BAB III

METODE PENELITIAN

3.1. Objek dan Variabel Penelitian

Dalam penelitian kali ini, objek penelitian yang dimaksud yakni *tweets* yang mengandung kata kunci ‘indomie’. Adapun variabel penelitian ini diantaranya:

1. Variabel Independen

   Variabel independen yang dimaksud yakni :

   X_1 = Kata di dalam *tweets* yang mengandung kata kunci ‘indomie’

2. Variabel Dependen

   Dalam penelitian ini, variabel dependen yang dimaksud yakni :

   Y_1 = Sentimen positif

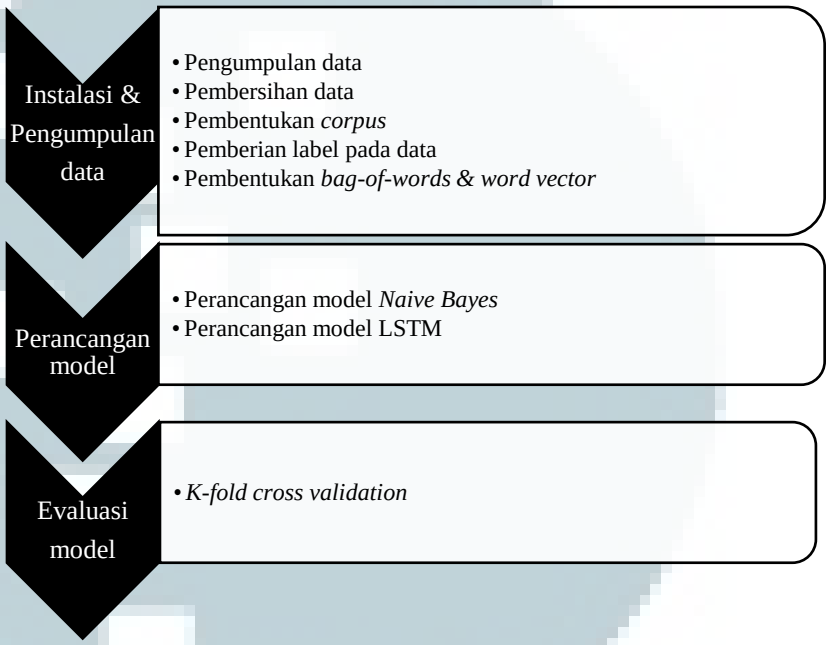
   Y_2 = Sentimen negatif

3.2. Metode Penelitian

Dalam penelitian ini terdapat 3 langkah, diantaranya: (1) instalasi dan pengumpulan data, (2) perancangan model, dan (3) evaluasi model. Seluruh langkah diselesaikan di dalam mata kuliah: SKRIPSI I, kecuali: pembentukan *word2vec: skip-gram model*, dan perancangan model *Long Short Term Memory* (LSTM). Untuk dapat merancang model LSTM dengan tepat, *datasets* harus

diamati terlebih dahulu. Pengamatan *datasets* dan perancangan model *Naïve Bayes* diselesaikan dalam mata kuliah: SKIRPSI I.

Ketiga langkah penelitian dapat dilihat dalam diagram berikut:



Gambar 3.1. Deskripsi langkah penelitian

3.2.1. Pengumpulan Data

A. Pengumpulan Data

Data yang dikumpulkan yakni *tweets* yang mengandung kata kunci: "indomie". *Tweets* dengan kata kunci tersebut dikumpulkan minimal sebanyak 10.000 *tweets*. Secara garis besar, terdapat 6 tahapan dalam pengumpulan data:

1. Pembuatan akun *Twitter* dan akun *developer Twitter* pada situs: *developer.twitter.com*. Pembuatan akun dimaksudkan untuk

memperoleh kunci dan akses token yang diperlukan dalam memperoleh *tweets*.

2. Pengambilan data dilakukan dengan menggunakan *tools Python*, dan *library Tweepy*. Kunci dan token yang diperoleh kemudian dimasukkan untuk keperluan autentikasi akses token *API Twitter.*
3. Penetapan kata kunci *tweets* yang hendak dikumpulkan, yakni "indomie".
4. Pengumpulan *tweets* yang mengandung kata "indomie" dan menampung *tweets* tersebut ke dalam *array.*
5. *Tweets* yang mengandung *emoticon* ataupun karakter khusus diterjemahkan ke dalam karakter *Unicode.*
6. *Tweets* disimpan ke dalam *file* ekstensi *.txt.*

Tweets sendiri dikumpulkan dari Mei 2017 – Juni 2017. *Tweets* diperoleh secara bertahap setiap harinya dengan *tools Python*. Adapun jumlah *tweets* yang diperoleh dalam setiap harinya minimal 400 *tweets*. Total *tweets* yang diperoleh yakni sebanyak: 50.000 *tweets*.

B. Pembersihan Data

Berikut tahapan *preprocessing data* yang dilakukan yakni :

1. Penghapusan URL dan link gambar yang tertera pada setiap *tweets*

2. Pembersihan data dengan menghilangkan tanda kutip pada awal dan akhir *tweets*

Sementara itu, *hashtags*, *mentions*, dan *retweets* tetap dipertahankan demi menjaga orisinalitas makna *tweets*. Pembersihan data juga dilakukan dengan *tools Python*.

C. Pembentukan *Corpus*

Corpus merupakan bank kata atau kumpulan kata secara keseluruhan yang terdiri atas *tweets* yang sudah mengalami tahap *preprocessing* dan sudah siap digunakan sebagai data *training* dan *testing* model. *Corpus* sendiri disimpan dalam *file* berekstensi *.txt*.

D. Pemberian Label pada Data

Untuk mempermudah proses pemberian label *tweets* dan pengolahan *tweets*, maka *tweets* tersebut dipindahkan ke dalam *file* dengan ekstensi *.csv*. Adapun format kolom dalam *file* yakni: (1) *tweet*, dan (2) nilai sentimen *tweet*.

Usai mengumpulkan seluruh data yang dimaksud dalam *corpus*, tahap selanjutnya yakni pemberian label setiap *tweets* pada *corpus*; label untuk sentimen positif bernilai: 1, dan label sentimen negatif bernilai: 0. *Tweets* yang terdapat pada file *.txt* dipindahkan ke dalam file ekstensi *.csv* untuk mempermudah proses pemberian label *tweets* dan pengolahan data lebih lanjut. Pemberian label pada

seluruh *tweets* dilakukan secara manual. Adapun jumlah *tweets* yang diberi label yakni: 13,715 *tweets*.

E. Pembentukan *Bag-of-words* dan *Word Vector*

Setelah memberi label pada data yang dimaksud, terdapat proses pembentukan *bag-of-words*, yakni dengan membuat indeks yang terdiri atas seluruh kata yang terdapat di dalam *tweets*, dan menghitung frekuensi kata tersebut. Indeks *bag-of-words* sendiri digunakan untuk menghitung probabilitas suatu kata pada model *Naïve Bayes*. Adapun pembentukan *word vector* yakni dengan membuat *vector* yang terdiri atas semua kata pada *tweets*. *Word vector* digunakan sebagai input data bagi model *Long Short Term Memory*.

3.2.2. Perancangan Model

A. Perancangan Model *Naïve Bayes*

Naïve Bayes merupakan salah satu model yang dapat digunakan dalam mengklasifikasi sentimen kelas terhadap kata dengan menghitung probabilitas kata terhadap kelas, probabilitas kelas dan kata itu sendiri. Adapun tipe *Naïve Bayes* yang dirancang yakni *multinomial Naïve Bayes*. Tujuan utama dari perancangan model *Naïve Bayes* ialah untuk mengetahui tingkat akurasi klasifikasi *tweets* dengan asumsi data secara independen. Secara garis besar,

berikut tahapan yang dilakukan dalam merancang model *Naïve Bayes* dengan menggunakan *tools SPYDER*:

1. Impor beberapa *libraries* yang diperlukan. Adapun *libraries* yang digunakan diantaranya:
   - *Panda, library* yang dapat mengolah data yang terdapat pada file ekstensi "*.csv*"
   - *NumPy, library* yang digunakan untuk mengolah perhitungan bilangan yang kompleks
   - *Matplotlib, library* yang diperuntukkan untuk memvisualisasikan data yang telah diolah
   - *Sklearn, library* dengan modul : (1) *K-fold* untuk *cross validation*, (2) *metrics* dan *AUC* untuk perhitungan ROC (*Receiver Operating Characteristic*)
2. Setiap *tweets* beserta skornya yang bersumber dari file beresktensi *.csv* disimpan ke dalam *array list.*
3. Konfigurasi *K-fold cross validation* yang hendak digunakan, yakni *n_folds*: 10.
4. Pisahkan seluruh data sesuai dengan *n_folds* yang ditetapkan. Seluruh data dipisahkan untuk keperluan *training* dan *testing* dan ditampung ke dalam *array list*.
5. Pembuatan fungsi yang menghitung probabilitas kata.

Berikut tahapan perhitungan dalam model *Naïve Bayes*, yakni:

1. Pencatatan setiap kata yang terdapat pada *training tweets*

   Kata-kata yang bermakna sama dan memiliki bentuk berbeda tetap disimpan sebagai kata baru. Bentuk berbeda yang dimaksud yakni kata yang mengalami kapitalisasi, penggunaan kombinasi huruf dan angka yang membentuk kata tertentu. Sebagai contoh, kata: "makan" dan "Makan" disimpan sebagai 2 kata baru. Kata-kata tersebut disimpan ke dalam *dictionary*.
2. Perhitungan frekuensi kata yang terdapat pada *training tweets.*
3. Perhitungan *prior probability* untuk kedua kelas; positif dan negatif. Hasil perhitungan kemudian dibagi dengan jumlah kelas yang ada, yakni: 2, untuk menghindari nilai probabilitas mencapai lebih dari 100%
4. Perhitungan jumlah kata yang terdapat pada *tweets* yang memiliki makna positif ataupun negatif. Jumlah kata tersebut disimpan ke dalam *dictionary*.
5. Perhitungan *likelihood* atau probabilitas untuk setiap kata terhadap kelas yang muncul pada *training tweets*.
6. Perhitungan probabilitas kata yang terdapat pada *training tweets*.
7. Perhitungan probabilitas kelas terhadap setiap kata pada *training tweets* terhadap kelas positif dan negatif.
8. Tingkat keyakinan atau *certainty* suatu *tweet* juga diperhitungkan, yakni: 1 - (kata *testing* data yang tidak terdapat pada *training* data / total kata dalam suatu *tweet*)

Dalam merancang model *Naïve Bayes*, terdapat kustomisasi fitur diantaranya:

1. *Add-One Laplace Smoothing*

   Add-One Laplace Smoothing merupakan salah satu teknik untuk menghitung probabilitas kata terhadap kelas sekalipun kata tersebut tidak pernah hadir di dalam *training data*. Dalam hal ini, *Add-One Laplace Smoothing* menambah setiap probabilitas kata terhadap kelas sejumlah 1, terlepas dari ada atau ketiadaan suatu kata di kedua kelas; positif dan negatif. Meski dapat menghitung seluruh probabilitas kata, namun pada kasus tertentu *Add-One Laplace Smoothing* dapat mengurangi sensitivitas kata unik yang hanya terdapat dalam *tweets* bermakna positif atau negatif. Untuk itu, dilakukan tes model *Naïve Bayes* dengan fitur *Add-One Laplace Smoothing* dan tanpa fitur *Add-One Laplace Smoothing*.

2. Kapitalisasi kata pada *tweets*

   Kapitalisasi kata yang dimaksud yakni tetap mempertahankan kapitalisasi yang ada pada *tweets* tanpa adanya pengubahan. Kata yang sama namun ditulis dengan atau tanpa kapitalisasi dimasukkan sebagai kata yang berbeda dalam proses pembuatan indeks kata. Sementara itu, penelitian mengenai klasifikasi *tweets* sebelumnya (Jiang, 2011) menghilangkan seluruh kapitalisasi pada *tweets* agar dapat menghitung probabilitas kata

bermakna sama. Diperlukan eksplorasi lebih lanjut terkait penggunaan kapitalisasi kata terhadap sentimen *tweets*. Untuk itu, dilakukan tes model *Naïve Bayes* dengan fitur kapitalisasi kata dan tanpa fitur kapitalisasi kata.

Berdasarkan kustomisasi fitur tersebut, berikut kombinasi pengembangan model *Naïve Bayes*:

- *Naïve Bayes* tanpa fitur *Add-One Laplace Smoothing* dan dengan kapitalisasi kata.
- *Naïve Bayes* tanpa fitur *Add-One Laplace Smoothing* dan tanpa kapitalisasi kata.
- *Naïve Bayes* dengan fitur *Add-One Laplace Smoothing* dan dengan kapitalisasi kata.
- *Naïve Bayes* dengan fitur *Add-One Laplace Smoothing* dan tanpa kapitalisasi kata.

Selain mengembangkan model, terdapat pula eksplorasi model *Naïve Bayes* dalam menganalisa sentimen *tweets* dengan jumlah *datasets* yang seimbang, antara *tweets* bermakna positif dan *tweets* bermakna negatif.

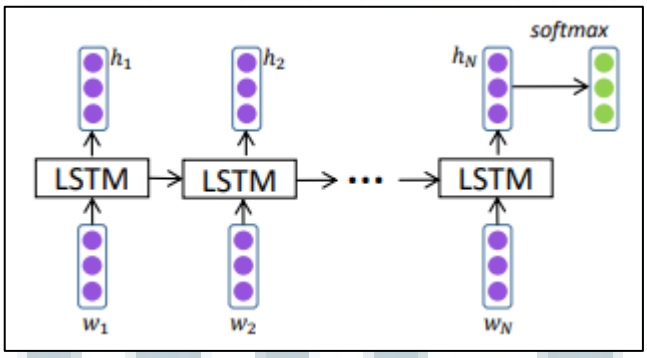
B. Perancangan Model *Long Short Term Memory* (LSTM)

Long Short Term Memory (LSTM) dirancang dengan bahasa pemrograman *Python* dan menggunakan dependensi *library Tensorflow*. Inputan model LSTM yakni berupa *word vector* setiap

kata yang terdapat di dalam *tweets*. Dalam hal ini, kata direpresentasikan dalam bentuk *one-hot vector*, kemudian dimasukkan ke dalam model *word2vec skip-gram model* untuk menghasilkan *word vector*. Berikut konfigurasi untuk *word2vec skip-gram model*:

- Dimensi (*dimension*): 32
- Ukuran skip (*skip size*): 8
- Ukuran window (*skip window*): 1
- Jumlah skips (*number skips*): 2
- Jumlah iterasi training model (*number iterations*): 10,000

Model LSTM mengolah inputan berupa *word vector* menjadi *hidden vector* yang kemudian dijadikan nilai probabilitas untuk masing-masing kelas; positif dan negatif. Berikut arsitektur model LSTM yang diimplementasikan (Wang Y. a., 2016):



Gambar 3.2. Arsitektur model LSTM

Dimana w_1, w_2, ..., w_N merepresentasikan *word vector* dalam sebuah kalimat dengan jumlah kata sebanyak *N*. h_1, h_2, ..., h_N ialah *hidden vector*. Hasil terakhir dari *hidden vector* kemudian diteruskan menuju *layer softmax* sehingga diperolehlah probabilitas untuk kedua kelas; positif dan negatif.

Model lstm yang dirancang menggunakan konfigurasi berikut:

- Jumlah layer LSTM (*size lstm layer*): 1
- Jumlah sel LSTM (*size lstm cell*): 128
- Tingkat pembelajaran (*learning rate*): 0.001
- *Batch data*: 1
- Ukuran *word embedding* (*size embedding*): 140
- Nilai *dropout* (*dropout value*): 0.5

Berikut tahapan yang dilakukan dalam mengembangkan model LSTM:

1. Penggunaan beberapa library yang mendukung perancangan model, diantaranya:
    - *Tensorflow*, library utama yang digunakan dalam perancangan model LSTM dan *word2vec*
    - Pandas, *library* yang dapat mengolah data yang terdapat pada *file* ekstensi "*.csv*"

- *NumPy*, *library* yang digunakan untuk mengolah perhitungan bilangan yang kompleks, kerap kali digunakan untuk operasi perhitungan pada matriks
- *Collections*, *library* yang digunakan untuk mengelola kata
- *Random*, *library* untuk menghasilkan nilai angka secara acak
- *Time*, *library* untuk menampilkan waktu saat fungsi pada model dijalankan dan digunakan untuk menghitung kecepatan model dalam mengolah data
- *Sklearn.Model_Selection Kfold*, *library* yang digunakan untuk membagi *dataset* ke dalam *training* dan *testing* sejumlah nilai *k*

2. Penyimpanan *tweets* yang terdapat pada *file* dengan ekstensi "*.csv*" ke dalam *list*. *List* tersebut kemudian dikonversi menjadi *array*.
3. Penyimpanan kata-kata yang terdapat pada *tweets* ke dalam bentuk *array*, dan pengindeksan kata.
4. Pembuatan *batch dataset* untuk keperluan *training model skip-gram*. *Batch* sendiri merupakan jumlah kumpulan *training tweets* yang diolah dalam satu kali *fold* atau sesi *training*.
5. Pengaturan konfigurasi untuk sesi *training word2vec* dan model *skip-gram*.

6. Penginputan *one-hot vector* ke dalam model *skip-gram* berdasarkan *batch dataset* yang telah dibuat, kemudian mulai melatih model hingga menghasilkan *output* berupa *word vector*. Terdapat pencatatan *loss* model dan durasi *training* untuk setiap *epoch*.
7. Pengaturan konfigurasi untuk model LSTM dan penetapan nilai *k-fold*, yakni: 10.
8. Pemisahan *dataset* sesuai dengan nilai *k-fold* yang ditetapkan, dan pengubahan bentuk *dataset* dari *list* menjadi *array*.
9. Perancangan model LSTM:
    - Pembuatan 2 *placeholder* untuk: (1) *tweets* dalam bentuk *vector*, (2) skor *tweets* dalam bentuk *array*
    - Penetapan nilai *dropout*
    - Pembuatan *layer* LSTM sesuai dengan konfigurasi
    - Pembuatan sel LSTM sesuai dengan konfigurasi
    - Inisialisasi untuk memori LSTM
    - Penetapan nilai *weight* dan *bias* secara *random*
    - Perhitungan *logits* dengan mengalikan *hidden vector* terakhir dalam *tweet* dengan *weight* yang kemudian ditambah dengan *bias*
    - Perhitungan nilai *loss* untuk *training model* LSTM
    - Penggunaan algoritma *Adam* untuk *training model*

- Perhitungan akurasi prediksi model terhadap *dataset* yang diberikan

10. Pengaturan konfigurasi perihal penggunaan *Graphics Processing Unit* (GPU) dalam menjalankan sesi *training model* LSTM. Dalam penelitian ini, *gpu usage* diatur menjadi 0.6.
11. Pembuatan *batch dataset* untuk keperluan *training* dan *testing dataset*.
12. Pengaturan sesi *training model* LSTM, serta pencatatan *loss*, *weight*, dan nilai akurasi ke dalam bentuk teks.

3.2.3. Evaluasi Model

Model *Naïve Bayes* dan LSTM akan dievaluasi dengan menggunakan metode *K-fold cross validation*, dimana evaluasi dilakukan pada *datasets* yang sama, baik untuk keperluan *training* maupun *testing*. Terdapat pembagian proporsi *training* dan *testing* yang berbeda-beda sesuai dengan nilai k yang ditetapkan. Dalam penelitian kali ini, kedua model dievaluasi dengan *K-fold cross validation* dengan $k = 10$.

3.3. Teknik Olah Data

Setelah data diberi label, selanjutnya *datasets* akan dibagi ke dalam 2 bagian; *training* dan *testing* data. *Training* data dijadikan sebagai bahan latih bagi model *Naïve Bayes* dan *Long Short Term Memory*, kemudian dijadikan dasar bagi kedua model dalam mengklasifikasi *testing* data. Dalam hal ini, kedua model

mengklasifikasi *testing* data ke dalam sentimen kelas positif, yang bernilai 1 atau negatif yang bernilai 0. Kedua model tersebut dirancang dengan menggunakan *tools SPYDER* berbasis *Python*, dan *software library TensorFlow*.

Dalam mengolah *datasets*, terdapat pertimbangan untuk memisahkan karakter dan *emoticons* khusus yang tergabung ke dalam kata lain tanpa adanya spasi ataupun *whitespace* dan menghitungnya sebagai satu karakter atau *emoticons*. Sebagai contoh *tweet* yang mengandung kata: "Indomie seleramu\ufffd\ufffd" mengandung karakter khusus \ufffd yang jika di*encode* akan menjadi karakter **.** Jika terdapat pemisahan pada karakter khusus, maka indeks kata pada *tweet* tersebut menjadi: "Indomie", "seleramu", "/ufffd". Namun, jika tidak terdapat pemisahan pada karakter khusus, maka indeks kata menjadi: "Indomie", "seleramu\ufffd\ufffd".

Penelitian saat ini tidak memisahkan karakter ataupun *emoticons* khusus yang tergabung ke dalam kata lainnya. Dalam hal ini, kata tetap dimasukkan ke dalam indeks seperti apa adanya. Beberapa tujuannya yakni: (1) menghindari terjadinya perubahan makna suatu *tweet* dan (2) mempertahankan orisinalitas data.