BAB 3

METODOLOGI DAN PERANCANGAN SISTEM

3.1. Metodologi Penelitian

Metode penelitian yang dilakukan untuk mengimplementasi penelitian ini antara lain sebagai berikut.

a. Telaah literatur

Pada tahap telaah literatur, proses penelitian dimulai dari mempelajari teori-teori yang dibutuhkan dan analisis dari berbagai karya ilmiah, buku, jurnal dan *website* yang pernah diterbitkan oleh peneliti lainnya. Teori-teori yang dipelajari meliputi *Natural Language Processing*, *supervised machine learning, Recurrent Neural Network*, *Long Short-Term Memory,* FastText dan *softmax activation*

b. Analisis Kebutuhan dan Perancangan Sistem

Pada tahap ini, teori yang dipelajari sudah cukup untuk melakukan proses analisis kebutuhan dan perancangan sistem untuk penelitian. Analisis kebutuhan meliputi berbagai kebutuhan agar dapat mengeksekusi penelitian yang meliputi kebutuhan dataset dan juga fungsi-fungsi yang dapat menghasilkan penelitian yang

optimal. Selain itu proses analisis kebutuhan juga meliputi alat penunjang penelitian yang akan digunakan untuk membuat perancangan kebutuhan.

Setelah mendapatkan hasil analisis, maka proses dilanjutkan dengan membuat rancangan sistem yang akan menjadi hasil akhir penelitian. Proses-proses tersebut meliputi pembuatan modul-modul yang memiliki fungsi untuk membuat *model machine learning* dan juga mengoptimalisasi *model* tersebut, rancangan antarmuka sistem dan alur kerja pembuatan model hingga dapat diakses melalui sistem.

c. Implementasi

Dalam tahap ini, proses implementasi metode LSTM-RNN yang telah dirancang akan dibuat fungsi-fungsi dan kebutuhan data untuk menopang pembuatan *model machine learning* yang mengimplementasi algoritma. Selain itu, setelah proses pembuatan *model* telah mendapatkan hasil yang optimal berupa tingkat akurasi terbaik, maka dilanjutkan dengan intergrasi *model* dengan sistem dalam bentuk *web* yang dapat diakses secara lokal. Tahap ini akan dibahas pada Bab 4. Modul-modul utama juga dibuat pada Bab 4

d. Uji Coba dan Evaluasi

Proses uji coba dilakukan melalui proses menentukan *hyperparameter* yang tepat agar menghasilkan tingkat akurasi yang paling optimal. Dengan memberi variasi *hyperparameter* yang berdampak dengan performa *model* dalam memprediksi hasil klasifikasi. Landasan dari pemberian variasi *hyperparameter* pada *model* berasal dari berbagai *paper* dan *trial error.* Evaluasi dilakukan dengan mengukur tingkat akurasi model memberikan hasil

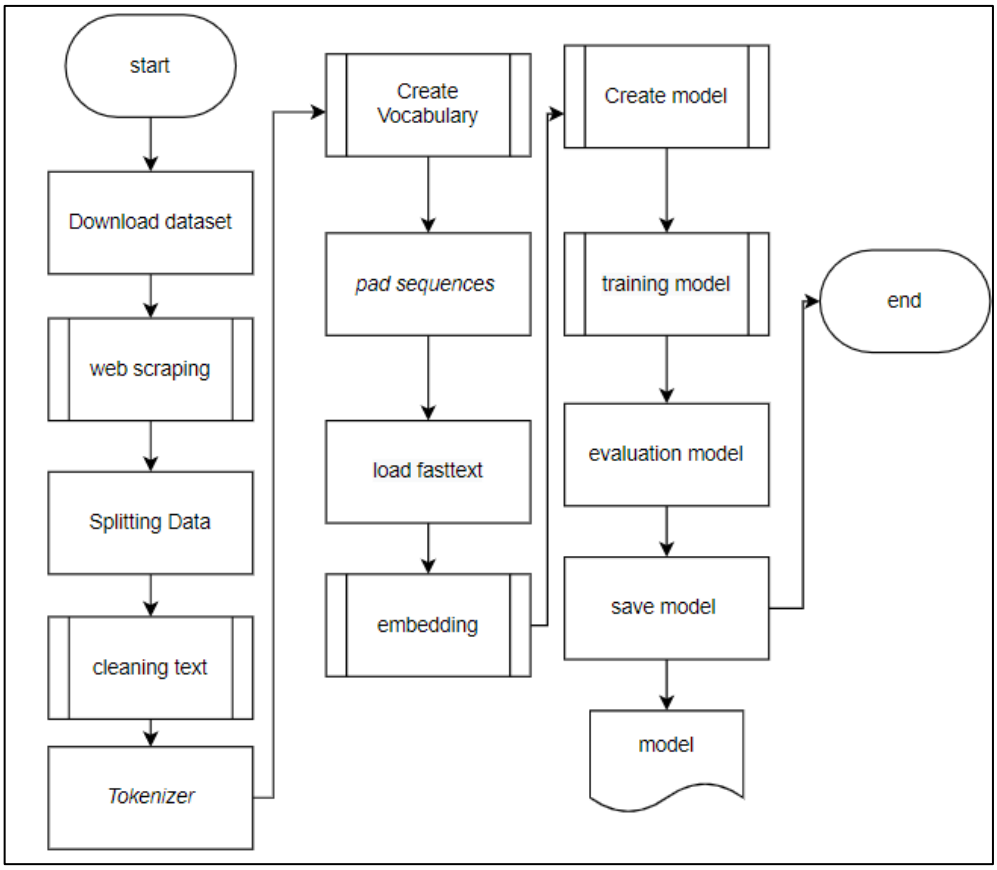
prediksi yang tepat mengenai kategori berita. Uji coba dan evaluasi akan dijabarkan pada Bab 4.

3.2. Analisis dan Alur Kerja

Setelah riset dari berbagai *paper* dan karya ilmiah lainnya. Penelitian ini memiliki berbagai kebutuhan agar menghasilkan *model* dengan performa yang optimal yaitu :

1. Kumpulan berita Indonesia beserta kategori yang didapat dari Jakartaresearch dan *web scraping*
2. Berita-berita yang sudah didapat perlu dilakukan proses pembersihan teks dan menerapkan *Natural Language Processing* agar *model* dapat melakukan prediksi dengan akurat
3. *Library-library* yang diimplementasi dengan tepat dalam proses *pre-processing* hingga pembentukan *model*.

Proses-proses pemenuhan kebutuhan tersebut tertuang pada gambar *flowchart* 3.1 yang akan menjelaskan tahapan-tahapan hingga proses implementasi.



Gambar 3.1 *Flowchart* Penelitian

Proses pencarian *dataset* di internet dan menemukan sebuah API dari Jakartaresearch yang memiliki *dataset* berita Indonesia yang terdiri dari lima kategori yaitu *news*, bola, tekno, otomotif dan bisnis. Namun, *dataset* tersebut memiliki penyebaran data yang kurang berimbang dimana kategori tekno dan otomotif memiliki jumlah berita yang sedikit. Dengan masalah tersebut, maka dilakukan proses *web scraping* pada kanal Kompas dengan mengambil berita dari kategori tekno dan otomotif yang masing-masing berjumlah 1600 berita dan 2000 berita. Setelah kebutuhan data dipenuhi maka proses pembagian data menjadi data latih dan data *testing.* Menurut Jalil dkk(2014) Pembagian data dengan komposisi 75% data latih dan 25% data *testing* dengan pertimbangan ukuran dataset yang cukup besar adalah salah satu alternatif. Selanjutnya akan dilanjutkan dengan

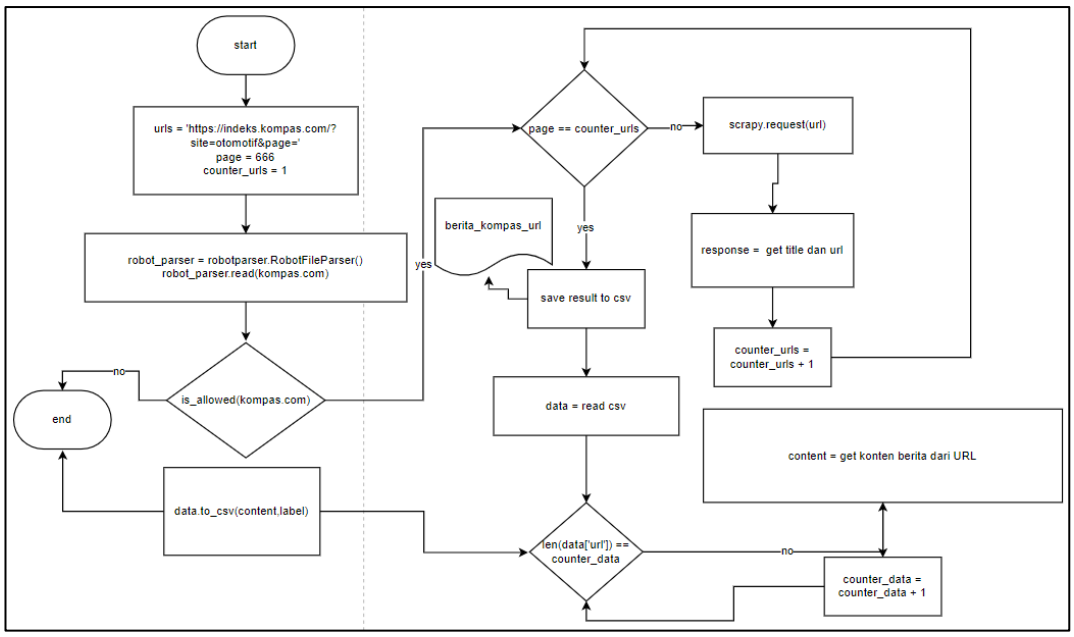
menerapakan ilmu *Natural Language Processing* pada setiap paragraf pada data-data berita. Selanjutnya dalam membantu pembuatan *embedding layer* yang memiliki fungsi untuk memberi *weight* pada setiap kata yang akan jadi masukan pada *model*, maka digunakan salah satu *pre trained model* yaitu FastText. *Pre trained model* tersebut akan memberi *weight* dari kumpulan *token-token* sesuai dengan nilai yang telah disimpan pada FastText. Setelah proses *pre-processing* dan FastText sukses diimplementasi maka dilanjutkan ke tahap pembuatan *model*. Tahap pembuatan *model* akan menggunakan *library* Keras Tensorflow dengan mengimplementasi fungsi Sequential yang memiliki fungsi untuk membuat *layer-layer* pada *neural network* secara berurutan. Pada penelitian ini, *layer-layer* tersebut meliputi satu *embedding layer,* satu *spatial dropout layer*, satu LSTM *layer* dan satu *dense layer*. Setelah proses pembuatan *model* telah selesai. *Model* tersebut akan dilatih menggunakan data latih yang sudah bersih dan akan dilakukan secara berulang hingga mendapatkan nilai akurasi, presisi, *recall*, *f1-score* paling optimal dengan nilai *loss* yang minimum. *Model* yang terbentuk akan disimpan dan diimpelementasi ke sistem *web* yang telah dibuat menggunakan *framework* Flask 1.1.2 untuk membuat *model* dapat melakukan prediksi secara *real-time* dan *framework* Laravel 7 untuk web.

3.3. Perancangan *Model*

Dalam pembuatan *model* agar dapat melakukan prediksi dengan tepat, maka ada langkah-langkah yang perlu dilakukan sebelum *model* tersebut dapat dilatih dengan data yang tepat yaitu dengan melalui tahap pengambilan data, *pre-processing* data dan pembuatan *model*

3.3.1. Pengambilan Data

Proses awal dalam pembuatan *model machine learning* adalah memenuhi kebutuhan data yang berguna untuk melatih *model* agar dapat melakukan prediksi sesuai data-data yang ada. Proses pengambilan data dibagi menjadi dua yaitu pengambilan data dari API dan pengambilan data menggunakan metode *web scraping* yang menggunakan *framework* Scrapy dan *library* Urlib. Pengambilan data melalui API berasal dari Jakartaresearch yang didapat dari kanal Liputan6.com dengan jumlah 9.727 berita. Kategori – kategori pada berita tersebut yaitu bola, *news*, bisnis, tekno dan otomotif. Selanjutnya agar memiliki data yang seimbang, maka dilakukan proses pengambilan data menggunakan metode *web scraping* pada kanal Kompas. Penjelasan langkah-langkah *web scraping* terdapat pada gambar 3.2



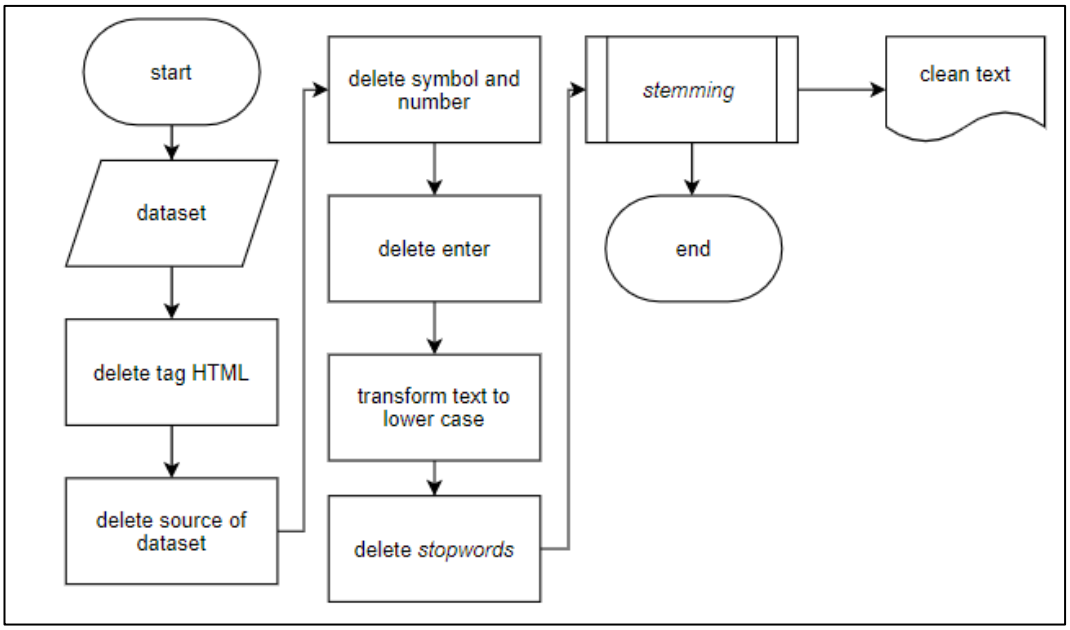
Gambar 3.2 Proses *Web Scraping*

Pada gambar 3.2, proses diawali dengan menentukan *website* Kompas beserta jumlah halaman yang akan diambil *url* beritanya dan disimpan kedalam

sebuah *file* csv dengan nama berita_kompas_url. Sebelum proses *scraping* dilakukan pengecekan pada website tersebut melalui robots.txt. Pengecekan tersebut bertujuan untuk mengetahui apakah kita boleh melakukan *scraping* pada website tersebut dan hal-hal apa saja yang tidak boleh diambil dari website tersebut. Jika pengecekan website memberikan status bahwa diperbolehkan untuk *scraping*, maka proses dilanjutkan dengan mengambil *url* dari tiap judul berita yang ada dan disimpan. Setelah mendapatkan seluruh *url* dari berita-berita yang ada, *file* yang berisi *url* tersebut akan dibaca dan dimasukan pada *variable* data. *Variable* yang memiliki data tersebut memiliki kolom *url* yang akan diambil sebagai teks yang akan digunakan oleh Urlib. Setelah Urlib sukses mengambil teks maka dilakukan proses pembersihan dari *tag* html dan disimpan pada *file* berekstensi csv dengan nama kompas_content. Proses pengambilan data berakhir dan dapat dilanjutkan ke *pre-processing*.

3.3.2. Pre Processing

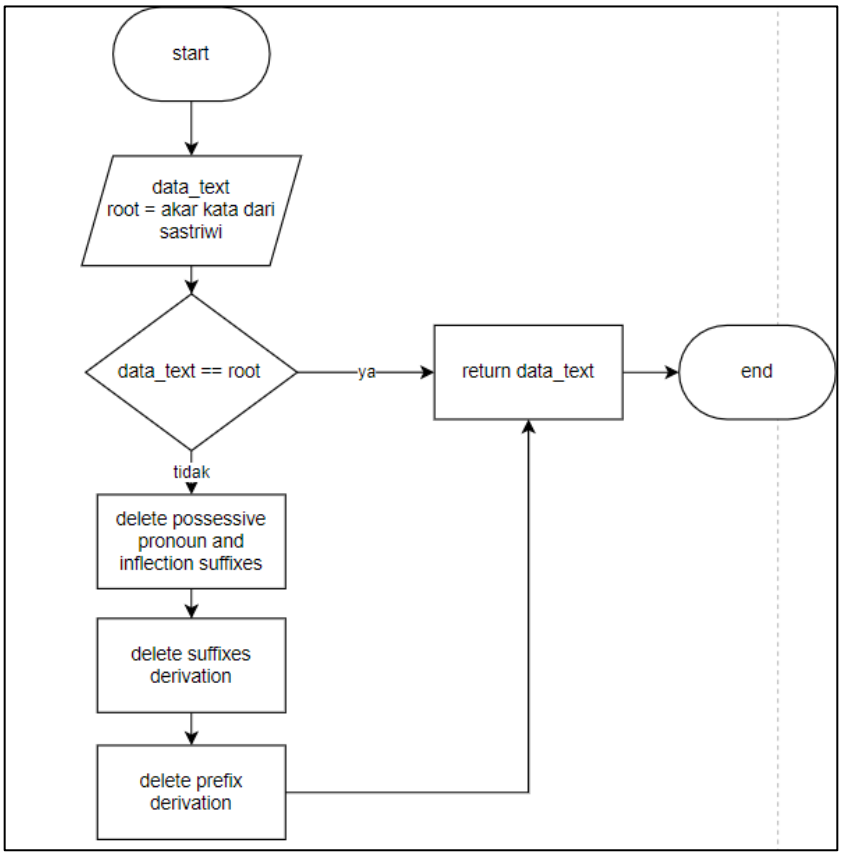
Pada tahap ini, seluruh data sudah didapatkan dan akan melalui berbagai proses seperti *cleaning text*, *stemming*, *tokenizing*, *pad sequencing* dan implementasi FastText. Langkah-langkah pembersihan teks terdapat pada gambar 3.3



Gambar 3.3 *Flowchart cleaning text*

Persiapkan *dataset* yang ada dan dideklarasi pada suatu *variable*. Setelah itu data tersebut diawali dengan menghapus berbagai *tag* HTML seperti <p>, <b> dan sebagainya. Penghapusan *tag* bertujuan agar teks-teks yang tidak memuat informasi agar dibuang karena tidak memiliki fungsi untuk memberi makna dari sebuah kalimat ataupun kata. Proses dilanjutkan dengan menghapus sumber web berita seperti Kompas dan Liputan6. Setelah sumber web berita telah dihapuskan maka dilanjutkan dengan menghapus simbol, angka dan enter pada setiap kalimat ataupun kata yang ada. Setelah proses menghapus kata yang tidak memiliki konteks pada berita maka proses dilanjutkan dengan membuat semua kata menjadi huruf kecil, agar program membaca “Maka” dan “maka” dengan kata yang sama. Penghapusan *stopwords* atau disebut juga sebagai kata-kata yang berulang dan tidak memiliki makna yang cukup berarti. Pada penelitian ini, proses penghapusan *stopwords* menggunakan *library* Sastrawi yang salah satu dari *stopwords* itu adalah kata

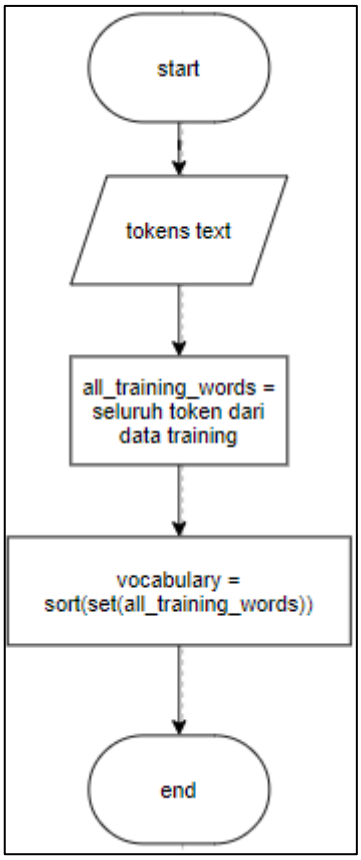
‘yang’. Setelah *stopwords* berhasil dihapus, maka dilakukan proses *stemming* yang menggunakan *library* Sastrawi. Proses *stemming* digambarkan pada gambar 3.4



Gambar 3.4 Proses *Stemming*

Proses *stemming* merupakan langkah untuk mentransformasi sebuah kata menjadi kata dasar atau *root.* Proses ini masuk dalam cabang ilmu *Natural Language Processing* (NLP). Jika sebuah kata masukan merupakan kata dasar, maka kata tersebut akan dikembalikan dan proses *stemming* pada kata tersebut selesai. Namun, jika kata tersebut belum kata dasar, maka proses dilanjutkan dengan menghapus *inflection suffixes* seperti -lah,-kah,-ku dan sebagainya. Jika kata memiliki partikel berupa -lah, -kah, -tah, dan -pun, maka proses penghapusan

inflection suffixes diulangi dan menghapus *possesive pronouns* seperti -ku, -mu, dan -nya. Selanjutnya dilakukan penghapusan imbuhan turunan seperti -i, -kan, -an dan penghapusan awalan turunan seperti be-, di-, ke-, me- dan sebagainya. Setelah proses penghapusan selesai, maka teks sudah bertransformasi menjadi kata dasar dan dikembalikan ke fungsi *cleaning text*. Pada fungsi *cleaning text*, teks yang sudah mencapai proses akhir akan disimpan. Proses selanjutnya akan menerapkan NLP pada setiap data teks dengan tujuan utama adalah membuat teks tersebut dapat dimengerti oleh *model machine learning*. Diawali dengan proses tokenisasi pada kumpulan teks yang sudah proses sebelumnya. Proses tokenisasi adalah proses memecah sebuah kalimat menjadi kata-kata. Pada saat tokenisasi dilakukan, terdapat proses POS(*Part-Of-Speech) tagging* pada setiap *token-token* yang dihasilkan. Proses POS *tagging* merupakan tahapan setiap *token* diberikan kelas kata seperti kata kerja, kata benda dan lainnya. Menggunakan *library* dari nlp-id yang dibuat oleh Kumparan. Cara kerja POS *tagging* pada *library* tersebut dengan menggunakan *model* pembelajaran mesin yaitu RandomForest yang membantu menentukan sebuah kata dengan kelas kata yang paling cocok. Diawali dengan proses tokenisasi yang selanjutnya dilakukan menentukan apakah sebuah *token* adalah simbol atau bukan, jika merupakan simbol maka *token* tersebut langsung diberi kelas kata 'SYM' yang berarti simbol. Selanjutnya, setiap *token* yang terbentuk akan diprediksi oleh *model* dari *library* dan hasilnya akan dipasang dengan *token* tersebut. Setelah proses tokenisasi berhasil, akan dilanjutkan dengan proses *create vocabulary*. Proses ini bertujuan untuk membuat kamus berdasarkan *token-token* yang ada. Proses *create vocabulary* dijelaskan pada gambar 3.5

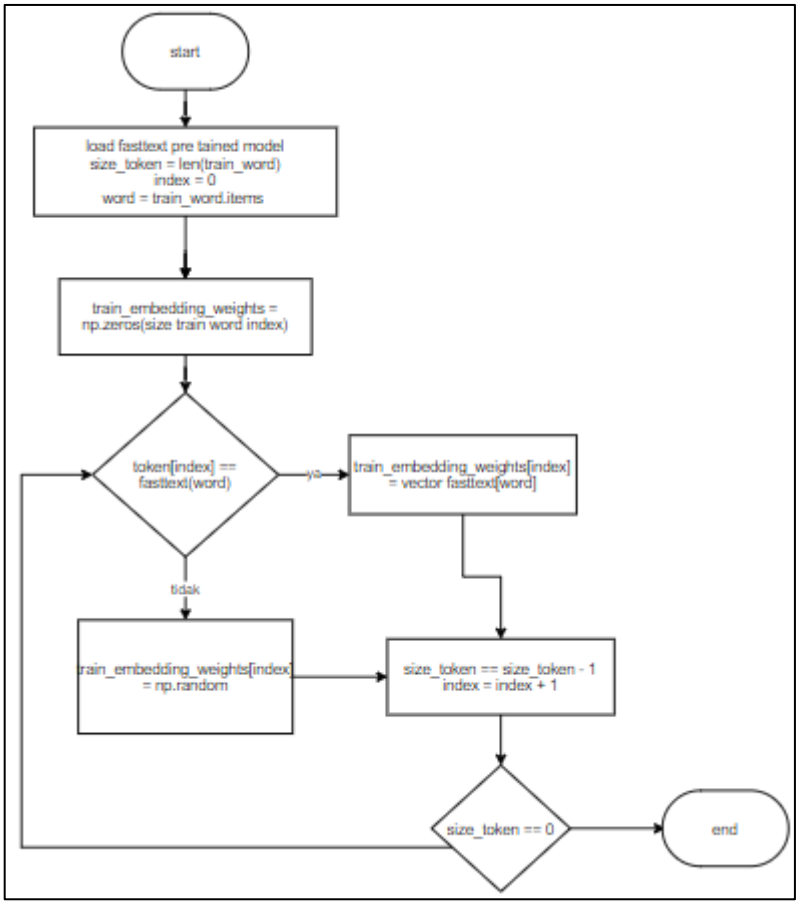


Gambar 3.5 Proses *Create Vocabulary*

Kumpulan *token* akan dijadikan menjadi kata-kata yang akan dilatih dengan mensortir *token* yang berbeda-beda. Jika sebuah *token* memiliki makna yang sama dengan lainnya, maka hanya diambil salah satu dari *token* tersebut dan akan disortir secara *ascending*. Selanjutnya, Teks yang sudah bersih diproses menjadi urutan angka-angka dengan menggunakan fungsi *pad sequences*. Fungsi ini bertujuan untuk membuat kumpulan teks pada sebuah berita menjadi nilai-nilai numerikal agar dapat dibaca oleh *model*. Dengan menerima masukan berupa teks dan menggunakan fungsi Tokenizer, maka setiap kata pada *vocabulary* memiliki nilai indeks yang akan menentukan nilai dari setiap kata pada sebuah berita. Hasil dari proses Tokenizer itu menjadi data kata latih. Panjang dari *pad sequences* ditentukan dari panjang rata-rata pada setiap teks berita yang ada.

3.3.3. Implementasi FastText

Setelah langkah *pre-processing* selesai, proses dilanjutkan dengan mengimplementasi FastText dengan menggunakan *library* Gensim sebagai wadah dari *model* Fasttext. *Model* Fasttext yang sudah diimplementasikan ke *model* Gensim akan dilakukan proses pemberian nilai *weight* yang langkah-langkahnya dapat dilihat pada gambar 3.6

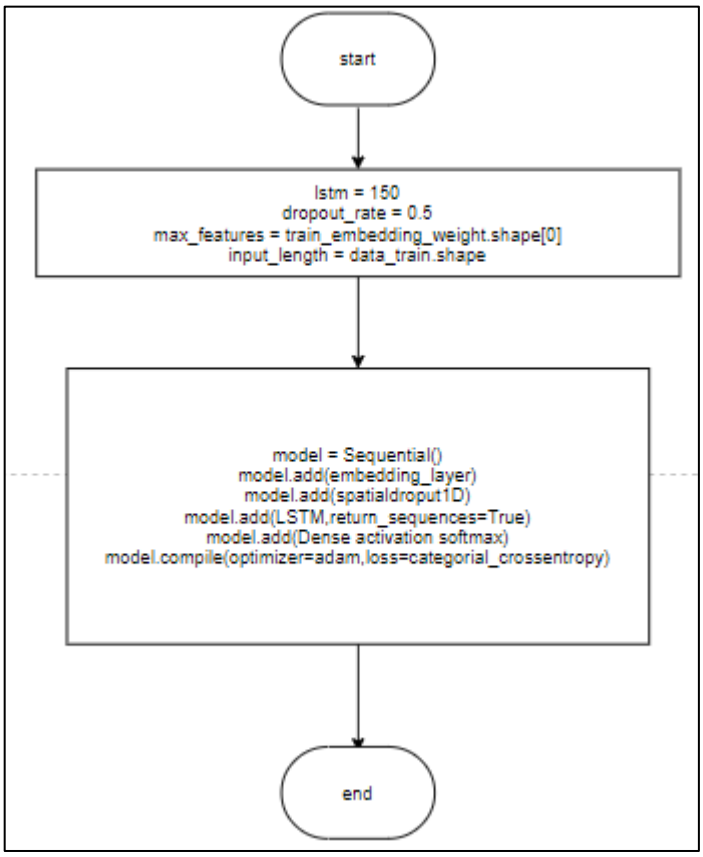


Gambar 3.6 Proses Implementasi Fasttext

Dengan menggunakan *model* yang sudah diimplementasi Fasttext, maka kata-kata di kata latih akan diberikan nilai *weight* dengan melakukan perulangan disetiap kata tersebut hingga setiap kata memiliki *weight* yang sesuai dengan *model.*

3.3.4. Pembuatan *Model* LSTM-RNN

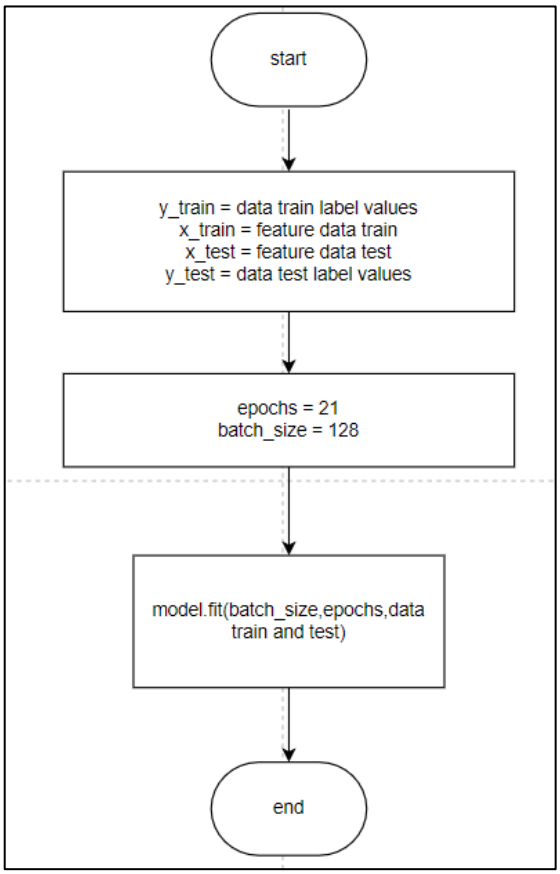
Pada tahap ini, proses yang akan dilakukan adalah pembuatan *model* LSTM-RNN menggunakan *library* Keras Tensorflow. Langkah-langkah pembuatan *model* dapat dilihat pada gambar 3.7



Gambar 3.7 Pembuatan *Model* LSTM-RNN

Pembuatan *model* diawali dengan menerapkan fungsi Sequential kepada *model* yang bertujuan untuk membuat *layer-layer* yang berurutan sesuai dengan urutan saat *model* ditambahkan dengan fungsi *add*. *Layer-layer* yang dibuat meliputi 1 *embedding layer,* 1 *spatialdropout*, 1 LSTM *layer* dan 1 dense *layer*.

Embedding layer merupakan *layer* awal untuk setiap masukan *pad sequences* yang akan dikonversi dengan nilai Fasttext. *Spatialdropout layer* merupakan layer yang berfungsi untuk mencegah terjadinya *overfitting* pada *model*. LSTM *layer* merupakan arsitektur utama pada *model* ini yang memiliki berbagai *gate* yaitu *forget, input, output* dan *cell state*. Selanjutnya dense *layer* yang mengkonversi hasil perhitungan menjadi probabilitas dan penentuan kategori dari berita yang menjadi masukan diawal pada *model.* Setelah proses pembuatan *model* berhasil, maka dilanjutkan dengan melatih *model* dengan data *training* dan data *testing*. Data *training* memiliki jumlah berita sebanyak 9.727 berita dan data *testing* sebanyak 3.327 berita. Langkah-langkah pelatihan *model* dapat dilihat pada gambar 3.8



Gambar 3.8 Pelatihan *Model*

Pelatihan *model* dilakukan sebanyak 21 *epochs*. *Epochs* adalah saat seluruh data sudah melalui proses pelatihan sampai kembali ke awal dan diulangin sebanyak jumlah *epochs*. Satu *epochs* terlalu besar jika dijalankan menggunakan data yang penuh, maka agar tidak terlalu besar dibutuhkan *batch* yang membagi-bagi setiap data dalam bentuk *batch*. Jika proses pelatihan *model* telah selesai, maka dilanjutkan dengan proses evaluasi performa *model* menggunakan *library* Sklearn yang menampilkan nilai *presisi, recall, accuracy* dan *f1-score*. Selain itu, pada tahap evaluasi performa model juga menggunakan *confusion matrix* yang bekerja dengan cara menampilkan hasil prediksi dari kategori sesungguhnya pada berita dalam bentuk matriks dengan ukuran sebesar jumlah kategori berita yang ada. Nilai dari isi matriks dihitung dari hasil prediksi sebuah kategori berita dengan kategori sesungguhnya. Misalnya, pada sebuah kategori *news* terdapat 880 berita, *model* memprediksi bahwa terdapat 815 *news,* 16 berita bola, 45 berita bisnis, 3 berita otomotif dan 1 berita tekno. Dengan hasil tersebut, maka dapat disimpulkan bahwa *model* memprediksi dengan benar sebanyak 815 berita dengan tingkat akurasi 92% yang dihitung dari jumlah hasil prediksi *model* / jumlah total berita *news*. Hasil prediksi *model* Proses pembuatan *model* dilakukan berulang kali hingga mendapatkan hasil evaluasi terbaik. Setelah mendapatkan *model* yang terbaik, maka *model* tersebut disimpan dan akan digunakan oleh sistem dalam melakukan prediksi secara *real-time*.

3.3.5. Rancangan dan Alur Kerja Sistem

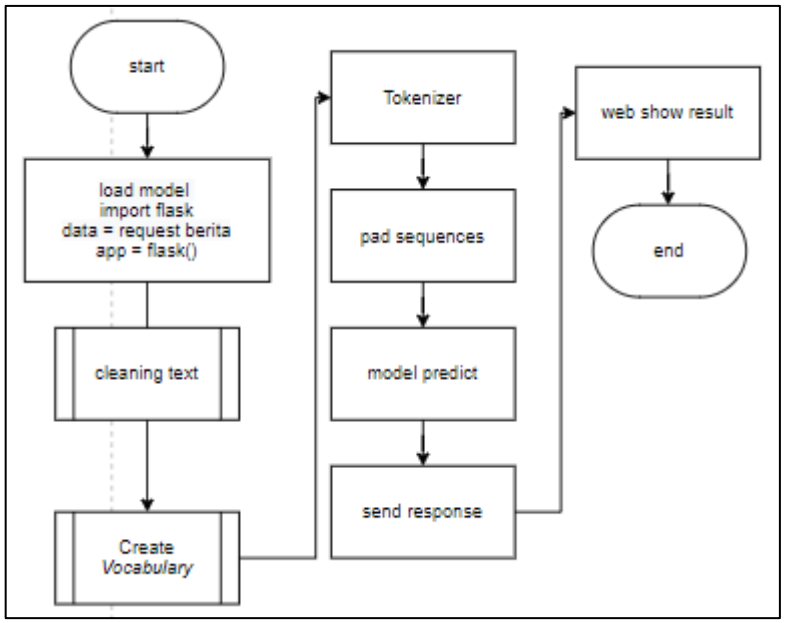
Sistem yang akan menjadi tempat pengguna dapat melakukan prediksi dari kategori berita sesuai masukan dari berbagai berita yang ada dibuat

menggunakan *framework* berbasis bahasa PHP yaitu Laravel versi 7 dan pada gambar 3.9 merupakan rancangan antarmuka dari *web*

Gambar 3.9 Rancangan Antarmuka Web

Pada rancangan antarmuka tersebut terdapat judul dari *web*, dibawah *label* isi berita merupakan *text area* agar dapat menampung jumlah kata yang banyak dan tombol *submit* untuk memulai proses prediksi yang hasilnya akan ditampilkan pada bagian hasil prediksi tepat dibawah tombol tersebut.

Alur kerja dari *web* adalah dengan menerima *request* dari *web* yang dikirimkan menggunakan AJAX ke *model* yang menggunakan *framework* Flask. *Flask* akan menerima data berupa teks berita dan dilakukan proses *pre-processing* hingga menghasilkan prediksi. Setelah prediksi selesai dilakukan, maka akan dikirimkan *response* berupa JSON yang akan diterima oleh ajax pada *web* tersebut. Langkah-langkah implementasi *model* dengan *framework* Flask terdapat pada gambar 3.10



Gambar 3.10 Implementasi *model* dengan *Web Service*

Proses diawali dengan mengaktifkan Flask agar dapat menerima *request* dari *web*. Setelah itu proses dilakukan sama persis seperti diawal pembuatan model. Perbedaan terletak pada *model* tidak melakukan pelatihan namun langsung melakukan prediksi dari *model* yang sudah dimuat dan mengirimkan hasil prediksi sebagai akhir dari proses prediksi.