BAB IV
METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan metodologi penelitian yang digunakan untuk melakukan penelitian beserta penjelasan. Metodologi penelitian dibagi menjadi beberapa tahap, yaitu pengumpulan data dan *tool*, pembuatan model analisis sentimen, menguji dan membandingkan hasil analisis sentimen, analisis sentimen menggunakan data asli, agregasi, dan implementasi sistem *electoral college*. Diagram alir metodologi penelitian dapat dilihat pada Lampiran 1. Berikut akan dijelaskan penjelasan lengkap untuk masing-masing tahap dalam metodologi penelitian.

4.1. Pengumpulan Data dan *Tool*

Penelitian ini menggunakan data dan *tool* yang didapatkan dari sumber *online* maupun *offline*. Data dan *tool* tersebut digunakan sebagai komponen utama dalam menjalankan proses implementasi dan pengujian dalam metodologi penelitian. Data yang digunakan pada metodologi penelitian berupa kamus singkatan, daftar negara bagian dan jumlah *elector* yang termasuk dalam sistem *electoral college,* data latih, data uji, dan data asli. *Tool* yang digunakan pada metodologi penelitian antara lain: daftar *stop words*, leksikon SentiWordNet, dan leksikon AFINN-111. Data dan *tool* yang disebutkan akan dijelaskan secara rinci dalam segmen berikut:

4.1.1. Kamus Singkatan

Kamus singkatan berisi daftar singkatan yang sering digunakan dalam media sosial dan arti atau kepanjangan dari singkatan tersebut. Singkatan dan artinya dikumpulkan dengan cara mengekstrak informasi (*scraping*) dari sumber *online* pada situs All Acronyms (Acronyms, 2016). Penelitian ini menerapkan beberapa kondisi untuk menyaring singkatan yang tidak digunakan dengan tujuan untuk mengurangi ambiguitas. Singkatan yang tidak dapat digunakan adalah yang memenuhi kondisi berikut: singkatan yang hanya terdiri dari satu karakter saja (contoh: "2" dapat diartikan sebagai *to*, "C" dapat diartikan sebagai *see*, dan "Y" dapat diartikan sebagai *why*) dan singkatan yang membentuk kata dalam Bahasa Inggris (contoh: "HOPE" yang merupakan singkatan dari *have only positive expectations* dan merupakan kata dalam Bahasa Inggris yang berarti harapan).

4.1.2. Daftar Negara Bagian

Implementasi *electoral college* membutuhkan daftar negara bagian Amerika Serikat yang termasuk dalam distribusi *electoral votes*. Daftar negara bagian dan alokasi *electoral votes*-nya yang digunakan dalam penelitian ini dikumpulkan dengan mengekstrak informasi (*scraping*) dari sumber *online* (U.S. Census Bureau, 2010).

4.1.3. Data Latih dan Data Uji

Analisis sentimen dengan model *learning-based* atau *machine learning* seperti Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes

membutuhkan data latih untuk dapat memprediksi sentimen. Selain itu, diperlukan pula data uji untuk menguji hasil tingkat keakuratan dari model yang digunakan. Sentiment140 (Go, et al., 2009) telah menyediakan *corpus* untuk kepentingan akademik yang berisi data *tweet* beserta sentimennya. *Corpus* tersebut memuat 1.600.000 data latih (dibagi menjadi 800.000 data positif dan 800.000 data negatif) dan 497 data uji (dibagi menjadi 181 data positif, 177 data negatif, dan 139 data netral). Penelitian ini hanya menggunakan 26.000 data latih (13.000 data positif dan 13.000 data negatif) untuk melatih model analisis sentimen. Pengujian model analisis sentimen hanya menggunakan 358 data uji yang data positif dan data negatif, karena data latih yang digunakan hanya menyediakan data dengan kelas sentimen positif dan negatif saja.

4.1.4. Data Asli

Prediksi pemilihan presiden Amerika Serikat 2016 dilakukan dengan menggunakan data asli yang dikumpulkan dari Twitter dan Facebook selama tahun 2016 dengan batas waktu hingga penelitian ini selesai atau pemilihan presiden telah selesai. Target minimal jumlah data yang harus dikumpulkan dalam penelitian ini adalah 2000. Data yang disimpan harus memiliki teks status, tanggal status dibuat, penulis status, dan lokasi penulis status tersebut. Teks status akan diproses pada pra-proses dan analisis sentimen untuk menentukan kandidat pilihan penulis status. Lokasi penulis digunakan untuk menentukan kandidat yang akan dipilih pada suatu negara bagian, yang selanjutnya akan dapat menentukan kandidat yang memenangkan negara bagian tersebut. Data dari Twitter didapatkan dengan cara

mengakses REST API Twitter. Parameter yang diterapkan saat mengakses REST API Twitter, antara lain: teks harus dalam Bahasa Inggris, rentang waktu antara 1 Januari 2015 hingga saat *request* dilakukan, dan kata kunci yang berhubungan dengan pemilihan presiden Amerika Serikat 2016. Pada penelitian ini, kata kunci yang digunakan adalah "Hillary Clinton", "Donald Trump", "democrat", dan "republican". *Response* yang diberikan oleh REST API Twitter telah menyediakan data yang dibutuhkan untuk penelitian (teks status, tanggal status dibuat, penulis status, dan lokasi penulis status). Data dari Facebook dikumpulkan dari komentar pada beberapa *post* yang dituliskan pada halaman kandidat presiden dan partai yang berpartisipasi dalam pemilihan presiden Amerika Serikat 2016. Data tersebut didapatkan dari Graph API Facebook, dengan memberikan parameter *post_id* yang terdaftar pada *post* yang ditulis pada halaman kandidat presiden dan partai. *Response* yang diberikan oleh Graph API Facebook hanya menyediakan teks komentar, waktu komentar dibuat, dan penulisnya saja. Peneliti harus membuka atau mengakses akun Facebook penulis komentar untuk mendapatkan lokasi pengguna.

4.1.5. Daftar *Stop Words*

Stop words adalah kata-kata umum yang tidak memiliki arti yang penting (Mostafa, 2013). Contoh *stop words* antara lain: *a, and, from, of, there, you,* dan lain sebagainya. Pada penelitian ini, daftar *stop words* didapatkan dari sumber *online* Ranks.nl (Doyle, 2014). Daftar tersebut berisi 665 kata umum yang ditulis dalam Bahasa Inggris.

4.1.6. Leksikon SentiWordNet

SentiWordNet merupakan leksikon yang digunakan untuk *opinion mining*, serta digunakan untuk membantu mengklasifikasikan sentimen (Baccianella, et al., 2010). SentiWordNet berisi *synset* (*synonym sets*) dari WordNet yang memiliki nilai sentimen *"positivity", "negativity"*, dan *"objectivity"* sebesar nol hingga satu. Apabila ketiga nilai sentimen tersebut dijumlahkan, hasilnya harus bernilai satu. Pada penelitian ini, versi SentiWordNet yang digunakan adalah versi resmi yang terbaru yaitu versi 3.0 yang dapat diunduh dari situs http://sentiwordnet.isti.cnr.it/.

4.1.7. Leksikon AFINN-111

AFINN merupakan daftar kata-kata dalam Bahasa Inggris yang telah diberi nilai *integer* antara -5 (negatif) hingga 5 (positif) (Nielsen, 2011). AFINN-111 merupakan versi terbaru yang memuat 2477 kata dan frase. Kata-kata yang terdaftar pada AFINN diberi label secara manual oleh Finn Årup Nielsen.

4.2. Pembuatan Model Analisis sentimen

Proses analisis sentimen mencakup dua proses yaitu pra-proses dan klasifikasi sentimen. Pra-proses merupakan proses untuk mempersiapkan data menjadi format yang lebih mudah untuk diproses pada klasifikasi sentimen. Klasifikasi sentimen merupakan proses menentukan kelas sentimen data. Setiap model analisis sentimen memerlukan tahapan pra-proses yang berbeda. Berikut akan dijelaskan model analisis sentimen yang digunakan dalam penelitian ini:

4.2.1. Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes

Tahap pra-proses pada model analisis sentimen dibagi menjadi tujuh proses, yaitu: menghapus elemen HTML dan *mention* Twitter dengan format @username, tokenisasi, mengartikan singkatan, filtrasi, lematisasi atau *stemming*, dan membuat unigram dan bigram. Setiap tahapan pra-proses harus diterapkan pada data latih, data uji, dan data asli agar format yang dihasilkan sama. Langkah yang dilakukan setelah pra-proses adalah klasifikasi sentimen dengan menggunakan algoritma Multinomial Naïve Bayes atau Binarized Multinomial Naïve Bayes.

Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes merupakan algoritma klasifikasi dengan menggunakan metode *machine learning*. Maksud dari *machine learning* adalah menanamkan pengetahuan pada mesin atau komputer agar komputer dapat melakukan pekerjaan layaknya manusia. Tugas dari algoritma Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes dalam analisis sentimen adalah untuk mengklasifikasikan data set yang ke dalam kelas positif, negatif, atau netral sesuai dengan pengetahuan yang ditanamkan dengan menggunakan data latih. Data set dan data latih yang digunakan dalam penelitian ini merupakan data berjenis teks, sehingga pra-proses menjadi bagian yang krusial sebelum melakukan klasifikasi.

Penelitian ini menerapkan batasan kelas sentimen dalam klasifikasi menggunakan algoritma *machine learning*. Batasan tersebut adalah algoritma Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes hanya dapat mengklasifikasikan data uji atau data asli ke dalam kelas positif dan negatif saja. Batasan tersebut diterapkan karena data latih yang digunakan hanya menyediakan

data dengan kelas sentimen positif dan negatif, sehingga sangat kecil kemungkinan untuk menghasilkan kelas sentimen netral. Berikut adalah penjelasan untuk masing-masing tahapan dalam pra-proses serta penerapan algoritma Multinomial Naïve Bayes dan Binarized Multinomial Naïve Bayes:

4.2.1.1. Menghapus elemen HTML dan Twitter *mention*

Pada tahap ini seluruh elemen HTML dan *mention* akan dihapus karena tidak memiliki arti khusus saat klasifikasi sentimen. Gambar 4.1 adalah kode program untuk menghapus elemen HTML dan *mention*.

```
public string clearHtmlElements(string sentence)
{
  //remove xml/html tag
  sentence = Regex.Replace(sentence, "<.*?>", "");

  //remove url
  sentence = Regex.Replace(sentence, @"http[^\s]+",
"");

  //remove mentions in tweets
  sentence = Regex.Replace(sentence, @"@(\w+)", "");

  return sentence;
}
```

Gambar 4.1. Fungsi Hapus Elemen HTML dan *Twitter Mention*

Pada Gambar 4.1, setiap kata atau frase pada kalimat masukan yang diawali dan diakhiri dengan kurung siku '<' dan '>' akan dihapus dan situs atau *Uniform Resource Locator* (URL) juga akan dihapus. Setelah elemen HTML dihapus, langkah selanjutnya adalah menghapus *mention*, yaitu kata yang diawali dengan tanda @, contoh: @username.

4.2.1.2. Tokenisasi

Tokenisasi merupakan proses untuk mengubah kalimat input menjadi kumpulan *token* atau *bag of words*. Gambar 4.2 merupakan kode untuk melakukan tokenisasi.

```
public string[] tokenize(string sentence)
{
  sentence = sentence.ToLower();
  char[] delimiterChars = { ' ', ',', '.', ':', '\t' };
  return sentence.Split(delimiterChars);
}
```

Gambar 4.2. Fungsi Tokenisasi

Perintah yang dilakukan menurut Gambar 4.2 adalah kalimat masukan akan diubah menjadi huruf kecil (*lower case*), setelah itu akan dipisahkan berdasarkan karakter yang disebutkan pada *array* `delimiterChars`, yaitu spasi, koma, titik, titik dua, dan *tab*. Hasilnya adalah *array* bertipe data *string* yang berisi kumpulan *token* atau *term* yang ada pada kalimat atau teks masukan.

4.2.1.3. Mengartikan Singkatan

Token yang terdaftar sebagai singkatan akan diartikan menurut kamus singkatan yang telah tersimpan. Isi kamus singkatan dapat dilihat pada Lampiran 2. Sebelum mengartikan singkatan, karakter non-alfanumerik pada *token* harus dihapus karena dalam kamus singkatan tidak terdapat karakter non-alfanumerik. Apabila suatu *token* terdaftar dalam kamus singkatan, maka *token* tersebut akan digantikan dengan arti singkatannya. Arti singkatan tidak selalu dalam bentuk sebuah kata, namun dapat berbentuk frase. Gambar 4.3 merupakan kode yang digunakan untuk mengartikan singkatan yang ditemukan dalam kumpulan *token*.

```
public string[] expand_abbreviations(string[] tokens)
{
  if (abbreviations_dictionary != null &&
abbreviations_dictionary.Count() > 0)
  {
    for (int i = 0; i < tokens.Count(); i++)
    {
      string tkn = Regex.Replace(tokens[i], @"[^0-9a-
zA-Z\']+", "");
      if (abbreviations_dictionary.ContainsKey(tkn))
        tokens[i] = abbreviations_dictionary[tkn];
    }
    tokens = string.Join(" ", tokens).Split(' ');
  }
  return tokens;
}
```

Gambar 4.3. Fungsi Mengartikan Singkatan

Pada fungsi mengartikan singkatan (Gambar 4.3), variabel `abbreviations_dictionary` merupakan objek *Dictionary* dengan *key* bertipe data *string* yang berisi singkatan dalam huruf kecil (*lower case*) dan *value* bertipe data *string* yang berisi arti singkatan dalam huruf kecil (*lower case*). Pada akhir program, *token* akan disatukan kembali menjadi sebuah *string* dan dipisahkan berdasarkan spasi untuk mengubah arti singkatan yang berupa frase menjadi *term* atau *token*.

4.2.1.4. Filtrasi

Filtrasi merupakan proses untuk menghapus *token* yang termasuk dalam daftar *stop words*. Tujuan dari tahap ini adalah menghilangkan seluruh kata atau *token* yang terlalu umum dan menyisakan kata atau *token* yang lebih memiliki makna atau arti yang lebih penting. Gambar 4.4 adalah kode program untuk melakukan filtrasi.

```
public string[] filtration(string[] tokens)
{
  List<string> filtered_tokens = new List<string>();
  for (int i = 0; i < tokens.Count(); i++)
  {
    string token = tokens[i].Trim();
    token = Regex.Replace(token, @"[^0-9a-zA-Z]+", "");
    if (token.Length > 0)
    {
      if (Array.IndexOf(stopwords, token) < 0)
        filtered_tokens.Add(token);
    }
  }
  return filtered_tokens.ToArray();
}
```

Gambar 4.4. Fungsi Filtrasi

Pada kode fungsi filtrasi (Gambar 4.4), variabel `stopwords` merupakan *array* yang berisi *stop words* dalam huruf kecil (*lower case*). Karakter non-alfanumerik pada *token* harus dibuang. Apabila *token* tidak terdaftar pada *array* `stopwords`, maka *token* akan disimpan pada variabel penampung bernama `filtered_tokens` yang nantinya akan digunakan sebagai nilai kembalian.

4.2.1.5. Lematisasi atau *Stemming*

Lematisasi atau *stemming* merupakan proses untuk mengubah *token* atau *term* menjadi kata dasarnya. Pada proses ini imbuhan-imbuhan yang menempel pada *token* akan dihapus. Algoritma yang digunakan pada proses ini adalah algoritma Porter2 atau Snowball (Porter, 2001) yang merupakan pengembangan dari Porter Stemmer. *Library* Snowball stemmer yang digunakan oleh penelitian ini ditulis dalam Bahasa pemrograman C# yang dibuat oleh Iveonik System ltd (Сотник, 2011). Berikut adalah kode implementasi lematisasi dengan menggunakan Snowball stemmer (Gambar 4.5).

```
public string[] snowball_stemer(string[] tokens)
{
  var stemmer = new Iveonik.Stemmers.EnglishStemmer();
  List<string> stem_tokens = new List<string>();
  foreach (string s in tokens)
    stem_tokens.Add(stemmer.Stem(s));
  return stem_tokens.ToArray();
}
```

Gambar 4.5. Fungsi Lematisasi atau *Stemming*

Seluruh proses lematisasi telah diimplementasikan ke dalam *library* yang dibuat oleh Iveonik System ltd, sehingga pengguna dapat langsung menggunakannya. Cara untuk menggunakan kode Snowball Stemmer adalah dengan memanggil fungsi `Stem(token)` yang telah disediakan dalam kelas `EnglishStemmer`.

4.2.1.6. Pembuatan *Unigram* dan *Bigram*

Unigram dan *bigram* merupakan bagian dari n-gram, yaitu potongan kata sebanyak n berdasarkan pada urutan sekuensial teks *string*. *Unigram* merupakan n-gram dengan nilai n adalah satu (n-gram berukuran satu), sedangkan *bigram* merupakan n-gram dengan nilai n adalah dua (n-gram berukuran dua). Contoh: terdapat sebuah teks “Universitas Atma Jaya Yogyakarta”, maka *unigram* dari teks tersebut adalah “Universitas”, “Atma”, “Jaya”, “Yogyakarta”, sedangkan *bigram* dari teks tersebut adalah “Universitas Atma”, “Atma Jaya”, “Jaya Yogyakarta”. Tujuan dari penggunaan n-gram adalah untuk meningkatkan efektivitas model klasifikasi sentimen Binarized Multinomial Naïve Bayes. Berikut adalah kode program untuk membuat *unigram* dan *bigram* (Gambar 4.6).

```
public string[] generate_ngram(string[] tokens)
{
  List<string> copy = new List<string>(tokens);
  for (int i = 0; i < tokens.Count() - 1; i++)
    copy.Add(tokens[i] + "_" + tokens[i + 1]);
  return copy.ToArray();
}
```

Gambar 4.6. Fungsi Pembuatan Unigram dan Bigram

Pada Gambar 4.6, *token* yang telah diproses sudah dimuat dalam bentuk *unigram*, sehingga langkah selanjutnya adalah membentuk *bigram*. *Bigram* dibentuk dengan cara menyambungkan sebuah *token* dengan *token* selanjutnya dengan menambahkan karakter "_" (garis bawah) sebagai karakter pemisah.

4.2.1.7. Penerapan Multinomial Naïve Bayes

Multinomial Naïve Bayes merupakan metode untuk klasifikasi dengan cara menghitung probabilitas kelas data uji terhadap kelas data latih yang digunakan. Berikut ini adalah contoh penerapan algoritma Multinomial Naïve Bayes sesuai dengan contoh tabel (Tabel 2.1) yang diberikan pada situs NLP Stanford (Press, 2008).

Tabel 4.1. Contoh Data pada Situs NLP Stanford

	docID	Kata dalam dokumen	Kelas
Data latih	1	Chinese Beijing Chinese	China
	2	Chinese Chinese Shanghai	China
	3	Chinese Macao	China
	4	Tokyo Japan Chinese	Japan
Data uji	5	Chinese Chinese Chinese Tokyo Japan	?

** Tabel ini diambil dari http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html*

Tugas yang perlu dilakukan adalah menentukan kelas data uji dengan docID 5. Menurut Tabel 4.1, kelas yang tersedia adalah China dan Japan, sehingga

data docID 5 hanya akan diklasifikasikan ke dalam kelas China atau Japan. Langkah awal yang harus dilakukan adalah menentukan nilai *prior* atau nilai perbandingan dokumen berdasarkan kelasnya dalam data latih. Perhitungan nilai *prior* ditampilkan pada persamaan (4.1):

$$\hat{P}(c) = \frac{N_c}{N} \tag{4.1}$$

Nilai N_c merupakan jumlah data latih dalam kelas *c* dan N adalah jumlah seluruh data latih, sehingga ditemukan $\hat{P}(China) = \frac{3}{4}$ dan $\hat{P}(Japan) = \frac{1}{4}$. Langkah selanjutnya adalah menghitung nilai probabilitas bersyarat (*conditional probabilities*) untuk setiap *term* atau *token* pada data uji. Nilai probabilitas bersyarat dapat dihitung dengan menggunakan persamaan (4.2):

$$\hat{P}(t|c) = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B'} \tag{4.2}$$

Nilai T_{ct} adalah frekuensi kemunculan kata *t* dalam data kelas *c,* $\sum_{t' \in V} T_{ct'}$ adalah jumlah seluruh *term* pada data latih kelas *c,* dan *B* adalah jumlah *term* unik yang terdapat pada semua data latih. Berikut adalah contoh hasil perhitungan nilai probabilitas beryarat *term* Chinese dalam kelas China pada data uji dengan menggunakan algoritma Multinomial Naïve Bayes (4.3):

$$\hat{P}(Chinese|China) = \frac{5+1}{8+6} = \frac{6}{14} = \frac{3}{7} \tag{4.3}$$

Dengan menggunakan persamaan (4.2) didapatkan nilai probabilitas bersyarat semua *term* pada data uji terhadap kelas China dan Jepang, yaitu: $\hat{P}(Chinese|China) = 3/7$, $\hat{P}(Tokyo|China) = \hat{P}(Japan|China) = 1/14$, $\hat{P}(Chinese|Japan) = 2/9$, dan $\hat{P}(Tokyo|Japan) = \hat{P}(Japan|Japan) = 2/9$. Setelah didapatkan seluruh nilai probabilitas beryarat, maka langkah terakhir adalah

mengklasifikasikan data uji dengan mengalikan nilai *prior* dan semua nilai probabilitas bersyarat pada masing-masing kelas, hasil probabilitas klasifikasi data uji dijabarkan dalam persamaan (4.4) dan (4.5):

$$\hat{P}(China|d_5) \propto \frac{3}{4}.\left(\frac{3}{7}\right)^3.\frac{1}{14}.\frac{1}{14} \approx 0.0003 \quad (4.4)$$

$$\hat{P}(Japan|d_5) \propto \frac{1}{4}.\left(\frac{2}{9}\right)^3.\frac{2}{9}.\frac{2}{9} \approx 0.0001 \quad (4.5)$$

Nilai $\frac{3}{7}, \frac{1}{14}$, dan $\frac{1}{14}$ pada persamaan (4.4) merupakan nilai probabilitas bersyarat *term* Chinese, Tokyo, dan Japan terhadap kelas China. Nilai $\frac{2}{9}, \frac{2}{9}$, dan $\frac{2}{9}$ pada persamaan (5) merupakan nilai probabilitas bersyarat *term* Chinese, Tokyo, dan Japan terhadap kelas Japan. Nilai probabilitas bersyarat *term* Chinese dipangkatkan dengan tiga karena *term* Chinese muncul sebanyak tiga kali dalam data uji. Kesimpulan yang didapatkan dari persamaan (4.4) dan (4.5) adalah bahwa data uji docID diklasifikasikan dalam kelas China karena hasil probabilitas klasifikasi kelas China lebih besar dari pada kelas Japan.

4.2.1.8. Penerapan Binarized Multinomial Naïve Bayes

Binarized Multinomial Naïve Bayes merupakan pengembangan atau penambahan fitur pada algoritma Multinomial Naïve Bayes yang tujuannya adalah untuk mengubah format data latih dan data set yang digunakan ke dalam bentuk biner atau *boolean.* Maksud dari bentuk biner atau *boolean* adalah dalam data set dan data latih tidak diperbolehkan adanya *term* yang sama (duplikat). Pada data set dan data latih yang digunakan pada Binarized Multinomial Naïve Bayes, *term*

duplikat akan dihapus dan hanya disisakan *term* unik yang ditemukan pertama kali saja. Tabel 4.2 adalah contoh bentuk biner atau *boolean* dari Tabel 4.1.

Tabel 4.2. Bentuk Biner Contoh Data pada Situs NLP Stanford

	docID	Kata dalam dokumen	Kelas
Data latih	1	Chinese Beijing	China
	2	Chinese Shanghai	China
	3	Chinese Macao	China
	4	Tokyo Japan Chinese	Japan
Data uji	5	Chinese Tokyo Japan	?

Persamaan yang digunakan untuk menghitung nilai *prior* (4.1) dan probabilitas bersyarat (4.2) masih sama seperti Multinomial Naïve Bayes. Persamaan (4.6) dan (4.7) menunjukkan hasil perhitungan probabilitas klasifikasi dengan menggunakan Tabel 4.2 sebagai referensi data uji dan data latih.

$$\hat{P}(China|d_5) \propto \frac{3}{4}.\frac{1}{3}.\frac{1}{12}.\frac{1}{12} \approx 0.0017 \quad (4.6)$$

$$\hat{P}(Japan|d_5) \propto \frac{1}{4}.\frac{2}{9}.\frac{2}{9}.\frac{2}{9} \approx 0.0027 \quad (4.7)$$

Hasil perhitungan probabilitas klasifikasi Binarized Multinomial Naïve Bayes pada persamaan (4.6) dan (4.7) terlihat berbeda dengan hasil Multinomial Naïve Bayes pada persamaan (4.4) dan (4.5). Hasil klasifikasi Multinomial Naïve Bayes menunjukkan bahwa data uji docID 5 diklasifikasikan dalam kelas China, sedangkan Binarized Multinomial Naïve Bayes menunjukkan bahwa data uji docID 5 diklasifikasikan dalam kelas Japan. Penerapan algoritma Multinomial Naïve Bayes dan bentuk binernya dalam kode C# ditunjukkan pada Gambar 4.7.

```
public double multinomial_naive_bayes(string[] dataTrainPos,
string[] dataTrainNeg, List<string> dataTestTerms, bool
binarized = false)
{
  double countPosDocs = dataTrainPos.Count();
```

Gambar 4.7. Fungsi Multinomial Naïve Bayes

```
  double countNegDocs = dataTrainNeg.Count();
  double countDocs = countPosDocs + countNegDocs;
  double posPrior = countPosDocs / countDocs;
  double negPrior = countNegDocs / countDocs;

  List<string> dataTrainPosTerms = new List<string>();
  List<string> dataTrainNegTerms = new List<string>();

  if (binarized)
  {
    dataTestTerms = new List<string>(dataTestTerms.Distinct());
    foreach (string data in dataTrainPos)
      dataTrainPosTerms.AddRange(data.Split(' ').Distinct());
    foreach (string data in dataTrainNeg)
      dataTrainNegTerms.AddRange(data.Split(' ').Distinct());
  } else {
     foreach (string data in dataTrainPos)
       dataTrainPosTerms.AddRange(data.Split(' '));
     foreach (string data in dataTrainNeg)
       dataTrainNegTerms.AddRange(data.Split(' '));
  }
  double countPosTerms = dataTrainPosTerms.Count();
  double countNegTerms = dataTrainNegTerms.Count();
  double countVocabulary =
dataTrainPosTerms.Concat(dataTrainNegTerms).Distinct().Count();

  //Conditional Probabilities
  double posScore = posPrior;
  double negScore = negPrior;
  foreach (string term in dataTestTerms)
  {
    var termInPosDocs = from word in dataTrainPosTerms
                        where word == term select word;
    var termInNegDocs = from word in dataTrainNegTerms
                        where word == term select word;
    double posCount = termInPosDocs.Count();
    double negCount = termInNegDocs.Count();
    posScore *= (posCount + 1) / (countPosTerms +
countVocabulary);
    negScore *= (negCount + 1) / (countNegTerms +
countVocabulary);
  }
  return posScore - negScore;
}
```

Gambar 4.7. Lanjutan

Penerapan algoritma Binarized Multinomial Naïve Bayes akan dilakukan apabila parameter `binarized` diisi dengan nilai *true*. Nilai kembalian fungsi pada Gambar 4.7 merupakan pengurangan dari nilai probabilitas bersyarat kelas positif dengan nilai probabilitas bersyarat kelas negatif. Apabila nilai positif lebih besar

dari nilai negatif, maka data uji diklasifikasikan dalam kelas sentimen positif, begitu pula sebaliknya. Nilai kembalian fungsi tersebutlah yang menentukan kelas sentimen data uji.

4.2.2. SentiWordNet

Pra-proses yang diterapkan pada SentiWordNet dibagi menjadi tiga tahap, yaitu: mengartikan singkatan, tokenisasi, dan *part of speech* (POS) *tagging* (Kristina Toutanova, 2003). Fungsi dari POS *tagging* adalah untuk menentukan jenis kata (*part of speech*) dari setiap kata pada kalimat. Contoh *part of speech* yaitu: *noun* (kata benda), *verb* (kata kerja), *adjective* (kata sifat), *adverb* (kata keterangan), dan lain-lain. *Part of speech* yang diberikan juga telah disertai dengan keterangan waktunya, contoh: *past*, *present*, *future*, dan lain sebagainya. Kode yang digunakan untuk mengartikan singkatan sama dengan yang tertulis pada Gambar 4.3. Proses tokenisasi dan POS *tagging* diterapkan dengan menggunakan sebuah *library* dalam bahasa pemrograman C# yang dibuat oleh Sergey Tihon (Tihon, 2014). Gambar 4.8 adalah kode implementasi tokenisasi dan POS *tagging* penggunaan *library* POS *tagger* Sergey Tihon.

```
public List<object[]> postag(string text){
  var jarRoot = @"stanford-postagger-full-2015-12-09";
  var modelsDirectory = jarRoot + @"\models";
  var tagger = new MaxentTagger(modelsDirectory + @"\wsj-0-18-
bidirectional-nodistsim.tagger"); // Loading POS Tagger
  var sentences = MaxentTagger.tokenizeText(new
StringReader(text)).toArray();
  List<object[]> taggedSentences = new List<object[]>();
  foreach (ArrayList sentence in sentences)

taggedSentences.Add(tagger.tagSentence(sentence).toArray());
  return taggedSentences;
}
```

Gambar 4.8. Fungsi POS *Tagging*

Penerapan library POS *tagging* harus disertai dengan model dan *file* jar yang dapat diunduh dari situs Sergey Tihon. Misalkan masukan parameter `text` yang diberikan pada fungsi tersebut adalah "Barack Obama is pretty cool", keluaran yang dihasilkan adalah "Barack/NNP Obama/NNP is/VBZ pretty/RB cool/JJ". POS pada setiap kata dituliskan dalam berbagai macam kode, contoh: NNP adalah kode *proper noun* (kata benda, nama diri), VBZ adalah kode *verb, 3rd person singular present* (kata kerja, kata ganti orang ketiga), RB adalah kode *adverb* (kata keterangan), dan JJ adalah kode *adjective* (kata sifat). Pada dokumen leksikon SentiWordNet hanya terdapat empat jenis POS yaitu 'a' untuk *adverb*, 'n' untuk *noun*, 'r' untuk *adverb*, dan 'v' untuk *verb.* SentiWordNet tidak mengenal POS yang dihasilkan oleh *library* POS *tagger*, sehingga POS yang dihasilkan oleh POS *tagger* harus dikonversikan menjadi POS yang sesuai dengan SentiWordNet. Hasil konversi POS *tag* ke bentuk POS SentiWordNet dapat dilihat pada Tabel 4.3:

Tabel 4.3. Tabel Konversi POS *Tagger* ke POS SentiWordNet

POS *Tagger*	POS SentiWordNet
NN	n
NNS	
NNP	
NNPS	
VB	v
VBD	
VBG	
VBN	
VBP	
VBZ	
JJ	a
JJR	
JJS	
RB	r
RBR	
RBS	

Kode implementasi SentiWordNet dalam bahasa pemrograman Java telah disediakan pada situs SentiWordNet (http://sentiwordnet.isti.cnr.it/) yang dibuat oleh Peter Tönberg. Penelitian ini membuat kode implementasi SentiWordNet dalam bahasa pemrograman C# dengan mengacu pada kode yang telah dibuat oleh Peter Tönberg. Gambar 4.9 merupakan kode implementasi SentiWordNet dalam bahasa pemgrograman C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Thesis
{
  class SentiWordNet
  {
    private Dictionary<string, double> dictionary;
    public SentiWordNet()
    {
      dictionary = new Dictionary<string, double>();
      Dictionary<string, Dictionary<int, double>>
tempDictionary = new Dictionary<string, Dictionary<int,
double>>();
      string[] lines =
System.IO.File.ReadAllLines(@"SentiWordNet_3.0.0_20130122.txt")
;
      int lineNumber = 0;
      try
      {
        foreach (string line in lines)
        {
          if (!line.Trim().StartsWith("#"))
          {
            string[] data = line.Split('\t');
            string wordTypeMarker = data[0];
            if (data.Count() != 6)
            {
              throw new Exception("Incorrect tabulation format
in file, line: " + lineNumber);
            }
            double synsetScore = double.Parse(data[2]) -
double.Parse(data[3]);
            string[] synTermsSplit = data[4].Split(' ');
```

Gambar 4.9. Kelas Implementasi SentiWordNet

```
            foreach (string synTermSplit in synTermsSplit)
            {
              string[] synTermAndRank =
synTermSplit.Split('#');
              string synTerm = synTermAndRank[0] + "#" +
wordTypeMarker;
              int synTermRank = Int32.Parse(synTermAndRank[1]);
              if (!tempDictionary.ContainsKey(synTerm))
              {
                tempDictionary.Add(synTerm,
                new Dictionary<int, double>());
              }
              tempDictionary[synTerm].Add(synTermRank,
synsetScore);
            }
          }
        }

        foreach (KeyValuePair<string, Dictionary<int, double>>
entry in tempDictionary)
        {
          String word = entry.Key;
          Dictionary<int, Double> synSetScoreMap = entry.Value;
          double score = 0.0;
          double sum = 0.0;
          foreach (KeyValuePair<int, double> setScore in
synSetScoreMap)
          {
            score += setScore.Value / (double) setScore.Key;
            sum += 1.0 / (double)setScore.Key;
          }
          score /= sum;
          dictionary.Add(word, score);
        }
      } catch (Exception e)
      {
        Console.WriteLine(e);
      }
    }

    public double extract(string word, string pos)
    {
      word = word.ToLower();
      double score = dictionary.ContainsKey(word + "#" + pos) ?
dictionary[word + "#" + pos] : 0;
      Console.WriteLine(word + "#" + pos + " " + score);
      return score;
    }

    public double extractPos(string word, string pos)
    {
      if (pos == "NN" || pos == "NNS" || pos == "NNP" || pos ==
"NNPS")
        pos = "n";
```

Gambar 4.9. Lanjutan

```
      else if (pos == "VB" || pos == "VBD" || pos == "VBG" ||
pos == "VBN" || pos == "VBP" || pos == "VBZ")
        pos = "v";
      else if (pos == "JJ" || pos == "JJR" || pos == "JJS")
        pos = "a";
      else if (pos == "RB" || pos == "RBR" || pos == "RBS")
        pos = "r";
      return extract(word, pos);
    }
  }
}
```

Gambar 4.9. Lanjutan

Nilai sentimen pada setiap *synset* atau kata dapat diperoleh dengan menyesuaikan pada daftar *synset* dan POS pada leksikon SentiWordNet. Nilai nol dapat muncul dalam hasil perhitungan nilai sentimen dengan menggunakan SentiWordNet, berarti data diklasifikasikan ke dalam kelas netral.

4.2.3. AFINN-111

Pra-proses yang perlu dilakukan sebelum klasifikasi menggunakan leksikon AFINN-111 adalah tokenisasi dan mengartikan singkatan. Fungsi tokenisasi dan mengartikan singkatan yang digunakan pada pra-proses sama dengan yang tertulis pada Gambar 4.2 dan Gambar 4.3. Gambar 4.10 merupakan kode untuk mendapatkan nilai sentimen dengan AFINN-111:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Thesis {
  class AFINN {
    private Dictionary<string, double> dictionary;
    public AFINN() {
      int lineNumber = 0;
      dictionary = new Dictionary<string, double>();
      string[] lines =
System.IO.File.ReadAllLines(@"AFINN/AFINN-111.txt");
```

Gambar 4.10. Kelas AFINN-111

```
      foreach (string line in lines) {
        lineNumber++;
        string[] data = line.Split('\t');
        if (data.Count() != 2)
          throw new Exception("Incorrect tabulation format in
file, line: " + lineNumber);
        else
          dictionary.Add(data[0], double.Parse(data[1]));
      }
    }

public double extract(string word){
      return dictionary.ContainsKey(word) ? dictionary[word] :
0; ;
    }
  }
}
```

Gambar 4.10. Lanjutan

Seluruh nilai sentimen dari *token* atau kata dalam data uji akan dijumlahkan untuk mendapatkan nilai total sentimen dari data uji tersebut. Apabila nilai total sentimen yang diperoleh bernilai positif, maka kelas sentimen data uji adalah positif, dan apabila nilai total sentimen yang diperoleh bernilai negatif, maka kelas sentimen data uji adalah negatif. Sama halnya dengan SentiWordNet, pada AFINN-111 juga terdapat kemungkinan nilai total sentimen yang didapatkan adalah nol yang berarti bahwa kelas sentimen data uji adalah netral.

4.3. Menguji dan Membandingkan Hasil Analisis sentimen

Sebelum memasuki tahap prediksi, diperlukan uji coba untuk menguji dan membandingkan hasil analisis sentimen yang diberikan oleh model sentimen yang disebutkan pada subbab 4.2 tentang model analisis sentimen. Model yang digunakan untuk menguji model analisis sentimen adalah dengan menggunakan model evaluasi *confusion matrix* dengan perhitungan *precision, recall, accuracy*,

dan *F1-score*. Berikut adalah penjelasan masing-masing metode yang digunakan dalam pengujian.

Confusion Matrix merupakan matriks atau tabel yang menunjukkan perbandingan hasil kelas data yang sudah diklasifikasikan dengan hasil prediksi klasifikasi data tersebut. Dari *confusion matrix* dapat ditemukan nilai *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN). Data pada penelitian ini dibagi kedalam dua kelas, yaitu positif (+) dan negatif (-). Tabel 4.4 menunjukkan ilustrasi *confusion matrix* dengan menggunakan kelas positif dan negatif beserta posisi nilai TP, TN, FP, dan FN.

Tabel 4.4. Ilustrasi *Confusion Matrix*

		Hasil Prediksi	
		+	-
Kelas Aktual	+	TP	FN
	-	FP	TN

True positive (TP) adalah data kelas positif yang diprediksi sebagai data kelas positif, *true negative* (TN) adalah data kelas negatif yang diprediksi sebagai data kelas negatif, *false positive* (FP) adalah data kelas negatif yang diprediksi sebagai data kelas positif, dan *false negative* (FN) adalah data kelas positif yang diprediksi sebagai data kelas negatif. Setelah *confusion matrix* dibuat, maka nilai TP, TN, FP, dan FN dapat digunakan untuk menghitung *positive predictive value* (PPV) atau *precision*, *negative predictive value* (NPV), *sensitivity* atau *recall*, *specificity, accuracy*, dan *F1-Score*. Berikut adalah rumus untuk menghitung PPV

atau *precision* (4.8), NPV (4.9), *sensitivity* atau *recall* (4.10), *specificity* (4.11), *accuracy* (4.12), dan *F1-Score* (4.13):

$$PPV = precision = \frac{TP}{TP+FP} \quad (4.8)$$

$$NPV = \frac{TN}{TN+FN} \quad (4.9)$$

$$sensitivity = recall = \frac{TP}{TP+FN} \quad (4.10)$$

$$specificity = \frac{TN}{TN+FP} \quad (4.11)$$

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.12)$$

$$F1score = 2\frac{precision*recall}{precision+recall} \quad (4.13)$$

Selain indeks perbandingan yang telah disebutkan, penelitian ini juga membandingkan waktu yang dibutuhkan oleh masing-masing metode analisis sentimen untuk memproses data uji yang diberikan. Hal tersebut dilakukan untuk mengetahui dan menilai performa masing-masing metode analisis sentimen dalam mengolah data secara *real time*.

4.4. Analisis Sentimen dengan Menggunakan Data Asli

Setelah pengujian dan perbandingan selesai, didapatkan sebuah model analisis sentimen yang memiliki performa dan hasil terbaik. Langkah selanjutnya adalah menganalisis sentimen pada data asli dengan menggunakan model analisis sentimen tersebut. Pada tahap ini semua data asli akan diklasifikasikan kedalam dua jenis sentimen, yaitu positif atau negatif. Kelas sentimen tersebut nantinya akan

digunakan untuk mewakili dukungan suara atau *vote* yang diberikan oleh penulis status sesuai dengan lokasi dan kata kunci yang diterapkan saat pengumpulan data.

4.5. Agregasi

Setelah sistem menentukan kelas sentimen data asli, maka langkah selanjutnya adalah melakukan agregasi untuk menentukan *elector* yang menang di masing-masing negara bagian. Data asli harus dikelompokkan berdasarkan lokasi atau negara bagiannya. Dukungan suara atau *vote* akan diberikan kepada kandidat atau partai sesuai dengan kelas sentimen data asli. Apabila kelas sentimen adalah positif, maka suara akan diberikan kepada *elector* partai politik yang disebutkan, tetapi apabila kelas sentimen adalah negatif, maka suara akan diberikan kepada pihak lawan. *Elector* partai politik yang memiliki suara terbanyak akan memenangkan pemilihan suara pada negara bagian tersebut, dan akan memberikan suaranya pada *electoral college* sebagai *electoral votes.*

4.6. Implementasi *Electoral College*

Pada tahap agregasi, *elector* yang memenangkan negara bagian telah ditentukan. Pada tahap ini akan dilakukan perhitungan *electoral votes* yang dapat menentukan partai politik atau kandidat presiden yang diprediksikan menang dalam pemilihan presiden. *Elector* yang menang dalam negara bagian akan memberikan *electoral votes* untuk partai politik yang mengutusnya. Jumlah *electoral votes* yang diberikan sesuai dengan Lampiran 3. Partai politik atau kandidat presiden yang

mendapatkan lebih dari 270 *electoral votes* akan diprediksi sebagai pemenang pemilihan presiden Amerika Serikat 2016.