BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN APLIKASI

3.1 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Studi Literatur

   Dalam studi literatur yang dilakukan, dipelajari teori mengenai RNN, LSTM dalam analisis sentimen, penggunaan Word2Vec, F-*measure* dan k-*fold cross-validation.*

2. Pengumpulan Data

   Sebelum membuat sistem, terlebih dahulu dilakukan pengumpulan data berupa tweets menggunakan Twitter Streaming API. *Crawling* dilakukan sebanyak tiga kali pada tiga tanggal Debat Pilpres 2019 yang berbeda. Total *tweets* yang diperoleh secara keseluruhan berjumlah 1.786.213 dan disimpan dalam format *file* TXT. Potongan *source code* untuk *crawling data* dari Twitter dilihat pada Lampiran 1 dengan menggunakan *credentials* dari website Twitter Developer.

3. Pelabelan Data

   Pelabelan data berupa pemberian skor kepada setiap *tweet* dilakukan oleh pakar Bahasa Indonesia, yaitu Niknik Mediyawati, S.Pd., M.Hum. Sebanyak 1.000 data *tweets* dari 1.786.213 *tweets* akan diberi label secara manual oleh pakar dalam format *file excel*. Skor yang digunakan pada proses pelabelan data ini yaitu 0 (Netral), 1 (Jokowi), dan 2 (Prabowo). Hasil dari proses ini dapat dilihat pada potongan file EXCEL pada Lampiran 2.

4. Perancangan dan Pembuatan Sistem

   Karena menggunakan pendekatan RNN, maka input akan disimpan dalam bentuk matriks untuk training. Pada proses ini, akan dilakukan *word embedding* menggunakan Word2Vec untuk memperoleh kata-kata yang tertanam. Output dari Word2Vec akan terlihat seperti kamus dari setiap kata yang di-*map* dalam sebuah *array*. Hasil analisis sentimen dari tiap kandidat akan diproses sehingga menghasilkan perbandingan dalam bentuk persentase. Perbandingan tersebut akan digunakan untuk menunjukkan kandidat mana yang lebih dominan.

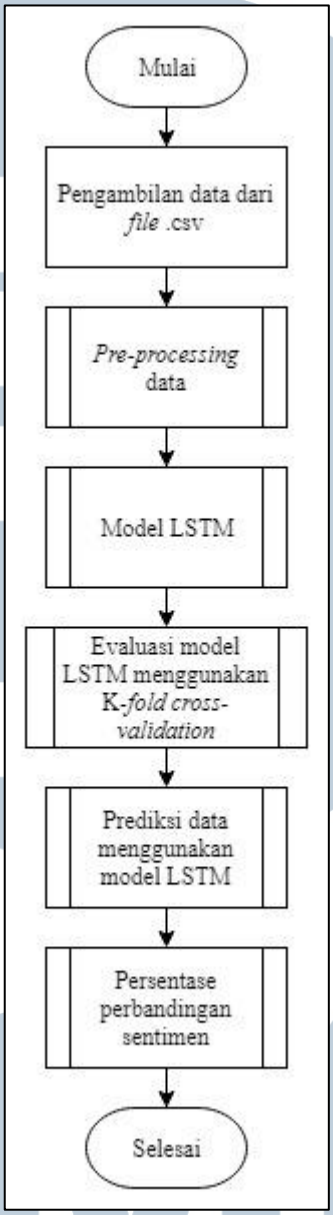
5. Uji Coba dan Evaluasi

   Uji coba dan evaluasi akan dilakukan untuk mengetahui performa F-*measure* metode *Long Short-Term Memory* pada *tweets* yang berkaitan dengan kandidat Presiden Indonesia 2019. *Tweets* yang telah diberi label akan diproses menggunakan K-*fold cross-validation*, yaitu sebuah metode statistik yang dapat digunakan untuk mengevaluasi model LSTM. Data akan dibagi menjadi dua bagian, yaitu data untuk *training* dan data untuk *testing* (prediksi). Setiap iterasi K-*fold cross-validation* akan menghasilkan nilai F-*measure*. Performa F-*measure* dari model LSTM yang telah dibuat akan diperoleh dari rata-rata nilai F-*measure* yang dihasilkan setiap iterasi.

3.2 Perancangan Aplikasi

Perancangan aplikasi dalam penelitian ini terdiri atas pembuatan *flowchart* sebagai berikut.

A. Flowchart Utama



Gambar 3.1 Flowchart Utama

Gambar 3.1 menunjukkan *flowchart* utama, dimulai dari proses pengambilan data yang telah diberi label oleh pakar dalam bentuk CSV. Data tersebut akan melewati tahap *pre-processing*, yang kemudian akan digunakan dalam model LSTM.

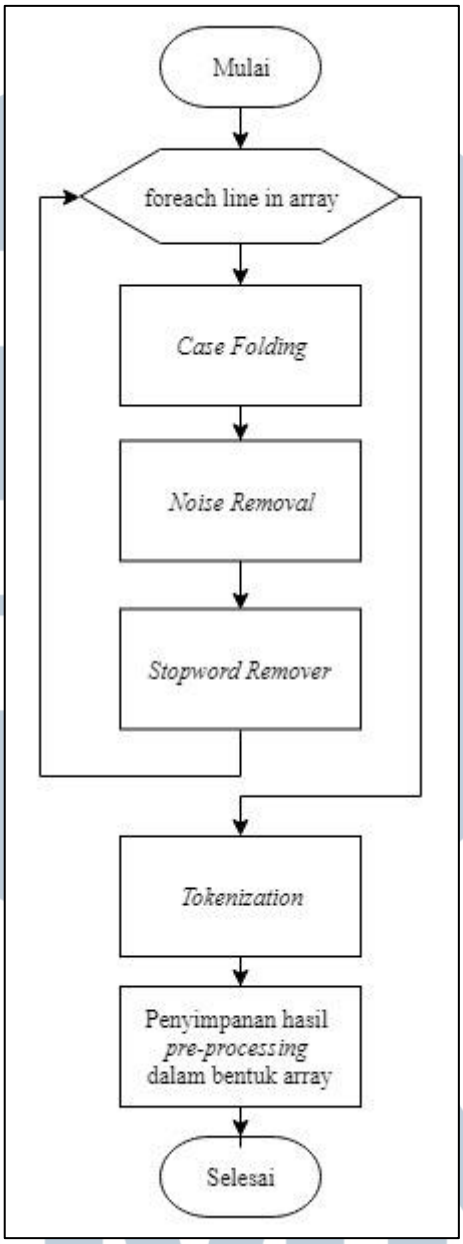
Model LSTM yang telah dibuat akan dievaluasi menggunakan K-*fold cross-validation* untuk mendapatkan hasil akhir berupa performa F-*measure*. Model

LSTM akan digunakan untuk memprediksi sentimen pada data sisanya sehingga mendapatkan persentase perbandingan sentimen pada tahap akhir.

B. Flowchart Pre-processing Data

Sesuai dengan Gambar 3.2, data yang telah diberi label akan dibaca per baris, dan setiap baris akan melewati proses *pre-processing* dalam empat tahap, yaitu *case folding*, *noise removal*, *stopword remover* dan *tokenization*. Beberapa tahap *pre-processing* dilakukan dengan menggunakan *regular expression* (*regex*). Salah satu contoh *pattern regex* untuk penghapusan URL pada data, yaitu "https:[^]+[ ]*". Kemudian data yang cocok dengan *pattern* tersebut akan di-*replace* dengan *empty string*.

Tahap pertama yaitu *case folding*, dimana seluruh data akan diubah menjadi *lowercase* dengan memanggil function lower(). Selanjutnya *noise removal* menggunakan *regex* untuk beberapa proses, seperti menghapus URL pada data, menghapus emoji "\\\\u.{4}", menghapus kata yang berkaitan dengan *retweet* "rt @[^:]+:", menghapus simbol *new line* dan simbol petik dua "\\\\n dan \\\\"", menghapus *username* dan simbol '@' yang tidak termasuk *keyword* tertentu "@(?!(?:jokowi|prabowo|sandiuno|gerindra|kpu|aniesbaswedan|bawaslu|fadlizon|fahrihamzah|dahnilanzar|psi|pdip|nasdem))\w+", menghapus karakter spesial yang tersisa pada data "[^a-zA-Z0-9\s]", serta membuang *whitespace* berlebih "\s+". Selanjutnya, *stopword remover* digunakan untuk menghilangkan kata sambung, kata depan dan kata-kata yang tidak memiliki makna. Proses ini menggunakan *library* python yaitu PySastrawi. Setelah proses *looping* untuk ketiga tahap diatas, *array of data* tersebut akan diproses ke tahap terakhir yaitu *tokenization*.

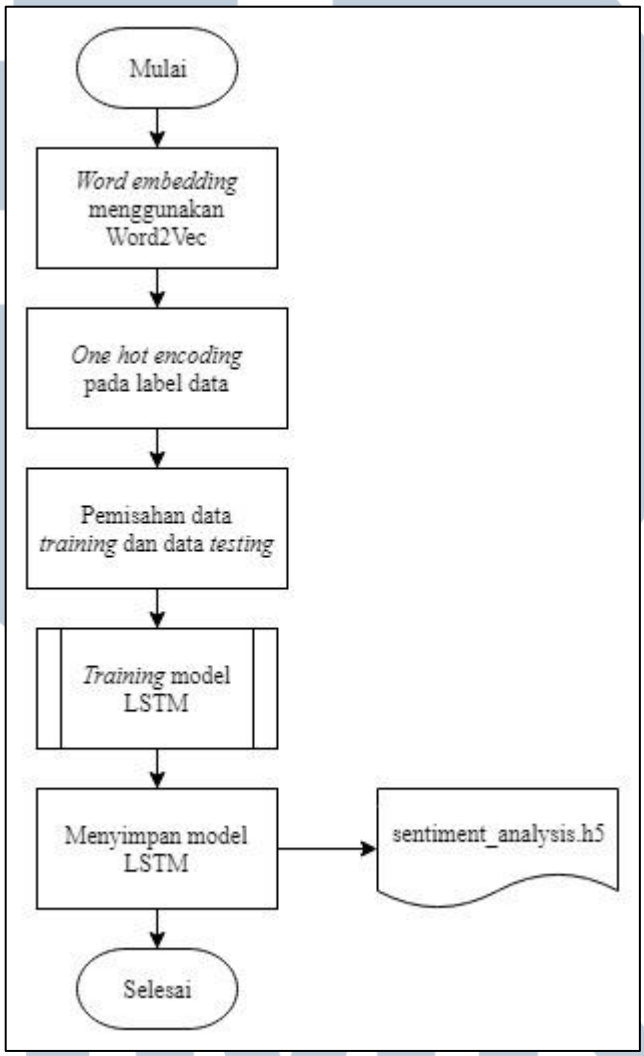


Gambar 3.2 Flowchart Pre-Processing Data

C. Flowchart Model LSTM

Gambar 3.3 menunjukkan proses perancangan dan pembangunan model LSTM yang dimulai dengan proses *word embedding*. Word2Vec merupakan algoritma *word embedding* yang akan memetakan kata menjadi bentuk vektor. Label data yang berupa sentimen akan melewati proses *one hot encoding*, yaitu proses konversi variabel kategorikal menjadi bahasa yang dikenal oleh algoritma *machine learning*. *Training* model LSTM yang telah selesai berjalan akan

menghasilkan sebuah model LSTM yang akan disimpan dalam "sentiment_analysis.h5". *File* H5 tersebut dapat di-*load* kembali untuk menggunakan model yang tersimpan didalamnya.

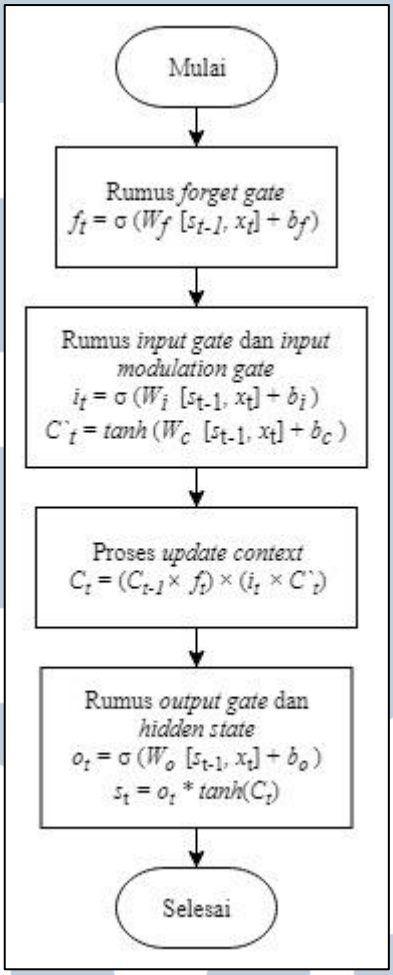


Gambar 3.3 Flowchart Model LSTM

D. Flowchart Training Model LSTM

Gambar 3.4 merupakan proses yang terjadi untuk menghasilkan model LSTM dalam bentuk rumus. Tahap pertama yaitu rumus *forget gate* (f_t) yang akan membaca nilai *state* sebelumnya (s_{t-1}) dan *input* (x_t) sehingga menghasilkan angka antara 0 dan 1 untuk setiap elemen dalam C_{t-1}. Selanjutnya, rumus *input gate* (i_t)

merupakan fungsi sigmoid dengan range 0 hingga 1, dan rumus *input modulation gate* ($C`_t$) memiliki fungsi aktivasi tanh dengan *range* -1 hingga 1.



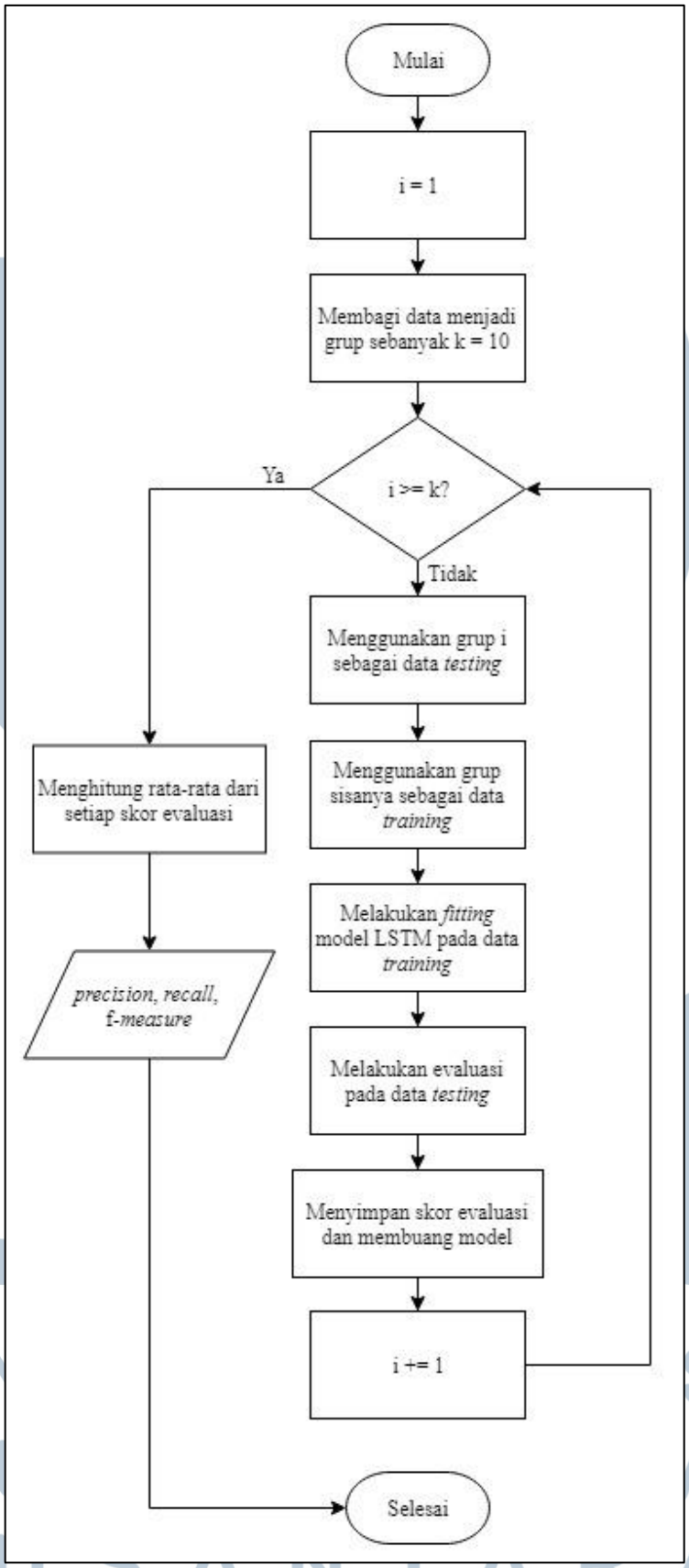
Gambar 3.4 Flowchart Training Model LSTM

Proses selanjutnya yaitu proses *update context* dari C_{t-1} menjadi C_t. Konteks lama (C_{t-1}) akan dikalikan dengan hasil *forget gate* untuk menentukan hasil apa saja yang akan dilupakan. Kemudian kandidat konteks baru ($C`_t$) akan dikalikan dengan hasil *input gate* untuk menentukan seberaa banyak kandidat konteks baru yang akan disertakan. Hasil dari kedua perkalian tersebut akan menghasilkan konteks baru yaitu C_t. Pada tahap akhir, *state* sebelumnya (s_{t-1}) dan *input* (x_t) akan melewati gerbang sigmoid sehingga menghasilkan *output gate*. Hasil dari proses *update context* (C_t) akan melalui fungsi aktivasi tanh untuk memiliki range nilai dari -1

hingga 1, kemudian akan dikalikan dengan hasil *output gate* untuk mendapatkan *hidden state* (s_t).

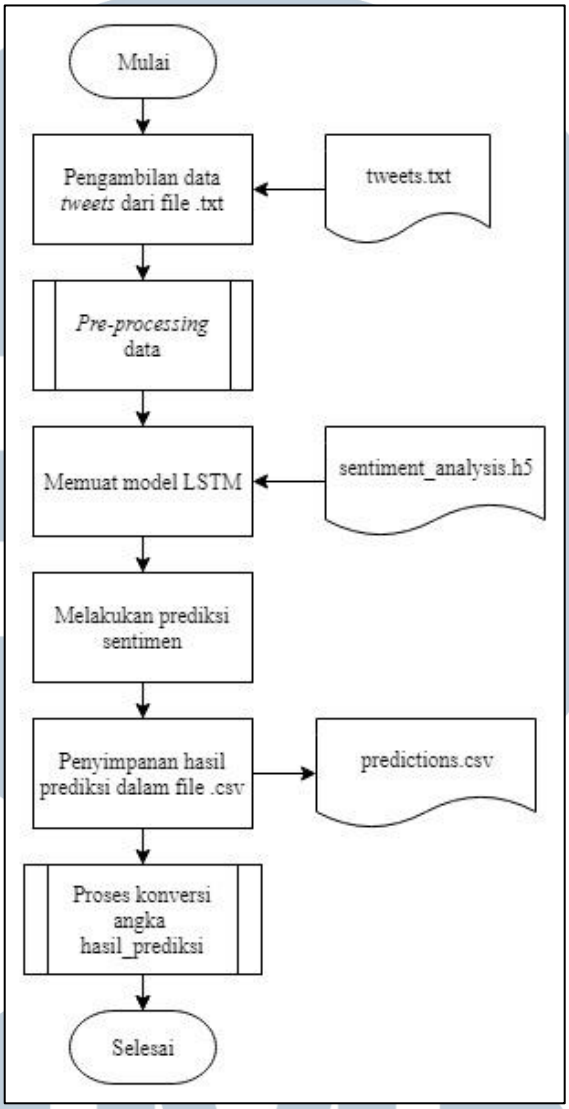
E. Flowchart Evaluasi Model LSTM Menggunakan K-fold Cross-validation

Berdasarkan *flowchart* yang terdapat pada Gambar 3.5, evaluasi model LSTM akan dilakukan menggunakan K-*fold cross-validation*. Data akan dibagi menjadi beberapa kelompok sesuai dengan nilai k, yaitu 10 dan evaluasi akan dilakukan sebanyak k iterasi. Dalam setiap iterasi, data pada kelompok k akan digunakan sebagai data *testing*, dan data sisanya akan digunakan sebagai data *training*. Selanjutnya setelah melewati proses *fitting* model LSTM pada data *training*, evaluasi akan dilakukan pada data *testing* sehingga menghasilkan skor evaluasi. Skor evaluasi dalam 4 metriks (*accuracy*, *precision*, *recall*, *F-measure*) akan disimpan, dan model LSTM tersebut akan dibuang. Tahap tersebut akan diulang hingga nilai iterasi sama dengan nilai k. Hasil evaluasi akhir akan didapatkan dengan menghitung rata-rata akhir untuk keempat metriks.



Gambar 3.5 Flowchart Evaluasi Model LSTM Menggunakan K-fold Cross-validation

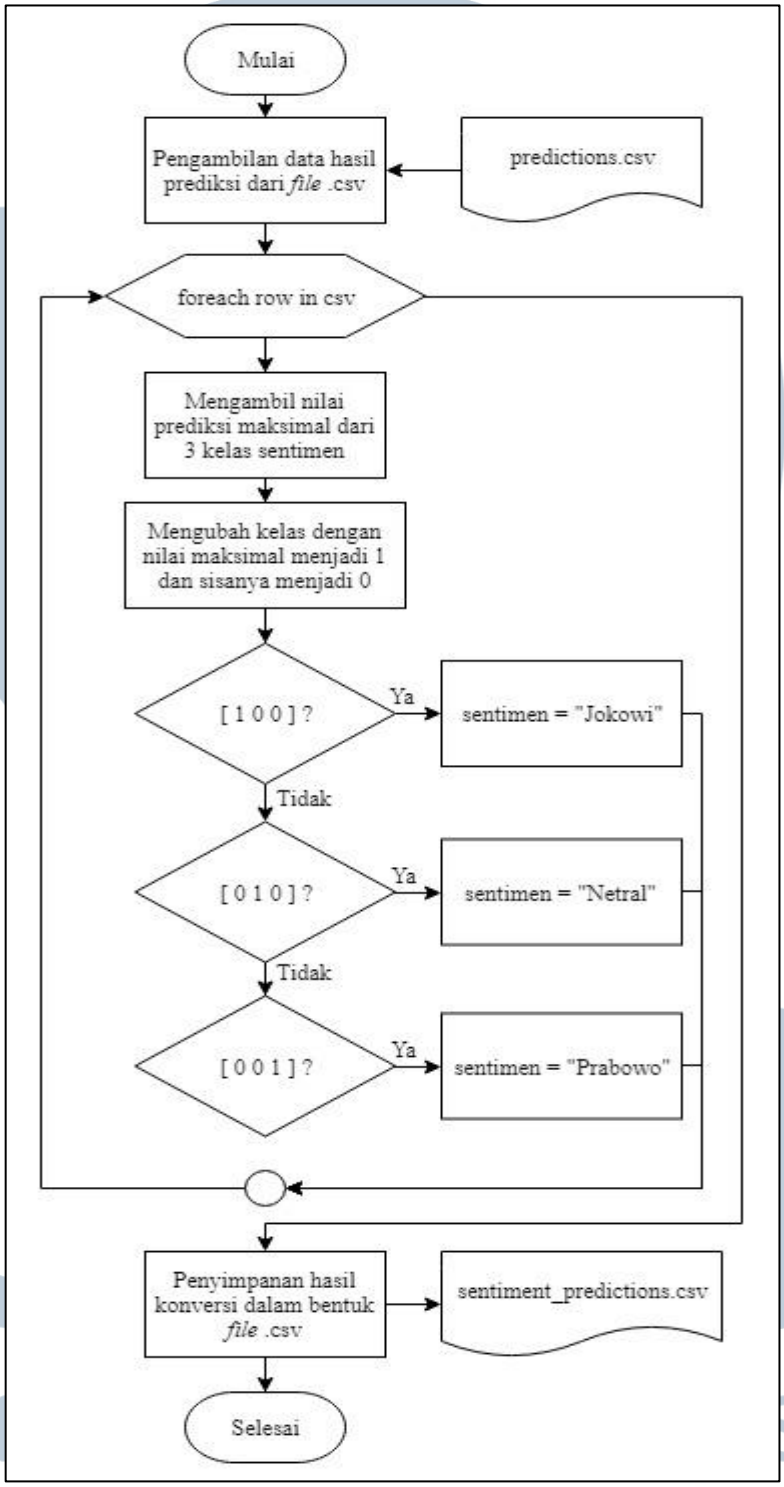
F. Flowchart Prediksi Data Menggunakan Model LSTM



Gambar 3.6 Flowchart Prediksi Data Menggunakan Model LSTM

Gambar 3.6 menunjukkan proses pengambilan data yang masih *raw* dari *file* TXT. Data berupa *tweets* lainnya yang belum dilabel akan melewati proses *pre-processing* terlebih dahulu. Model LSTM yang telah disimpan dalam bentuk *file* H5 akan dimuat kembali untuk melakukan prediksi sentimen. Setelah proses prediksi sentimen, hasil_prediksi akan dihasilkan dalam bentuk angka yang kemudian akan disimpan dalam file CSV. Proses konversi pada tahap terakhir akan dilakukan untuk mengubah angka menjadi sentimen.

G. Flowchart Proses Konversi Angka hasil_prediksi

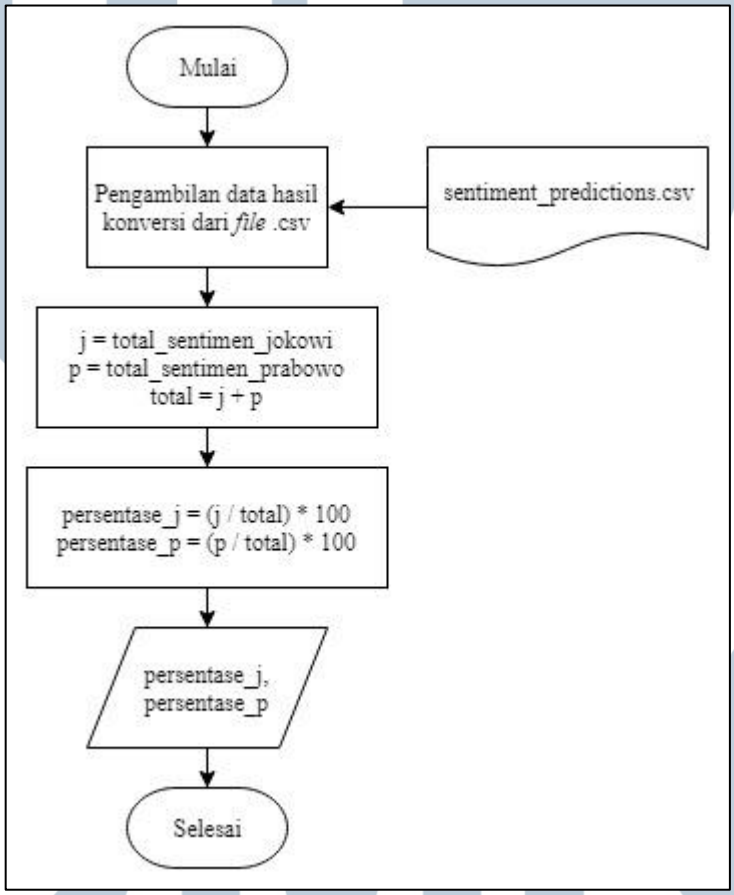


Gambar 3.7 Flowchart Proses Konversi Angka hasil_prediksi

Sesuai dengan Gambar 3.7, data hasil_prediksi akan dimuat dari *file* CSV sebelumnya. Data tersebut akan melewati sejumlah proses selama baris dalam file

CSV tersebut masih ada. Proses konversi dimulai dari penentuan nilai maksimal dari tiga kelas sentimen. Kelas dengan nilai terbesar akan diubah menjadi 1 dan dua kelas sisanya akan diubah menjadi 0. Apabila nilainya [1 0 0] maka akan dikonversi menjadi “Jokowi”, nilai [0 1 0] akan dikonversi menjadi “Netral” dan nilai [0 0 1] akan dikonversi menjadi “Prabowo”.

H. Flowchart Persentase Perbandingan Sentimen



Gambar 3.8 Flowchart Persentase Perbandingan Sentimen

Proses perhitungan untuk mendapatkan persentasi perbandingan sentimen dapat dilihat pada Gambar 3.8. *File* CSV yang berisi data hasil konversi sebelumnya akan dimuat terlebih dahulu. Sentimen yang berlabel “Jokowi” akan dijumlahkan semua menjadi total_sentimen_jokowi dan sentimen berlabel "Prabowo" akan dijumlahkan semua menjadi total_sentimen_prabowo. Nilai total

keseluruhan dari data akan didapatkan dari hasil penjumlahan total_sentimen_jokowi dan total_sentimen_prabowo. Persentase sentiken “Jokowi” didapatkan dengan membagi total_sentimen_jokowi dengan total keseluruhan, kemudian dikalikan dengan 100. Sedangkan persentase sentimen “Prabowo” didapatkan dengan membagi total_sentimen_prabowo dengan total keseluruhan, kemudian dikalikan dengan 100.