BAB II

LANDASAN TEORI

2.1. *Naïve Bayes*

Naïve Bayes merupakan model yang sederhana, dimana model ini mengasumsikan semua artibut dari data secara independen. Asumsi independen yang dimiliki oleh *Naïve Bayes* sendiri menyebabkan parameter dari setiap artibut dapat dipelajari terpisah. Hal tersebut membuat proses pembelajaran model *Naïve Bayes* menjadi lebih singkat, terlebih lagi jika data *training* berukuran besar.

Dalam penelitian ini, *Naïve Bayes* yang digunakan yakni *Naïve Bayes multinomial event model*, dimana model *Naïve Bayes* memperhitungkan frekuensi kata yang muncul dalam dokumen. Berikut perhitungan yang diterapkan pada *multinomial*:

$$\mathrm{P}(c_j|d_i) = \frac{\mathrm{P}(d_i|c_j)\,\mathrm{P}(c_j)}{\mathrm{P}(d_i)}$$

Rumus 2.1. Perhitungan model *Multinomial*

Sumber: (Xuan, 2017)

Dalam model *multinomial*, $\mathrm{P}(c_j|d_i)$ ialah probabilitas kelas c terhadap suatu kata d indeks ke-i , $\mathrm{P}(d_i|c_j)$ menunukkan probabilitas kata terhadap suatu kelas, $\mathrm{P}(c_j)$ menunjukkan probabilitas suatu kelas, dan $\mathrm{P}(d_i)$ menunjukkan probabilitas kata.

Implementasi tipe *Naïve Bayes* ditentukan dari data yang akan diolah lebih lanjut. Jika data memiliki varian kata yang beragam, dan jumlah yang banyak maka perhitungan frekuensi setiap kata yang muncul di dalam data sangatlah diperlukan. Untuk itu, dalam penelitian ini menerapkan *Naïve Bayes multinomial event model*.

Dalam penelitian ini, terdapat pengukuran tingkat akurasi prediksi model terhadap *tweets* bermakna positif dan negatif (*Total Accuracy*), tingkat akurasi prediksi model terhadap *tweets* bermakna positif (*Positive Prediction Accuracy*) dan tingkat akurasi prediksi model terhadap *tweets* bermakna negatif (*Negative Prediction Accuracy*) untuk setiap model *Naïve Bayes* yang dirancang.

Berikut perhitungan tingkat akurasi model terhadap *tweets* bermakna positif (*Positive Prediction Accuracy*):

$$PPA = \frac{True\ Positive}{Total\ Positive\ Tweets\ in\ Testing\ Set}$$

Rumus 2.2. Perhitungan *Positive Prediction Accuracy*

Dimana *PPA* merupakan tingkat akurasi model terhadap *tweets* bermakna positif (*Positive Prediction Accuracy*), yang dihitung dengan membagi *True Positive*, yakni jumlah prediksi *tweets* bermakna positif yang tepat, dengan *Total Positive Tweets in Testing Set*, yakni jumlah *tweets* bermakna positif yang berada di dalam *testing set*.

Berikut perhitungan tingkat akurasi model terhadap *tweets* bermakna negatif (*Negative Prediction Accuracy*):

$$NPA = \frac{True\ Negative}{Total\ Negative\ Tweets\ in\ Testing\ Set}$$

Rumus 2.3. Perhitungan *Negative Prediction Accuracy*

Dimana *NPA* merupakan tingkat akurasi model terhadap *tweets* bermakna negatif (*Negative Prediction Accuracy*), yang dihitung dengan membagi *True Negative*, yakni jumlah prediksi *tweets* bermakna negatif yang tepat, dengan *Total Negative Tweets in Testing Set*, yakni jumlah *tweets* bermakna negatif yang berada di dalam *testing set*.

Adapun tingkat akurasi model terhadap *tweets* bermakna positif dan negatif (*Total Accuracy*) dihitung sebagai berikut:

$$Total\ Accuracy = \frac{PPA + NPA}{2}$$

Rumus 2.4. Perhitungan *Total Accuracy*

Dimana *Total Accuracy* merupakan tingkat akurasi model terhadap *tweets* bermakna positif dan negatif, yang dihitung dengan menjumlahkan hasil *PPA (Positive Prediction Accuracy)* dengan *NPA (Negative Prediction Accuracy)*, kemudian membagi hasilnya dengan 2. Selain menghitung tingkat akurasi, terdapat pula perhitungan tingkat keyakinan model *Naïve Bayes* (*certainty*) dalam mengklasifikasi satu *tweets* dengan perhitungan sebagai berikut:

$$\text{Certainty} = \left(1 - \frac{Unknown\ words\ in\ a\ tweet}{Total\ word\ in\ a\ tweet}\right) X\ 100$$

Rumus 2.5. Perhitungan *Certainty*

Dimana *Certainty* ialah tingkat keyakinan hasil prediksi model terhadap *datasets*, yang diperoleh dari selisih 1 dengan hasil pembagian antara jumlah kata di dalam *tweets* yang tidak muncul di dalam *training set* namun hadir di dalam *testing set* (*Unknown words in a tweet*) dibagi dengan jumlah seluruh kata dalam satu *tweet* (*Total word in a tweet*), dikali dengan 100. *Certainty* pada satu *tweet* kemudian ditambahkan dengan *Certainty tweets* lainnya sejumlah *tweets* yang terdapat pada testing set, dan hasil akhirnya merupakan rata-rata *Certainty* dari seluruh *testing set*.

Tingkat keyakinan dimaksudkan untuk mengetahui seberapa kredibel hasil klasifikasi model *Naïve Bayes* terhadap *datasets*. Semakin tinggi tingkat keyakinan, maka semakin tinggi pula tingkat kredibilitas hasil klasifikasi model *Naïve Bayes*.

2.2. *Recurrent Neural Network*

Recurrent Neural Network (RNN) merupakan salah satu varian *Neural Network* yang memiliki *recurrent connections*, yang memungkinkan bagi model RNN untuk menyimpan data ke dalam *memory*. Adanya *recurrent connections* tersebut juga memungkinkan bagi RNN untuk menjalankan serangkaian tugas yang bersifat sekuensial. Dalam hal ini, perhitungan *hidden layer* h_t pada masa waktu tertentu *t* melibatkan input *layer* saat ini x_t dan *hidden layer* sebelumnya h_{t-1}. Kemudian, hasil akhirnya dihitung dengan menggunakan *hidden layer* saat ini h_t.

Berikut persamaan yang digunakan dalam menghitung *hidden layer* dan *output* :

$$h_t = f(Wx_t + Vh_{t-1} + b)$$

$$y_t = g(Uh_t + c)$$

Rumus 2.6. Perhitungan *hidden layer* & *output* LSTM

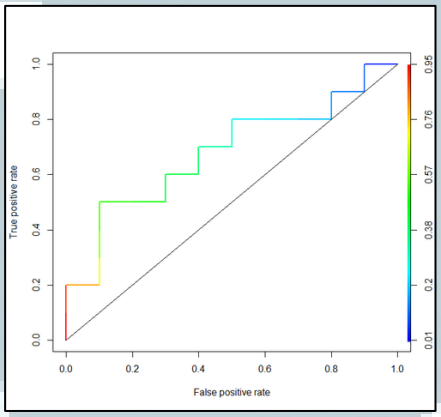
Dalam persamaan tersebut, f merupakan *non-linear functions*, seperti $sigmoid$ dan g merupakan *output nonlinearity*, seperti $softmax$. W dan V merupakan *weight matrices* antara *input* dengan *hidden layer*, dan antara *hidden layers* itu sendiri. U merupakan *output weight matrix*, sedangkan b dan c ialah *bias vector*. y_t merupakan *final output* yang dihasilkan.

2.3. *K-fold Cross Validation*

Agar dapat menciptakan model dengan tingkat klasifikasi yang optimal, maka diperlukan suatu metode evaluasi yang tepat untuk memprediksi *error rate* dalam model yang dibangun. Salah satu metode evaluasi yang dapat digunakan dengan jumlah *datasets* yang minim yakni *K-fold cross validation* (Bengio, 2004). *K-fold cross validation* merupakan metode yang membagi *datasets* untuk keperluan *training* dan *testing* secara proposional, dimana proses *training* sendiri dilakukan sebanyak *K* kali dengan proporsi: 1- $\frac{1}{K}$ dari total *datasets* yang diikutsertakan dalam *training*.

2.4. *Receiver Operating Characteristic*

Receiver Operating Characteristic (ROC) merupakan grafik yang digunakan untuk memvisualisasikan performa model klasifikasi biner yang telah dirancang (Fawcett, 2006). Salah satu tujuannya yakni untuk mengetahui prevalensi prediksi target populasi dari hasil *false positive* dan *false negative* (Greiner, 2000). Berikut contoh grafik ROC:



Gambar 2.2. Ilustrasi grafik ROC

Sumber: (Grigorev, 2015)

Sumbu y pada grafik ROC menunjukkan proporsi dari *true positive*, dan sumbu x menunjukkan proporsi dari *false positive*. Area dibawah kurva ROC atau AUC (*Area Under the Curve*) merefleksikan seberapa akurat model klasifikasi biner dalam memprediksi data yang diberikan. Semakin tinggi AUC, maka semakin akurat pula prediksi model. Jika nilai AUC berada dibawah 0.5, maka hasil performa klasifikasi dapat dikatakan sangat kurang (Akobeng, 2007). Secara visual, garis kurva akan menuju ke angka 1, jika hasil prediksi model akurat dan sebaliknya.

2.5. *Bag-of-words*

Bag-of-words kerap kali digunakan dalam klasifikasi teks. Klasifikasi teks dilakukan dengan memperhitungkan frekuensi kehadiran kata dalam suatu dokumen teks. Kata dengan frekuensi tertinggi namun tidak berarti apapun dalam suatu dokumen atau yang disebut dengan *stop words*, sebagai contoh: "*the*", "*a*", dapat dihilangkan dalam perhitungan kata. Tujuan utama dari implementasi *bag-of-words* ialah untuk mengetahui korelasi antar kata dan informasi yang terdapat di dalam topik suatu dokumen teks (Sethy, 2008).

2.6. *Add-One Laplace Smoothing*

Add-One Laplace Smoothing ialah salah satu teknik yang digunakan menghindari terjadinya nilai nol dalam perhitungan probabilitas kata (Juan, 2002). Dalam hal ini, terdapat penambahan nilai probabilitas kata sejumlah 1 dan penambahan nilai tersebut diterapkan untuk semua probabilitas kata, baik probabilitas kata yang bernilai nol dan lebih besar dari nol. *Add-One Laplace Smoothing* memungkinkan model *Naïve Bayes* untuk tetap memperhitungkan probabilitas kata terhadap kelas, sekalipun probabilitas kata tersebut bernilai nol.

2.7. *Word Vector*

Word vector merupakan salah satu teknik yang digunakan untuk merepresentasikan kata yang ada di dalam *corpus* ke dalam bentuk *vector* (Maas, 2011). *Word vector* dapat digunakan untuk mengetahui kedekatan makna antar kata, analisa semantik antar kata, dan dijadikan input data bagi model *Neural*

Network, salah satunya *Long Short Term Memory*. Kata dengan makna yang serupa memiliki representasi *vector* yang cenderung mirip.

2.8. *Long Short Term Memory* (LSTM)

Long Short Term Memory (LSTM) merupakan model yang mengelola data bersifat sekuensial, dan mampu menyempurnakan kekurangan pada model *Long Short Term Memory* (LSTM), yakni keterbatasan memori dalam mengingat konteks data, dan *vanishing exploding gradient problem* (Wang X. a., 2015). Sama halnya dengan LSTM, LSTM juga memiliki *recurrent layer* yang terdiri atas memori blok. Di setiap memori blok tersebut terdapat 3 jenis sel memori, diantaranya: (1) *forget gate*, gerbang memori yang mengelola penghapusan memori sebelumnya yang sudah disimpan, (2) *input gate*, gerbang memori yang mengelola penyimpanan informasi baru, (3) *output gate*, gerbang memori yang mengelola informasi yang dihasilkan (Huang, 2016). Berikut perhitungan yang digunakan pada masing-masing sel memori:

$$
\begin{aligned}
X &= \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \\
f_t &= \sigma(W_f \cdot X + b_f) \\
i_t &= \sigma(W_i \cdot X + b_i) \\
o_t &= \sigma(W_o \cdot X + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot X + b_c) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Rumus 2.7 Perhitungan pada sel memori LSTM

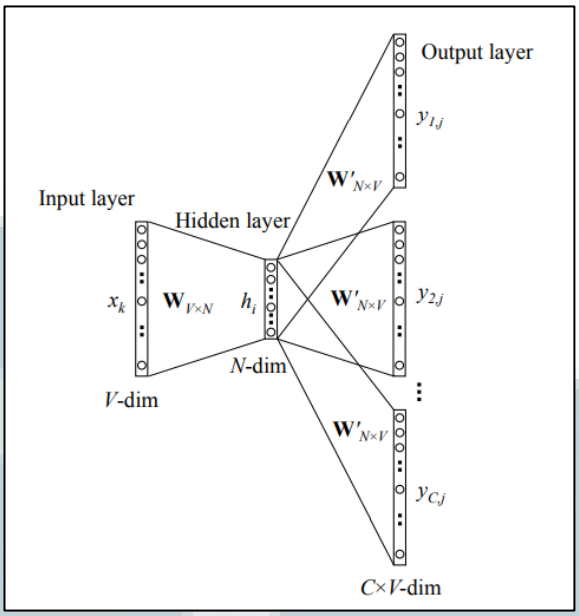
Dimana W_i, W_f, W_o ialah matriks yang terdiri atas *weight*, dan B_i, B_f, B_o ialah *bias* dari model LSTM. σ ialah fungsi *sigmoid* dan Θ merupakan operasi perkalian. x_t ialah input untuk sel LSTM yang berupa *word embedding*, h_t ialah

vector hidden layer. *Forget gate* ditandai dengan f_t, *input gate* ditandai dengan i_t, *output gate* ditandai dengan o_t, dan memori sel ditandai oleh c_t.

Keberadaan sel memori tersebut menjadikan LSTM sebagai model yang fleksibel dan mampu mengelola informasi secara efisien dalam jangkauan yang lebih luas dibandingkan LSTM, karena dapat berfokus pada informasi terpenting dalam konteks teks (Huang, 2016). Sebelum dimasukkan ke dalam model, teks harus dijadikan *vector* terlebih dahulu (*word embedding*).

2.9. *Word2vec: Skip-gram model*

Word2vec merupakan salah satu teknik yang digunakan untuk menghasilkan *vector* yang merepresentasikan kata atau kerap disebut dengan *word embedding* (Rong, 2014). Model yang kerap kali digunakan dalam membuat *word embedding* ialah *skip-gram model*, yakni model yang menghitung probabilitas konteks kata dari setiap target kata pada kalimat yang disesuaikan dengan jangkauan target kata (*window size*). Dalam hal ini, target kata ialah input, dan probabilitas konteks kata merupakan *output layer*. Sebelum diolah lebih lanjut, setiap kata direpresentasikan dalam bentuk *one-hot vector*. Adapun skema *skip-gram model* sebagai berikut:



Gambar 2.3 Skema pada model *skip-gram*

Input layer yang dimaksud ialah kata yang sudah diubah bentuknya menjadi *one-hot vector*, kemudian dikalikan dengan matriks *weight* pada *hidden layer* dengan dimensi (*V x N*), yakni *V* untuk jumlah kosakata (*vocabulary size*), dan *N* untuk jumlah neuron yang terdapat pada *hidden layer*. Hasil perkalian antara *one-hot vector* dengan matriks *weight hidden layer* menghasilkan *word vector*. *Word vector* tersebut kemudian digunakan kembali dalam perhitungan mencari probabilitas untuk kata-kata yang terdapat di dalam jangkauan target kata dalam satu kalimat (*window size*), dan menghasilkan matriks *W'* yang berisikan probabilitas dengan dimensi (*N x V*).

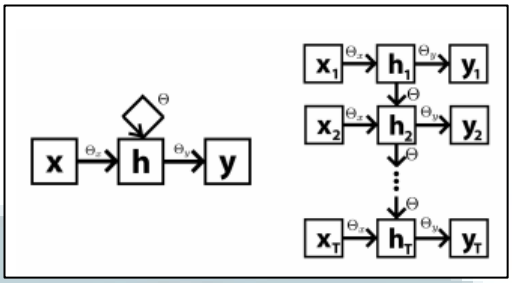
2.10. Penelitian Terdahulu

Sebelum merancang kedua model sentimen analisis, terdapat proses pengamatan atau proses evaluasi atas penelitian sebelumnya. Berikut penelitian yang dijadikan landasan serta bahan evaluasi bagi penelitian saat ini:

1. ***Long Short Term Memorys for Sentiment Analysis*** (Ruales, 2012)

   Tujuan utama dari penelitian ini yakni untuk mengklasifikasi secara biner sentimen terhadap *movie review*. Klasifikasi *movie review* sendiri didasarkan pada *rating* yang diberikan oleh *reviewer*, yakni *rating* 7/10 bahkan lebih untuk sentimen positif, dan *rating* 4/10 kebawah untuk sentimen negatif. *Datasets* yang digunakan berasal dari *movie review* IMDb, dengan jumlah masing-masing 25.000 untuk *training data*, dan 500 untuk *testing data*. Penelitian ini menggunakan model *Long Short Term Memory* (LSTM) dengan *Long Short Term Memory* (LSTM). Setelah dilatih, LSTM dapat menerima input berupa serangkaian *vector* $(x_{1, \ldots ,} x_T)$ dimana x_T ialah *vector* yang merepresentasikan sebuah kata. Hasil sebelumnya akan diteruskan ke dalam *hidden layer* untuk inputan berikutnya.

   Adapun *error rate* yang diperoleh yakni 0,134% dengan menggunakan LSTM Unit.



Gambar 2.1. Skema looping pada model LSTM

Sumber: (Ruales, 2012)

Adapun kesimpulan yang diperoleh dari penelitian ini yakni pengtesan LSTM dengan *layer sizes* yang beragam agar memperoleh hasil akurasi yang optimal.

2. ***And the Winner is ...: Bayesian Twitter-based Prediction on 2016 U.S. Presidential Election*** (Tunggawan, 2016)

Tujuan dari penelitian ini yakni untuk memprediksi kandidat presiden Amerika terbaik yang berasal dari partai Republik dan partai Demokratik berdasarkan sentimen *tweets* dengan menggunakan model *Naïve Bayes*. Adapun *datasets* yang digunakan yakni *tweets* sebanyak 33,708. Seluruh *tweets* yang digunakan memuat kata kunci "*#Election2016*". Sebelum merancang model *Naïve Bayes*, terdapat tahapan *data preprocessing*, diantaranya: (1) peniadaan *URLs* dan gambar, (2) filter *tweets* yang memiliki nama kandidat. *Hashtag*, *mention*, dan *retweets* tidak dihilangkan demi mempertahankan orisinalitas makna *tweets*.

Model dalam penelitian ini dilatih dengan menggunakan *Naive Bayes Classifier* yang diimplementasikan melalui *library Natural Language*

Toolkit pada *tools Python* dan dievaluasi dengan *10-fold cross validation*. Setelah model dilatih, model dapat memprediksi data *testing* dengan tingkat akurasi sebesar 95,8%. Adapun hasil prediksi kandidat yakni Bernie Sanders dari partai Demokratik dengan total 3,335 *tweets*, dan Ted Cruz dari partai Republik dengan total 1,432 *tweets*.

3. *Effective LSTMs for Target-Dependent Sentiment Classification* (Tang, 2015)

Penelitian ini bertujuan untuk menganalisa sentimen berdasarkan target kata tertentu dalam kalimat (*Target dependent sentiment classification*) dengan menguji coba 3 model: LSTM, TD-LSTM (*Target Dependent* LSTM), dan TC-LSTM (*Target Connection* LSTM). Sentimen diklasifikasikan ke dalam tiga kelas; netral, positif dan negatif. *Dataset* yang digunakan yakni *tweets* yang bersumber dari penelitian sebelumnya, dan terdiri atas 6.248 *tweets* untuk keperluan *training* dan 692 *tweets* untuk *testing*.

Adapun pengembangan model LSTM yang dilakukan yakni: (1) TD-LSTM, dan (2) TC-LSTM. Pada TD-LSTM, terdapat 2 model LSTM yang digunakan dalam menganalisa setiap *tweets*, yang disebut dengan $LSTM_L$ dan $LSTM_R$. Dalam hal ini, $LSTM_L$ digunakan untuk menganalisa konteks *tweets* pada bagian awal (sisi sebelah kiri), dan $LSTM_R$ digunakan untuk menganalisa konteks *tweets* pada bagian akhir. Hasil *hidden vector* terakhir dari masing-masing model kemudian digabungkan dan diteruskan menuju *softmax layer* untuk mengklasifikasi sentimen.

Model kedua, yakni TC-LSTM merupakan versi pengembangan dari TD-LSTM. Dalam menganalisa sentimen, TC-LSTM membagi *tweet* ke dalam tiga bagian; target kata, konteks awal *tweet*, dan konteks akhir *tweet*. *Target vector* diperoleh dari hasil rata-rata *vector* kata-kata yang terdapat di dalam *tweet*. Sama seperti TD-LSTM, TC-LSTM juga menggunakan 2 model LSTM dengan mekanisme yang hampir serupa dengan TD-LSTM. Perbedaannya terletak pada inputan, dimana TC-LSTM menerima inputan berupa kombinasi dari *word embedding* dan *target vector*, sedangkan TD-LSTM hanya menerima *word embedding*. Model dilatih dengan mengggunakan *cross entrophy* sebagai *loss function*. Hasil akurasi dari ketiga model ini juga dibandingkan dengan model lain dari penelitian sebelumnya. Model LSTM mampu melampaui hasil akurasi dari model SVM (63.4%), *Recursive* NN (63%), dan AdaLSTM (65.8%) dengan akurasi 66.5%. Kedua model pengembangan LSTM, yakni TD-LSTM dan TC-LSTM meraih hasil akurasi yang sedikit lebih unggul, yakni 70.8% dan 71.5%.