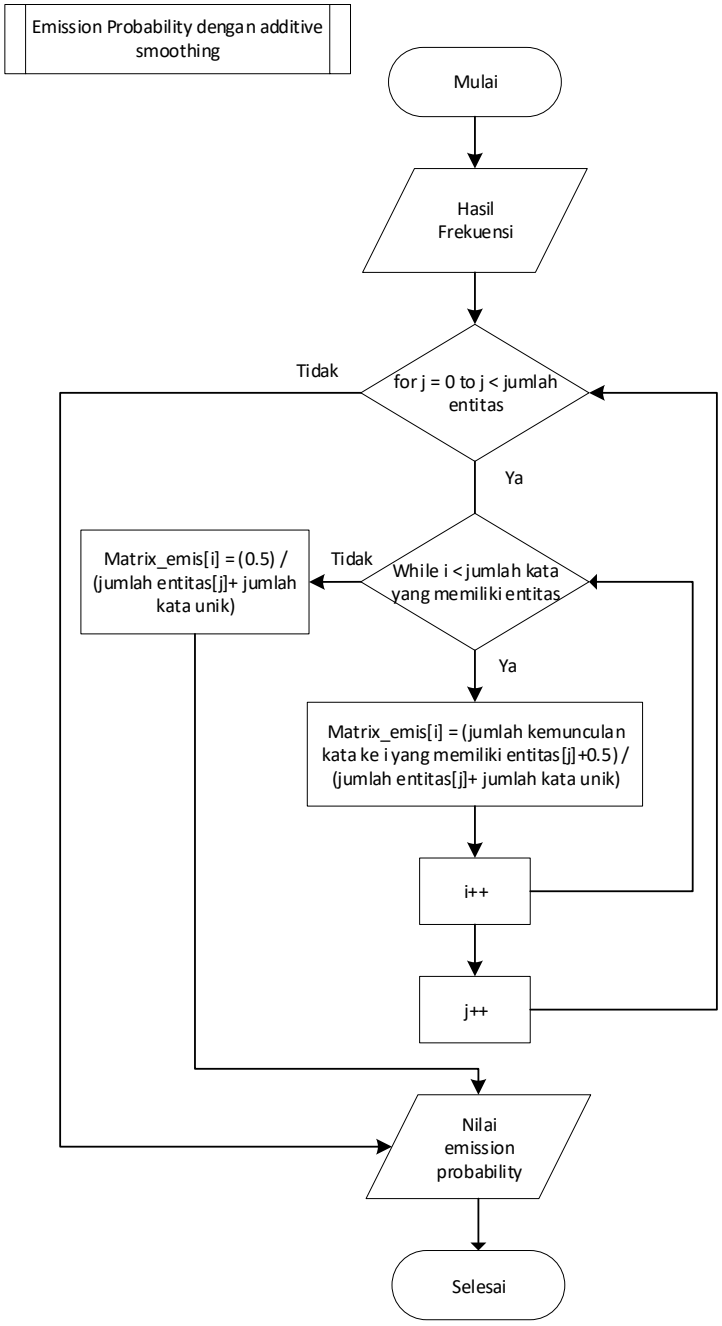
Gambar 1.11 Diagram alir proses *emission probability*

Berdasarkan Gambar 4.11 beberapa tahapan diagram alir proses *emission probability* sebagai berikut:

1. Memasukan hasil frekuensi yaitu frekuensi entitas dan data frekuensi kata dengan entitas tertentu.
2. Perulangan dimulai berdasarkan jumlah kata yang memiliki entitas.
3. Perulangan dimulai berdasarkan jumlah entitas.
4. Melakukan perhitungan *emission probability* dimana menghitung jumlah kemunculan kata ke i yang memiliki entitas j dibagi dengan jumlah entitas j.
5. Menghasilkan perhitungan *emission probability.*

1.3.3.5 Emission probability dengan additive smoothing

Sama seperti proses *emission probability sebelumnya*, proses ini menambahkan metode smoothing dalam perhitungan. Proses *emission probability dengan smoothing* ini ditunjukan pada Gambar 4.12.



Gambar 1.12 Diagram alir proses *emission probability* dengan *smoothing*

Berdasarkan Gambar 4.12 beberapa tahapan diagram alir proses *emission probability* dengan *additive smoothing* sebagai berikut:

1. Memasukan hasil frekuensi.

   Data frekuensi ini adalah data frekuensi entitas dan data frekuensi kata dengan entitas tertentu.

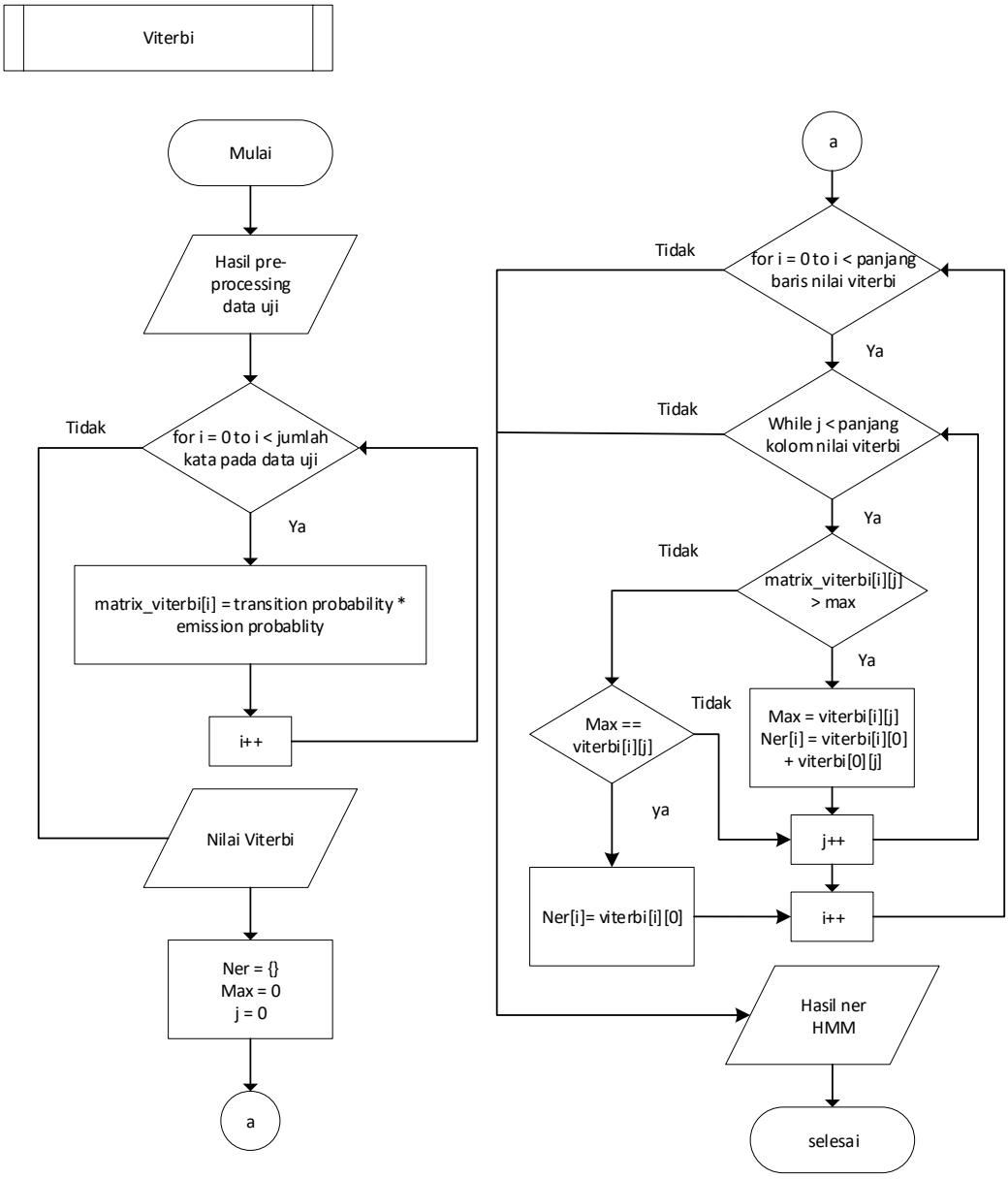
2. Perulangan dimulai berdasarkan jumlah kata yang memiliki entitas.
3. Perulangan dimulai berdasarkan jumlah entitas.
4. Melakukan perhitungan *emission probability* dimana menghitung jumlah kemunculan kata ke i yang memiliki entitas j ditambah 0.5 dibagi dengan jumlah entitas j ditambah dengan jumlah kata yang unik dimana jika terdapat kata 'samsung S7 dan samsung S8' maka 4 karena 'samsung' terhitung 1.
5. Menghasilkan perhitungan *emission probability.*

1.3.3.6 Viterbi

Proses Viterbi dilakukan untuk menghitung nilai probabilitas kata pada data dokumen uji terdapat pada data latih. Proses ini mencari urutan entitas terbaik atau optimum pada data dokumen uji. Proses Viterbi ini ditunjukan pada Gambar 4.13.

Berdasarkan Gambar 4.13 beberapa tahapan diagram alir proses Viterbi sebagai berikut:

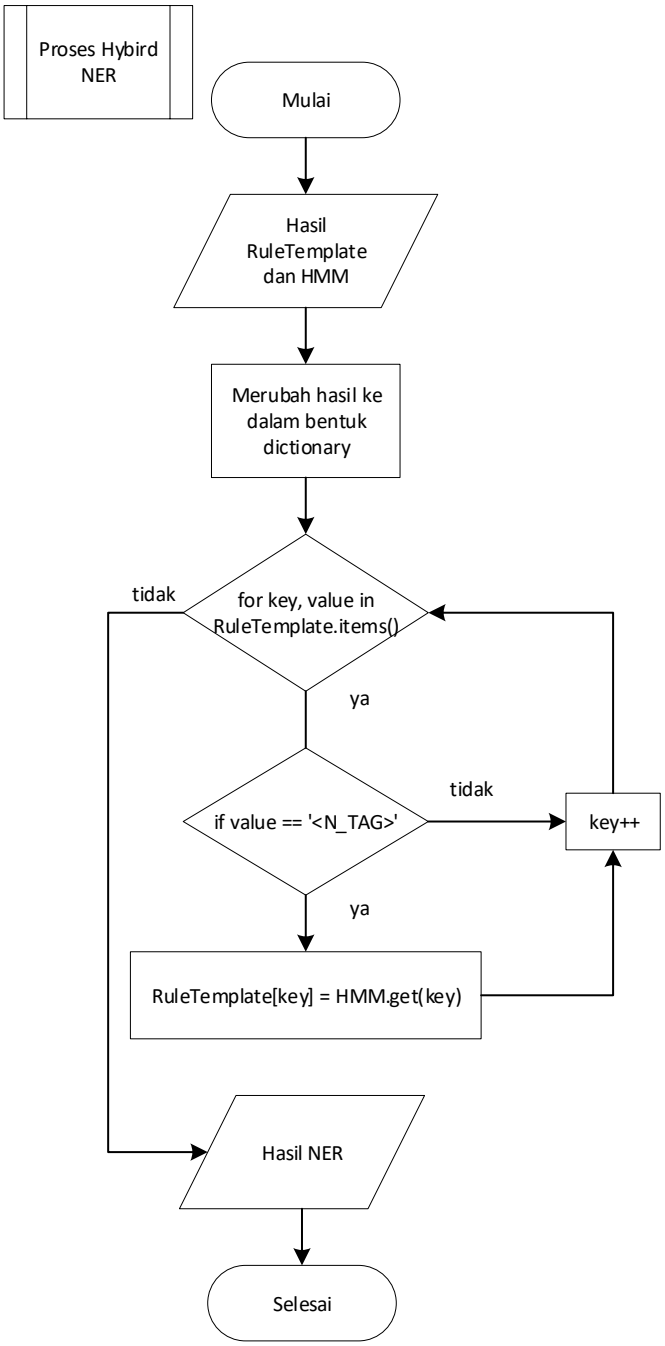
1. Memasukan hasil *pre-processing* data dokumen uji.
2. Perulangan dimulai berdasarkan jumlah kata pada data dokumen uji.
3. Melakukan perhitungan perkalian *transition probability* dengan *emission probability.*
4. Menghasilkan nilai Viterbi dalam bentuk *matriks* atau *list*.
5. Proses pembuatan *ner* dan mencari nilai *max* pada nilai Viterbi.
6. Perulangan dimulai berdasarkan panjang baris nilai Viterbi.
7. Perulangan dimulai berdasarkan panjang kolom nilai Viterbi.
8. Jika nilai Viterbi ke i lebih besar dari nilai *max* maka ner ke i adalah kata ke i dengan entitas ke j.
9. jika nilai *max* sama dengan nilai *viterbi* ke i maka *ner* ke i adalah kata ke i.
10. Menghasilan perhitungan HMM dengan Viterbi berupa data dokumen uji dengan entitas yang memiliki nilai maksimal atau optimum.



Gambar 1.13 Diagram alir proses Viterbi

1.3.4 Proses Hybird NER

Proses Hybird NER merupakan proses penggabungan hasil 2 metode yang sudah dilakukan sebelumnya, penggabungan ini terdiri dari penggabungan *Rule Template* dengan *Hidden Markov Model* (HMM) dan *Rule Template* dengan HMM *Additive smoothing*. Proses ini merupakan proses penggabungan entitas N_TAG pada *Rule Template* menjadi entitas yang dihasilkan oleh HMM berdasarkan kata tersebut. Proses Hybird NER ini ditunjukan pada Gambar 4.14.



Gambar 1.14 Diagram alir proses Hybird NER

Berdasarkan Gambar 4.14 beberapa tahapan diagram alir proses Hybird NER sebagai berikut:

1. Mengambil hasil dari *Rule Template* (RT) dan *Hidden Markov Model* (HMM).
2. Merubah hasil RT dan HMM ke bentuk *dictionary* dimana kata sebagai *key* dan *value* sebagai entitas.
3. Melakukan perulangan sebanyak *key* pada RT.
4. Jika nilai *value* pada RT bernilai <N_TAG> maka mengambil *value* atau entitas pada hasil HMM berdasarkan key atau kata tersebut.
5. Perulangan terjadi sebanyak key atau kata pada RT.

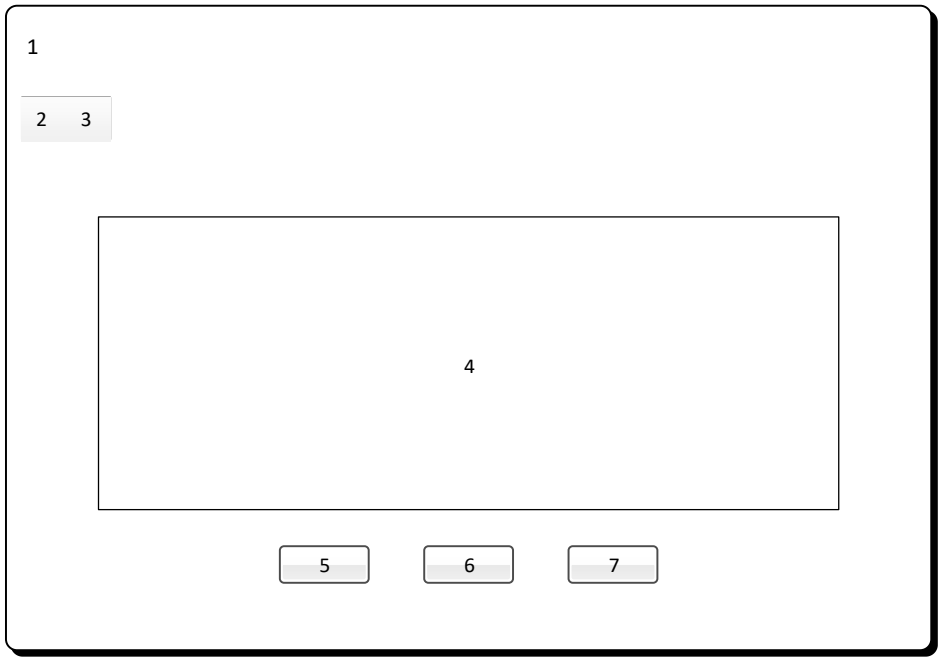
6. Menyimpan hasil dari penggabungan 2 metode.

1.4 Perancangan Antarmuka

Pada perancangan antarmuka terdiri dari 2 panel antarmuka yaitu antarmuka untuk masukan data dan antarmuka untuk menampilkan hasil. Gambar 4.15 menunjukan rancagan antarmuka untuk masukan data dan Gambar 4.16 menunjukan rancagan antarmuka untuk hasil.

Berikut ini merupakan penjelasan panel pada Gambar 4.15 antarmuka masukan data:

1. Judul aplikasi.
2. *Button* untuk beralih ke antarmuka masukan data.
3. *Button* untuk beralih ke antarmuka hasil.
4. *Field* yang menampilkan, mengubah data dokumen uji.
5. *Button* untuk membuka dan memilih dokumen uji.
6. *Button* untuk menyimpan data dokumen uji.
7. *Button* proses untuk memulai program.

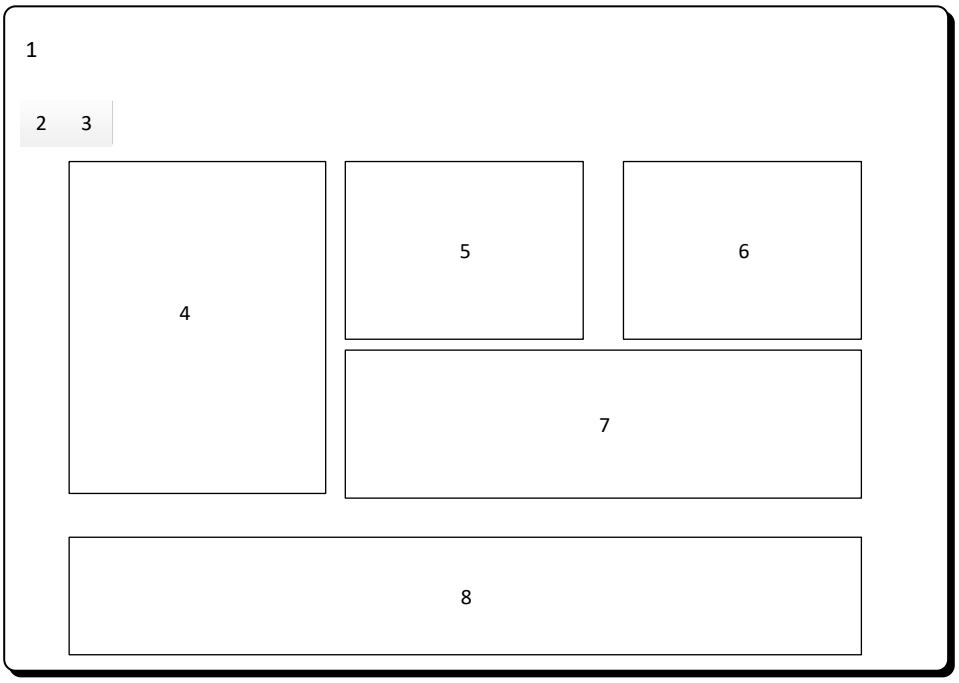


Gambar 1.15 Rancangan Antarmuka Masukan Data

Berikut ini merupakan penjelasan panel pada Gambar 4.16 antarmuka hasil:

1. Judul aplikasi.
2. *Button* untuk beralih ke masukan data.
3. *Button* untuk beralih ke antarmuka hasil.
4. *Field* yang menampilkan perhitungan frekuensi word.
5. *Field* yang menampilkan perhitungan frekuensi entitas.
6. *Field* yang menampilkan perhitungan frekuensi transisi.

7. *Field* yang menampilkan perhitungan probabilitas transisi, probabilitas emisi dan perhitungan data uji menggunakan algoritme Viterbi.
8. *Field* yang menampilkan hasil dari metode *Rule Template* dan *Hidden Markov Model.*



Gambar 1.16 Rancangan Antarmuka Hasil

1.5 Perancangan Pengujian

Perancangan pengujian digunakan untuk mengevaluasi kualitas pemberian entitas oleh sistem pada data uji. Pengujian ini dilakukan dengan menghitung *performance metrics* dan akurasi. Pengujian ini akan dilakukan dalam 5 skenario yaitu:

1. Pengujian *Rule Template*.
2. Pengujian *Hidden Markov Model.*
3. Pengujian *Hidden Markov Model* dengan penambahan *Additive smoothing*.
4. Pengujian *Rule Template* dan *Hidden Markov Model.*
5. Pengujian *Rule Template* dan *Hidden Markov Model* dengan penambahan *Additive smoothing.*