LAMPIRAN

Lampiran 1 Biodata Penulis

DATA PRIBADI

Nama	: Artha Ilma Imanidanantoyo
Nomor Induk Mahasiswa	: 1941727008
Tempat, Tanggal Lahir	: Kediri, 24 Desember 1997
Jenis Kelamin	: Laki- Laki
Agama	: Islam
Alamat	: Jalan Balowerti 1 no.24 Kota Kediri
No. Handphone	: 082132343949
Email	: arthailma12345@gmail.com

RIWAYAT PENDIDIKAN

2010	SDN BALOWERTI 1
2013	SMPN 5 KEDIRI
2016	SMAN 7 KEDIRI
2019	POLITEKNIK KEDIRI
2020	POLITEKNIK NEGERI MALANG

Lampiran 2 Source Code

Proses Login

Proses Login adalah proses dimana user harus memasukan email dan password utnuk masuk ke aplikasi. Implementasi login terdapat pada fungsi LoginController().

```
class LoginController extends Controller
{
    use AuthenticatesUsers {
        logout as performLogout;
    }
    protected $redirectTo = '/dashboard';
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }

    public function logout(Request $request)
    {
        $this->performLogout($request);
        return redirect()->route('login');
    }
}
```

Load Data Training

Load Data Training adalah proses yang digunakan untuk menampilkan data training yang telah melalui proses scraping dari twitter yang akan digunakan untuk proses laplace correction. Impelementasi load data training terdapat pada fungsi index().

```
public function index()
    {
        $tb_datatraining = DB::table('tb_datatraining')
        ->select(DB::raw('tb_datatraining.*'))
        ->get();
```

```
        // dd($tb_datatraining);
        return
view('pages/proses_training/datatraining/index',['tb_datatraining'=>$tb_datatrai
ning]);
    }
```

Preprocessing Data Training

Preprocessing Data Training adalah untuk menghilangkan noise atau kata yang tidak digunakan pada tweet. Implementasi preprocessing data training terdapat pada fungsi cleaning_proses(), casefolding_proses(), tokenizing_proses(), normalisasi_proses(), filtering_proses(), stemming_proses().

```
public function cleaning_proses(){
    $tb_preprocessing_deleted = tb_preprocessing::truncate();
    $tb_preprocessing_deleted->delete();
    $tb_scraper = DB::table('tb_scraper')
    ->select(DB::raw('tb_scraper.*'))
    ->get();
    $tb_preprocessing = DB::table('tb_preprocessing')
    ->select(DB::raw('tb_preprocessing.*, tb_scraper.text'))
    ->join('tb_scraper', 'tb_scraper.ID', '=', 'tb_preprocessing.id_tweet')
    ->get();
      foreach ($tb_scraper as $data) {
        $tweet_asli = $data->text;
        $string_tweet_1 = str_replace("."," ",$tweet_asli);
        //menghilangkan koma
        $string_tweet_2 = str_replace(","," ",$string_tweet_1);
        $string_tweet_3 = preg_replace('/(?=[^ ]*[^A-Za-z \'-])([^ ]*)(?:\\s+|$)/', ",
$string_tweet_2);
        $string_tweet_4 = str_replace("-"," ",$string_tweet_3);
        $tb_preprocessing_tambah = new tb_preprocessing;
        $tb_preprocessing_tambah->id_tweet = $data->ID;
        $tb_preprocessing_tambah->cleaning = $string_tweet_4;
        $tb_preprocessing_tambah->casefolding = '';
```

```
        $tb_preprocessing_tambah->normalisasi = '';
        $tb_preprocessing_tambah->filtering = '';
        $tb_preprocessing_tambah->stemming = '';
        $tb_preprocessing_tambah->save();
       }
     return back();
    }
```

```
public function casefolding_proses(){
     DB::table('tb_preprocessing')
          ->update(['tb_preprocessing.casefolding' => ""]);
     $table_preprocessing = DB::table('tb_preprocessing')
     ->select(DB::raw('tb_preprocessing.*'))
     ->get();
     foreach ($table_preprocessing as $data) {
       $tweet_cleaning = $data->cleaning;
       $string_tweet_1 = strtolower($data->cleaning);
       tb_preprocessing::where('id_tweet', $data->id_tweet)-
>update(array('casefolding' => $string_tweet_1));
     }
     return back();
    }
```

```
public function tokenizing_index(){
     $tb_preprocessing = DB::table('tb_preprocessing')
     ->select(DB::raw('tb_preprocessing.*, tb_scraper.usernameTweet'))
     ->join('tb_scraper', 'tb_scraper.ID', '=', 'tb_preprocessing.id_tweet')
     ->get();
     return
view('pages/proses_testing/preprocessing/tokenizing/index',['preprocessing'=>$
tb_preprocessing]);
    }
```

```
public function normalisasi_proses(){
    set_time_limit(300);
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.normalisasi' => ""]);
     $tb_normalisasi = DB::table('tb_normalisasi')
     ->select(DB::raw('tb_normalisasi.*'))
     ->get();
     $kata_gaul = [];
     $kata_normalisasi = [];
     foreach ($tb_normalisasi as $data) {
       $kata_gaul[] = '/\b'.".$data->kata_gaul."'\b/u';
       $kata_normalisasi[] = ".$data->normalisasi.";
     }
     $tb_preprocessing = DB::table('tb_preprocessing')
     ->select('tb_preprocessing.*')
     ->get();
     foreach ($tb_preprocessing as $data) {
       $tweet_normalisasi_final                                    =
preg_replace($kata_gaul,$kata_normalisasi,$data->casefolding);
       tb_preprocessing::where('id_tweet',                $data->id_tweet)-
>update(array('normalisasi' => $tweet_normalisasi_final));
     }
     return back();
   }
```

```
public function filtering_proses(){
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.filtering' => ""]);
    $kata_stopword = [];
    $tb_stopword = DB::table('tb_stopword')
    ->select(DB::raw('tb_stopword.*'))
    ->get();
```

```
    foreach ($tb_stopword as $data) {
      $kata_stopword[] = '/\b'.$data->kata.'\b/u';
    }
    $tb_preprocessing = DB::table('tb_preprocessing')
    ->select(DB::raw('tb_preprocessing.*'))
    ->get();
    foreach ($tb_preprocessing as $data) {
      $tweet_filter_final = preg_replace($kata_stopword,"",$data->normalisasi);
      tb_preprocessing::where('id_tweet', $data->id_tweet)-
>update(array('filtering' => $tweet_filter_final));
    }
    return back();
   }
```

```
public function stemming_proses(){
    set_time_limit(1200);
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.stemming' => ""]);
    //deleted postagging
    $tb_postaging_deleted = tb_postagging::truncate();
    $tb_postaging_deleted->delete();
    $katadasar = [];
    $tb_katadasar = DB::table('tb_katadasar')
    ->select(DB::raw('tb_katadasar.*'))
    ->get();
    foreach ($tb_katadasar as $data) {
      $katadasar[] = '/\b'.$data->kata.'\b/';
    }
    $tb_preprocessing = DB::table('tb_preprocessing')
    ->select(DB::raw('tb_preprocessing.*'))
    ->get();
    foreach ($tb_preprocessing as $data) {
```

```
        $stemmerFactory = new \Sastrawi\Stemmer\StemmerFactory();
        $stemmer  = $stemmerFactory->createStemmer();
        $tweet_filtering = $data->filtering;
        $tweet_stemming_final  = $stemmer->stem($tweet_filtering);
        // proses seleksi kata dasar (jika tidak ada di kbbi maka akan di hapus)
        $kalimat = $tweet_stemming_final;
        $tweet_stemming_final_1 = preg_replace($katadasar,"",$kalimat);
        $kalimat_pecah_stemming_final_1 = explode(" ",$tweet_stemming_final_1);
        $kalimat_pecah_stemming_final_2 = [];
        for($i=0; $i<count($kalimat_pecah_stemming_final_1); $i++){
          $kalimat_pecah_stemming_final_2[] = '/\b'.$kalimat_pecah_stemming_final_1[$i].'\b/';
        }
        $tweet_stemming_final_3 = preg_replace($kalimat_pecah_stemming_final_2,"",$kalimat);
        //untuk negasi
        $tweet_stemming_final_4 = preg_replace('/\b'.'tidak '.'\b/u','tidak', $tweet_stemming_final_3);
        //menghapus 4 spasi
        $tweet_stemming_final_5 = str_replace('    ',' ', $tweet_stemming_final_4);
        //menghapus 3 spasi
        $tweet_stemming_final_6 = str_replace('   ',' ', $tweet_stemming_final_5);
        //menghapus 2 spasi
        $tweet_stemming_final_7 = str_replace('  ',' ', $tweet_stemming_final_6);
        //menghapus spasi di depan
        $ptn = "/^ /";  // Regex
        $rpltxt = "";  // Replacement string
        $tweet_stemming_final_8 = preg_replace($ptn, $rpltxt, $tweet_stemming_final_7);
        //menghapus spasi di belakang
        $tweet_stemming_final_9 = rtrim($tweet_stemming_final_8, " ");
```

```
        tb_preprocessing::where('id_tweet', $data->id_tweet)-
>update(array('stemming' => $tweet_stemming_final_9));
        //mengganti kata 'tidak' menjadi 'tidak ' untuk proses postagging
(menentukan opini atau tidak)
        $stemming_postagging = str_replace('tidak','tidak
',$tweet_stemming_final_9);
       $tb_postagging_tambah_id_tweet = new tb_postagging;
       $tb_postagging_tambah_id_tweet->id_tweet = $data->id_tweet;
       $tb_postagging_tambah_id_tweet->stemming_postagging =
$stemming_postagging;
       $tb_postagging_tambah_id_tweet->tag = '';
       $tb_postagging_tambah_id_tweet->rule_deteksi = '';
       $tb_postagging_tambah_id_tweet->label_postagging_ekspektasi = '';
       $tb_postagging_tambah_id_tweet->status_postagging = '';
       $tb_postagging_tambah_id_tweet->save();
     }
     $tb_naive_bayes_deleted = tb_naivebayes::truncate();
     $tb_naive_bayes_deleted->delete();
     //mengambil data scraper yang ber status negatif dan positif
     $tb_postagging_tambah_id_twwet_ke_tb_naivebayes =
DB::table('tb_scraper')
     ->select(DB::raw('tb_scraper.*'))
     ->whereIn('tb_scraper.label_naive_bayes_realita',array('positif','negatif'))
     ->get();
     foreach ($tb_postagging_tambah_id_twwet_ke_tb_naivebayes as $data) {
       $tambah_naive_bayes = new tb_naivebayes;
       $tambah_naive_bayes->id_tweet = $data->ID;
       $tambah_naive_bayes->detail_bobot_naive_bayes_positif = '';
       $tambah_naive_bayes->detail_bobot_naive_bayes_negatif = '';
       $tambah_naive_bayes->hasil_naive_bayes_sentimen_positif = '';
       $tambah_naive_bayes->hasil_naive_bayes_sentimen_negatif = '';
       $tambah_naive_bayes->label_naive_bayes_ekspektasi  = '';
```

```
        $tambah_naive_bayes->status_naive_bayes       = '';
        $tambah_naive_bayes->save();
      }
      return back();
    }
```

Laplace Correction

Laplace Correction adalah proses pemobotan kata, kata yang dibobotkan berasal dari hasil preprocessing data training, Implementasi preprocessing data training terdapat pada fungsi proses_seleksi_kata_laplacecorrection() dan proses_pembobotan_laplacecorrection().

```
public function proses_seleksi_kata_laplacecorrection(){
    set_time_limit(1800);
    $tb_laplacecorrection_deleted = tb_laplacecorrection::truncate();
    $tb_laplacecorrection_deleted->delete();
    foreach($tb_preprocessingtraining as $data){
      $kata_stemming = explode(" ",$data->stemming);
      if($data->sentimen == 'positif'){
        for($i=0; $i<count($kata_stemming); $i++) {
          // echo ' '.$kata_stemming[$i].' ';
          $tb_laplacecorrection = DB::table('tb_laplacecorrection')
          ->select('tb_laplacecorrection.*')
          ->where('tb_laplacecorrection.kata', '=', $kata_stemming[$i])
          ->where('tb_laplacecorrection.sentimen', '=', 'positif')
          ->get();
          if(count($tb_laplacecorrection) == 0){
            $tambah_tb_laplacecorrection = new tb_laplacecorrection;
            $tambah_tb_laplacecorrection->kata = $kata_stemming[$i];
            $tambah_tb_laplacecorrection->jumlah_kata = 1;
            $tambah_tb_laplacecorrection->bobot = 0;
            $tambah_tb_laplacecorrection->sentimen = 'positif';
            $tambah_tb_laplacecorrection->save();
          }else{
```

```
        foreach ($tb_laplacecorrection as $data2) {
          tb_laplacecorrection::where('kata', $kata_stemming[$i])
          ->where('sentimen', 'positif')
          ->update(array('jumlah_kata' => $data2->jumlah_kata + 1));
        }
      }
    }
  }else if($data->sentimen == 'negatif'){
    for($i=0; $i<count($kata_stemming); $i++) {
      // echo ' '.$kata_stemming[$i].' ';
      $tb_laplacecorrection = DB::table('tb_laplacecorrection')
      ->select('tb_laplacecorrection.*')
      ->where('tb_laplacecorrection.kata', '=', $kata_stemming[$i])
      ->where('tb_laplacecorrection.sentimen', '=', 'negatif')
      ->get();
      if(count($tb_laplacecorrection) == 0){
        $tambah_tb_laplacecorrection = new tb_laplacecorrection;
        $tambah_tb_laplacecorrection->kata = $kata_stemming[$i];
        $tambah_tb_laplacecorrection->jumlah_kata = 1;
        $tambah_tb_laplacecorrection->bobot = 0;
        $tambah_tb_laplacecorrection->sentimen = 'negatif';
        $tambah_tb_laplacecorrection->save();
      }else{
        foreach ($tb_laplacecorrection as $data2) {
          // code...
          tb_laplacecorrection::where('kata', $kata_stemming[$i])
          ->where('sentimen', 'negatif')
          ->update(array('jumlah_kata' => $data2->jumlah_kata + 1));
        }
      }
    }
  }
```

```
    }
    $res=tb_laplacecorrection::where('kata',' ')->delete();
    $res=tb_laplacecorrection::where('kata','-')->delete();
    return back();
    // dd($tb_laplacecorrectiontraining);
   }
```

```
public function proses_pembobotan_laplacecorrection(){
    $tb_laplacecorrection = DB::table('tb_laplacecorrection')
    ->select('tb_laplacecorrection.*')
    ->get();
    $jumlah_kata_positif_perkata = DB::table('tb_laplacecorrection')
    ->select(DB::raw('COUNT(DISTINCT `tb_laplacecorrection`.`kata`) as jumlah_kata_positif_perkata'))
    $jumlah_kata_positif_perkata_int = '';
    foreach ($jumlah_kata_positif_perkata as $data) {
      $jumlah_kata_positif_perkata_int = $data->jumlah_kata_positif_perkata;
    }
    $jumlah_kata_negatif_perkata = DB::table('tb_laplacecorrection')
    ->select(DB::raw('COUNT(DISTINCT `tb_laplacecorrection`.`kata`) as jumlah_kata_negatif_perkata'))
    $jumlah_kata_negatif_perkata_int = '';
    foreach ($jumlah_kata_negatif_perkata as $data) {
      $jumlah_kata_negatif_perkata_int = $data->jumlah_kata_negatif_perkata;
    }
    $jumlah_kata_positif_semua_kata = DB::table('tb_laplacecorrection')
    ->select(DB::raw('sum(tb_laplacecorrection.jumlah_kata) as jumlah_kata_positif_semua_kata'))
    ->where('tb_laplacecorrection.sentimen','=', 'positif')
    ->get();
    $jumlah_kata_positif_semua_kata_int = '';
    foreach ($jumlah_kata_positif_semua_kata as $data) {
```

```
        $jumlah_kata_positif_semua_kata_int = (int)$data->jumlah_kata_positif_semua_kata;
      $jumlah_kata_negatif_semua_kata = DB::table('tb_laplacecorrection')
      ->select(DB::raw('sum(tb_laplacecorrection.jumlah_kata) as jumlah_kata_negatif_semua_kata'))
      ->where('tb_laplacecorrection.sentimen','=', 'negatif')
      ->get();
      $jumlah_kata_negatif_semua_kata_int = '';
      foreach ($jumlah_kata_negatif_semua_kata as $data) {
        $jumlah_kata_negatif_semua_kata_int = (int)$data->jumlah_kata_negatif_semua_kata;
      }
      foreach ($tb_laplacecorrection as $data) {
        if($data->sentimen == 'positif'){
          $rumus_laplace_correction_positif = (int)($data->jumlah_kata+1)/($jumlah_kata_positif_semua_kata_int+$jumlah_kata_positif_perkata_int);
          tb_laplacecorrection::where('id_laplacecorrection', $data->id_laplacecorrection)
          ->update(array('bobot' => $rumus_laplace_correction_positif));
        }else if($data->sentimen = 'negatif'){
          $rumus_laplace_correction_negatif = (int)($data->jumlah_kata+1)/($jumlah_kata_negatif_semua_kata_int+$jumlah_kata_negatif_perkata_int);
          tb_laplacecorrection::where('id_laplacecorrection', $data->id_laplacecorrection)
          ->update(array('bobot' => $rumus_laplace_correction_negatif));
        }
      }
      return back();
    }
```

Load Data Testing

Load Data Training adalah proses yang digunakan untuk menampilkan data training yang telah melalui proses scraping dari twitter yang akan digunakan untuk proses laplace correction. Impelementasi load data training terdapat pada fungsi index().

```
public function index()
    {
        $scraper = DB::table('tb_scraper')
        ->select(DB::raw('tb_scraper.*'))
        ->get();
        return view('pages/proses_testing/scraping/index',['scraper'=>$scraper]);
    }
```

Preprocessing Data Testing

Preprocessing Data testing adalah untuk menghilangkan noise atau kata yang tidak digunakan pada tweet. Implementasi preprocessing data testing terdapat pada fungsi cleaning_proses(), casefolding_proses(), tokenizing_proses(), normalisasi_proses(), filtering_proses(), stemming_proses().

```
public function cleaning_proses(){
      $tb_preprocessing_deleted = tb_preprocessing::truncate();
      $tb_preprocessing_deleted->delete();
      $tb_scraper = DB::table('tb_scraper')
      ->select(DB::raw('tb_scraper.*'))
      ->get();
      $tb_preprocessing = DB::table('tb_preprocessing')
      ->select(DB::raw('tb_preprocessing.*, tb_scraper.text'))
      ->join('tb_scraper', 'tb_scraper.ID', '=', 'tb_preprocessing.id_tweet')
      ->get();
        foreach ($tb_scraper as $data) {
          $tweet_asli = $data->text;
          $string_tweet_1 = str_replace("."," ",$tweet_asli);
          //menghilangkan koma
          $string_tweet_2 = str_replace(","," ",$string_tweet_1);
```

```
        $string_tweet_3 = preg_replace('/(?=[^ ]*[^A-Za-z \'-])([^ ]*)(?:\\s+|$)/', '',
$string_tweet_2);
        $string_tweet_4 = str_replace("-"," ",$string_tweet_3);
        $tb_preprocessing_tambah = new tb_preprocessing;
        $tb_preprocessing_tambah->id_tweet = $data->ID;
        $tb_preprocessing_tambah->cleaning = $string_tweet_4;
        $tb_preprocessing_tambah->casefolding = '';
        $tb_preprocessing_tambah->normalisasi = '';
        $tb_preprocessing_tambah->filtering = '';
        $tb_preprocessing_tambah->stemming = '';
        $tb_preprocessing_tambah->save();
       }
     return back();
    }
```

```
public function casefolding_proses(){
     DB::table('tb_preprocessing')
          ->update(['tb_preprocessing.casefolding' => ""]);
     $table_preprocessing = DB::table('tb_preprocessing')
     ->select(DB::raw('tb_preprocessing.*'))
     ->get();
     foreach ($table_preprocessing as $data) {
       $tweet_cleaning = $data->cleaning;
       $string_tweet_1 = strtolower($data->cleaning);
       tb_preprocessing::where('id_tweet',                    $data->id_tweet)-
>update(array('casefolding' => $string_tweet_1));
     }
     return back();
    }
```

```
public function tokenizing_index(){
     $tb_preprocessing = DB::table('tb_preprocessing')
```

```
    ->select(DB::raw('tb_preprocessing.*, tb_scraper.usernameTweet'))
    ->join('tb_scraper', 'tb_scraper.ID', '=', 'tb_preprocessing.id_tweet')
    ->get();
    return
view('pages/proses_testing/preprocessing/tokenizing/index',['preprocessing'=>$
tb_preprocessing]);
  }
```

```
public function normalisasi_proses(){
    set_time_limit(300);
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.normalisasi' => ""]);
     $tb_normalisasi = DB::table('tb_normalisasi')
     ->select(DB::raw('tb_normalisasi.*'))
     ->get();
     $kata_gaul = [];
     $kata_normalisasi = [];
     foreach ($tb_normalisasi as $data) {
      $kata_gaul[] = '/\b'.".$data->kata_gaul."'\b/u';
      $kata_normalisasi[] = ".$data->normalisasi.";
     }
     $tb_preprocessing = DB::table('tb_preprocessing')
     ->select('tb_preprocessing.*')
     ->get();
     foreach ($tb_preprocessing as $data) {
      $tweet_normalisasi_final                                    =
preg_replace($kata_gaul,$kata_normalisasi,$data->casefolding);
      tb_preprocessing::where('id_tweet',                $data->id_tweet)-
>update(array('normalisasi' => $tweet_normalisasi_final));
     }
     return back();
  }
```

```
public function filtering_proses(){
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.filtering' => ""]);
    $kata_stopword = [];
    $tb_stopword = DB::table('tb_stopword')
    ->select(DB::raw('tb_stopword.*'))
    ->get();
    foreach ($tb_stopword as $data) {
      $kata_stopword[] = '/\b'.$data->kata.'\b/u';
    }
    $tb_preprocessing = DB::table('tb_preprocessing')
    ->select(DB::raw('tb_preprocessing.*'))
    ->get();
    foreach ($tb_preprocessing as $data) {
      $tweet_filter_final = preg_replace($kata_stopword,"",$data->normalisasi);
      tb_preprocessing::where('id_tweet', $data->id_tweet)-
>update(array('filtering' => $tweet_filter_final));
    }
    return back();
  }
```

```
public function stemming_proses(){
    set_time_limit(1200);
    DB::table('tb_preprocessing')
        ->update(['tb_preprocessing.stemming' => ""]);
    //deleted postagging
    $tb_postaging_deleted = tb_postagging::truncate();
    $tb_postaging_deleted->delete();
    $katadasar = [];
    $tb_katadasar = DB::table('tb_katadasar')
    ->select(DB::raw('tb_katadasar.*'))
```

```
    ->get();
    foreach ($tb_katadasar as $data) {
      $katadasar[] = '/\b'.$data->kata.'\b/';
    }
    $tb_preprocessing = DB::table('tb_preprocessing')
    ->select(DB::raw('tb_preprocessing.*'))
    ->get();
    foreach ($tb_preprocessing as $data) {
        $stemmerFactory = new \Sastrawi\Stemmer\StemmerFactory();
        $stemmer  = $stemmerFactory->createStemmer();
        $tweet_filtering = $data->filtering;
        $tweet_stemming_final  = $stemmer->stem($tweet_filtering);
        // proses seleksi kata dasar (jika tidak ada di kbbi maka akan di hapus)
        $kalimat = $tweet_stemming_final;
        $tweet_stemming_final_1 = preg_replace($katadasar,"",$kalimat);
        $kalimat_pecah_stemming_final_1 = explode(" ",$tweet_stemming_final_1);
        $kalimat_pecah_stemming_final_2 = [];
        for($i=0; $i<count($kalimat_pecah_stemming_final_1); $i++){
          $kalimat_pecah_stemming_final_2[] = '/\b'.$kalimat_pecah_stemming_final_1[$i].'\b/';
        }
        $tweet_stemming_final_3 = preg_replace($kalimat_pecah_stemming_final_2,"",$kalimat);
        //untuk negasi
        $tweet_stemming_final_4 = preg_replace('/\b'.'tidak '.'\b/u','tidak', $tweet_stemming_final_3);
        //menghapus 4 spasi
        $tweet_stemming_final_5 = str_replace('    ',' ', $tweet_stemming_final_4);
        //menghapus 3 spasi
        $tweet_stemming_final_6 = str_replace('   ',' ', $tweet_stemming_final_5);
        //menghapus 2 spasi
```

```
        $tweet_stemming_final_7 = str_replace('  ',' ', $tweet_stemming_final_6);
        //menghapus spasi di depan
        $ptn = "/^ /";  // Regex
        $rpltxt = "";  // Replacement string
        $tweet_stemming_final_8 = preg_replace($ptn, $rpltxt, $tweet_stemming_final_7);
        //menghapus spasi di belakang
        $tweet_stemming_final_9 = rtrim($tweet_stemming_final_8, " ");
        tb_preprocessing::where('id_tweet', $data->id_tweet)->update(array('stemming' => $tweet_stemming_final_9));
        //mengganti kata 'tidak' menjadi 'tidak ' untuk proses postagging (menentukan opini atau tidak)
        $stemming_postagging = str_replace('tidak','tidak ',$tweet_stemming_final_9);
       $tb_postagging_tambah_id_tweet = new tb_postagging;
       $tb_postagging_tambah_id_tweet->id_tweet = $data->id_tweet;
       $tb_postagging_tambah_id_tweet->stemming_postagging = $stemming_postagging;
       $tb_postagging_tambah_id_tweet->tag = '';
       $tb_postagging_tambah_id_tweet->rule_deteksi = '';
       $tb_postagging_tambah_id_tweet->label_postagging_ekspektasi = '';
       $tb_postagging_tambah_id_tweet->status_postagging = '';
       $tb_postagging_tambah_id_tweet->save();
     }
     $tb_naive_bayes_deleted = tb_naivebayes::truncate();
     $tb_naive_bayes_deleted->delete();
     //mengambil data scraper yang ber status negatif dan positif
     $tb_postagging_tambah_id_twwet_ke_tb_naivebayes = DB::table('tb_scraper')
     ->select(DB::raw('tb_scraper.*'))
     ->whereIn('tb_scraper.label_naive_bayes_realita',array('positif','negatif'))
     ->get();
```

```
    foreach ($tb_postagging_tambah_id_twwet_ke_tb_naivebayes as $data) {
      $tambah_naive_bayes = new tb_naivebayes;
      $tambah_naive_bayes->id_tweet = $data->ID;
      $tambah_naive_bayes->detail_bobot_naive_bayes_positif = '';
      $tambah_naive_bayes->detail_bobot_naive_bayes_negatif = '';
      $tambah_naive_bayes->hasil_naive_bayes_sentimen_positif = '';
      $tambah_naive_bayes->hasil_naive_bayes_sentimen_negatif = '';
      $tambah_naive_bayes->label_naive_bayes_ekspektasi  = '';
      $tambah_naive_bayes->status_naive_bayes       = '';
      $tambah_naive_bayes->save();
    }
    return back();
  }
```

Algoritma Viterbi

Algoritma Viterbi adalah algoritma POSTagging yang berguna untuk memberi tanda tag kata pada tweet. Implementasi Algoritma Viterbi terdapat pada fungsi opinion_postaggin_viterbi() dan opinion_detection_rule_proses().

```
public function opinion_postagging_viterbi(){
    set_time_limit(1800);
    DB::table('tb_postagging')
        ->update(['tag' => ""]);
    $tb_postagging = DB::table('tb_postagging')
    ->select(DB::raw('tb_postagging.*, tb_preprocessing.stemming'))
    ->join('tb_preprocessing', 'tb_preprocessing.id_tweet', '=',
'tb_postagging.id_tweet')
    ->get();
    foreach ($tb_postagging as $data) {
      $tweet_ganti_spasi_4      =      str_replace('                ','%20',$data-
>stemming_postagging);
      $tweet_ganti_spasi_3 = str_replace('   ','%20',$tweet_ganti_spasi_4);
      $tweet_ganti_spasi_2 = str_replace('  ','%20',$tweet_ganti_spasi_3);
      $tweet_ganti_spasi_1 = str_replace(' ','%20',$tweet_ganti_spasi_2);
```

```
        $getdataurl =
file_get_contents("http://localhost/IndoPOSTag/Home/postager?tweet=$tweet_
ganti_spasi_1");
        $viterbi_final_1 = str_replace(['[',']',""],",$getdataurl);
        $viterbi_final_2 = str_replace(',',' ',$viterbi_final_1);
        tb_postagging::where('id_tweet',  $data->id_tweet)->update(array('tag'  =>
$viterbi_final_2));
      }
      return back();
    }
```

```
public function opinion_detection_rule_proses(){
      DB::table('tb_postagging')
          ->update(['label_postagging_ekspektasi' => ""]);
      $tb_rule_opini = DB::table('tb_rule_opini')
      ->select(DB::raw('tb_rule_opini.*'))
      ->get();
      $rule = [];
      $rule_preg_match_all = [];
      foreach ($tb_rule_opini as $data) {
        $rule[] = $data->rule.";
        $rule_preg_match_all[] = '|'.$data->rule."."|';
      }
      $tb_postagging = DB::table('tb_postagging')
      ->select(DB::raw('tb_postagging.*, tb_preprocessing.*'))
      -
>join('tb_preprocessing','tb_preprocessing.id_tweet','=','tb_postagging.id_tweet'
)
      ->get();
      foreach ($tb_postagging as $data) {
        $string = $data->tag;
        if ($this->strposa($string, $rule, 1)) {
```

```
            tb_postagging::where('id_tweet', $data->id_tweet)-
>update(array('label_postagging_ekspektasi' => 'opini'));
        } else {
            tb_postagging::where('id_tweet', $data->id_tweet)-
>update(array('label_postagging_ekspektasi' => 'bukan opini'));
        }
        $rule_preg_match_all_string = implode(' ',$rule_preg_match_all);
        $rule_preg_match_all_string_2 = '#\b('.$rule_preg_match_all_string.')\b#';
        if(preg_match_all($rule_preg_match_all_string_2, $string, $matches)){
          $matches_string = implode(' ',$matches[0]);
          $matches_string_2 = trim(preg_replace('/\s+/', ' ', $matches_string));
          tb_postagging::where('id_tweet', $data->id_tweet)-
>update(array('rule_deteksi' => $matches_string_2));
        }
        $tb_postagging = DB::table('tb_postagging')
        ->select(DB::raw('tb_postagging.*, tb_scraper.*'))
        ->join('tb_scraper','tb_scraper.ID','=','tb_postagging.id_tweet')
        ->get();
        foreach ($tb_postagging as $data) {
          if($data->label_postagging_ekspektasi == $data-
>label_postagging_realita){
            tb_postagging::where('id_tweet', $data->id_tweet)-
>update(array('status_postagging' => 'cocok'));
          }else{
            tb_postagging::where('id_tweet', $data->id_tweet)-
>update(array('status_postagging' => 'tidak cocok'));
          }
        }
      }
      return back();
    }
```

Algoritma Naïve Bayes

Algoritma Naïve Bayes adalah algoritma klasifikasi untuk memisahkan antara opini postif dan opini negatif. Implementasi Algoritma Naïve Bayes terdapat pada fungsi naïve_bayes_status().

```
function naive_bayes_status(){
    $tb_naivebayes = DB::table('tb_naivebayes')
    ->select(DB::raw('tb_naivebayes.*,tb_scraper.*'))
    ->join('tb_scraper','tb_scraper.ID','=','tb_naivebayes.id_tweet')
    ->get();
    foreach ($tb_naivebayes as $data) {
     if($data->label_naive_bayes_ekspektasi == $data->label_naive_bayes_realita){
       tb_naivebayes::where('id_tweet', $data->id_tweet)->update(array('status_naive_bayes' => 'cocok'));
      }else{
       tb_naivebayes::where('id_tweet', $data->id_tweet)->update(array('status_naive_bayes' => 'tidak cocok'));
      }
    }
  }
```

Pengujian Sistem

Pada proses pengujian sistem dilakukan perhitungan untuk *accuracy*, *precision* positif, *precision* negatif, *recall* positif dan *recall* negatif dari hasil proses *testing*. Dengan membandingkan antara hasil klasifikasi sistem dengan hasil klasifikasi dari pelabelan manual. Implementasi Pengujian terdapat pada fungsi index().

```
public function index()
$tb_postagging_label_postagging_ekspektasi_samadengan_opini_status_postagging_samadengan_cocok = DB::table('tb_postagging')
      ->select(DB::raw('tb_postagging.*','tb_scraper.*'))
      ->where('tb_postagging.label_postagging_ekspektasi','=','opini')
```

```
        ->where('tb_postagging.status_postagging','=','cocok')
        ->join('tb_scraper','tb_scraper.ID','=','tb_postagging.id_tweet')
        ->get();
        $ekspektasi_opini_cocok =
count($tb_postagging_label_postagging_ekspektasi_samadengan_opini_status_
postagging_samadengan_cocok);
$tb_postagging_label_postagging_ekspektasi_samadengan_opini_status_postag
ging_samadengan_tidak_cocok = DB::table('tb_postagging')
        ->select(DB::raw('tb_postagging.*','tb_scraper.*'))
        ->where('tb_postagging.label_postagging_ekspektasi','=','opini')
        ->where('tb_postagging.status_postagging','=','tidak cocok')
        ->join('tb_scraper','tb_scraper.ID','=','tb_postagging.id_tweet')
        ->get();
        $ekspektasi_opini_tidak_cocok =
count($tb_postagging_label_postagging_ekspektasi_samadengan_opini_status_
postagging_samadengan_tidak_cocok);
        $tb_scraper_label_postagging_realita_samadengan_opini =
DB::table('tb_scraper')
        ->select(DB::raw('tb_scraper.*'))
        ->where('tb_scraper.label_postagging_realita','=','opini')
        ->get();
        $realita_opini =
count($tb_scraper_label_postagging_realita_samadengan_opini);
$tb_postagging_label_postagging_ekspektasi_samadengan_bukan_opini_status
_postagging_samadengan_cocok = DB::table('tb_postagging')
        ->select(DB::raw('tb_postagging.*','tb_scraper.*'))
        ->where('tb_postagging.label_postagging_ekspektasi','=','bukan opini')
        ->where('tb_postagging.status_postagging','=','cocok')
        ->join('tb_scraper','tb_scraper.ID','=','tb_postagging.id_tweet')
        ->get();
```

```
        $ekspektasi_bukan_opini_cocok =
count($tb_postagging_label_postagging_ekspektasi_samadengan_bukan_opini
_status_postagging_samadengan_cocok);
$tb_postagging_label_postagging_ekspektasi_samadengan_bukan_opini_status
_postagging_samadengan_tidak_cocok = DB::table('tb_postagging')
        ->select(DB::raw('tb_postagging.*','tb_scraper.*'))
        ->where('tb_postagging.label_postagging_ekspektasi','=','bukan opini')
        ->where('tb_postagging.status_postagging','=','tidak cocok')
        ->join('tb_scraper','tb_scraper.ID','=','tb_postagging.id_tweet')
        ->get();
        $ekspektasi_bukan_opini_tidak_cocok =
count($tb_postagging_label_postagging_ekspektasi_samadengan_bukan_opini
_status_postagging_samadengan_tidak_cocok);
        $tb_scraper_label_postagging_realita_samadengan_bukan_opini =
DB::table('tb_scraper')
        ->select(DB::raw('tb_scraper.*'))
        ->where('tb_scraper.label_postagging_realita','=','bukan opini')
        ->get();
        $realita_bukan_opini =
count($tb_scraper_label_postagging_realita_samadengan_bukan_opini);
dd($tb_postagging_label_postagging_ekspektasi_samadengan_bukan_opini_sta
tus_postagging_samadengan_tidak_cocok);
$tb_postagging_label_naivebayes_ekspektasi_samadengan_positif_status_naiv
ebayes_samadengan_cocok = DB::table('tb_naivebayes')
        ->select(DB::raw('tb_naivebayes.*','tb_scraper.*'))
        ->where('tb_naivebayes.label_naive_bayes_ekspektasi','=','positif')
        ->where('tb_naivebayes.status_naive_bayes','=','cocok')
        ->join('tb_scraper','tb_scraper.ID','=','tb_naivebayes.id_tweet')
        ->get();
        $ekspektasi_positif_cocok =
count($tb_postagging_label_naivebayes_ekspektasi_samadengan_positif_status
_naivebayes_samadengan_cocok);
```

```
$tb_postagging_label_naivebayes_ekspektasi_samadengan_positif_status_naivebayes_samadengan_tidak_cocok = DB::table('tb_naivebayes')
    ->select(DB::raw('tb_naivebayes.*','tb_scraper.*'))
    ->where('tb_naivebayes.label_naive_bayes_ekspektasi','=','positif')
    ->where('tb_naivebayes.status_naive_bayes','=','tidak cocok')
    ->join('tb_scraper','tb_scraper.ID','=','tb_naivebayes.id_tweet')
    ->get();
    $ekspektasi_positif_tidak_cocok = count($tb_postagging_label_naivebayes_ekspektasi_samadengan_positif_status_naivebayes_samadengan_tidak_cocok);
    $tb_scraper_label_naive_bayes_realita_samadengan_positif = DB::table('tb_scraper')
    ->select(DB::raw('tb_scraper.*'))
    ->where('tb_scraper.label_naive_bayes_realita','=','positif')
    ->get();
    $realita_positif = count($tb_scraper_label_naive_bayes_realita_samadengan_positif);
$tb_postagging_label_naivebayes_ekspektasi_samadengan_negatif_status_naivebayes_samadengan_cocok = DB::table('tb_naivebayes')
    ->select(DB::raw('tb_naivebayes.*','tb_scraper.*'))
    ->where('tb_naivebayes.label_naive_bayes_ekspektasi','=','negatif')
    ->where('tb_naivebayes.status_naive_bayes','=','cocok')
    ->join('tb_scraper','tb_scraper.ID','=','tb_naivebayes.id_tweet')
    ->get();
    $ekspektasi_negatif_cocok = count($tb_postagging_label_naivebayes_ekspektasi_samadengan_negatif_status_naivebayes_samadengan_cocok);
$tb_postagging_label_naivebayes_ekspektasi_samadengan_negatif_status_naivebayes_samadengan_tidak_cocok = DB::table('tb_naivebayes')
    ->select(DB::raw('tb_naivebayes.*','tb_scraper.*'))
    ->where('tb_naivebayes.label_naive_bayes_ekspektasi','=','negatif')
    ->where('tb_naivebayes.status_naive_bayes','=','tidak cocok')
```

```
        ->join('tb_scraper','tb_scraper.ID','=','tb_naivebayes.id_tweet')
        ->get();
        $ekspektasi_negatif_tidak_cocok =
count($tb_postagging_label_naivebayes_ekspektasi_samadengan_negatif_statu
s_naivebayes_samadengan_tidak_cocok);
        $tb_scraper_label_naive_bayes_realita_samadengan_negatif =
DB::table('tb_scraper')
        ->select(DB::raw('tb_scraper.*'))
        ->where('tb_scraper.label_naive_bayes_realita','=','negatif')
        ->get();
        $realita_negatif =
count($tb_scraper_label_naive_bayes_realita_samadengan_negatif);
      return view('pages/proses_akurasi/index',[
      'ekspektasi_opini_cocok'=>$ekspektasi_opini_cocok,
      'ekspektasi_opini_tidak_cocok'=>$ekspektasi_opini_tidak_cocok,
      'realita_opini'=>$realita_opini,
      'ekspektasi_bukan_opini_cocok'=>$ekspektasi_bukan_opini_cocok,
'ekspektasi_bukan_opini_tidak_cocok'=>$ekspektasi_bukan_opini_tidak_cocok
,
      'realita_bukan_opini'=>$realita_bukan_opini,
      'ekspektasi_positif_cocok'=>$ekspektasi_positif_cocok,
      'ekspektasi_positif_tidak_cocok'=>$ekspektasi_positif_tidak_cocok,
      'realita_positif'=>$realita_positif,
      'ekspektasi_negatif_cocok'=>$ekspektasi_negatif_cocok,
      'ekspektasi_negatif_tidak_cocok'=>$ekspektasi_negatif_tidak_cocok,
      'realita_negatif'=>$realita_negatif]);
    }
```