BAB II
LANDASAN TEORI

2.1 *Natural Language Processing*

Natural Language Processing (NLP) merupakan salah satu cabang ilmu AI yang berfokus pada pengolahan bahasa natural. Bahasa natural adalah bahasa yang secara umum digunakan oleh manusia dalam berkomunikasi satu sama lain. Bahasa yang diterima oleh komputer butuh untuk diproses dan dipahami terlebih dahulu supaya maksud dari *user* bisa dipahami dengan baik oleh komputer.

Ada berbagai terapan aplikasi dari NLP. Diantaranya adalah *Chatbot* (aplikasi yang membuat *user* bisa seolah-olah melakukan komunikasi dengan komputer), *Stemming* atau *Lemmatization* (pemotongan kata dalam bahasa tertentu menjadi bentuk dasar pengenalan fungsi setiap kata dalam kalimat), *Summarization* (ringkasan dari bacaan), *Translation Tools* (menterjemahkan bahasa) dan aplikasi-aplikasi lain yang memungkinkan komputer mampu memahami instruksi bahasa yang diinputkan oleh *user*. Ruang lingkup NLP antara lain adalah sebagai berikut:

a. *Question Answering Systems* (QAS). Kemampuan komputer untuk menjawab pertanyaan yang diberikan oleh *user*. Daripada memasukkan *keyword* ke dalam *browser* pencarian, dengan QAS, *user* bisa langsung bertanya dalam bahasa alami yang digunakannya, baik bahasa Inggris, Mandarin, ataupun Indonesia.
b. *Summarization*. Pembuatan ringkasan dari sekumpulan konten dokumen atau email. Dengan menggunakan aplikasi ini, *user* bisa dibantu untuk mengkonversikan dokumen teks yang besar ke dalam bentuk slide presentasi.
c. *Machine Translation*. Produk yang dihasilkan adalah aplikasi yang dapat memahami bahasa manusia dan menterjemahkannya ke dalam bahasa lain. Termasuk di dalamnya adalah *Google Translate* yang apabila dicermati semakin membaik dalam penterjemahan bahasa. Contoh lain lagi adalah BabelFish yang menterjemahkan bahasa pada *real time.*
d. *Speech Recognition*. *Field* ini merupakan cabang ilmu NLP yang cukup sulit. Proses pembangunan model untuk digunakan komputer dalam mengenali bahasa yang diucapkan sudah banyak dikerjakan. bahasa yang sering digunakan adalah berupa pertanyaan dan perintah.
e. *Document classification*. Sedangkan aplikasi ini adalah merupakan area penelitian NLP Yang paling sukses. Pekerjaan yang dilakukan aplikasi ini adalah menentukan dimana

tempat terbaik dokumen yang baru diinputkan ke dalam sistem. Hal ini sangat berguna pada aplikasi *spam filtering*, *news article classification*, dan *movie review.*

2.2 *Text Mining*

Text mining memiliki definisi menambang data berupa informasi sebuah teks dari hasil *crawling* data dari internet ataupun sebuah dokumen yang tidak terstruktur dan bertujuan untuk mengambil pengetahuan dari teks. Proses *mining* teks melibatkan penerapan struktur terhadap sumber data yang sebelumnya belum terstruktur kemudian melakukan ekstraksi informasi pengetahuan (Nurzahputra & Muslim, 2016).

Secara umum, *text mining* harus melewati beberapa tahapan proses untuk memperoleh hasil yang maksimal. Adapun proses proses yang biasa dilakukan dalam tahap *text mining* adalah sebagai berikut (Yasir & Zohar, 2002):

a. *Text Preprocessing*

   Proses pengubahan bentuk data yang belum terstruktur menjadi data yang terstruktur sesuai dengan kebutuhan, untuk proses *text mining* yang lebih lanjut (*sentiment analysis*, peringkasan, *clustering dokument*, etc.).

b. *Text Transformation*

   Proses transformasi teks melakukan perubahan kata menjadi kata dasar (stemming), pengurangan dimensi kata pada dokumen *(stop word removal*), *bag of word*, dan *vectorial dokument representation* untuk representasi dokumen.

c. *Feature Selection*

   Tahapan ini dilakukan pembuangan bagian-bagian yang tidak terkait, *feature* yang tidak berpengaruh akan dibuang dan memfokuskan feature yang berpengaruh dalam suatu kegiatan pemodelan atau penganalisaan data.

d. *Pattern Discovery*

   Pattern Discovery bertujuan untuk menemukan pola yang tepat sebagai pengetahuan dari teks dengan menggunakan teknik-teknik *data mining* seperti *classification* dan *clustering*.

e. *Interpretation*

   Tahapan ini melakukan interpretasi ke suatu bentuk untuk kemudian dilakukan evaluasi.

2.3 *Preprocessing*

Preprocessing merupakan tahapan awal dalam mengolah data input sebelum memasuki proses tahapan utama. *Preprocessing* terdiri dari beberapa tahapan. Adapun tahapan *preprocessing* berdasarkan (Triawati & Chandra, 2009), yaitu: *case folding*, *tokenizing* / *parsing*, *filtering*, *stemming*. Berikut penjelasan empat tahapan dalam proses *preprocessing* adalah sebagai berikut:

a. *Case Folding*

Case folding merupakan tahapan yang mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai dengan 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter* (pembatas) (Triawati & Chandra, 2009).

b. *Tokenizing*

Tahap *tokenizing* / parsing adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya (Triawati & Chandra, 2009). Selain itu, spasi digunakan untuk memisahkan antar kata tersebut.

c. *Filtering*

Tahap *filtering* adalah tahap mengambil kata - kata penting dari hasil tokenizing. Proses filtering dapat menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist* / *stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh *stopword* adalah "yang", "dan", "di", "dari" dan lain – lain (Triawati & Chandra, 2009).

d. *Stemming*

Stemming merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. *Stemming* kebanyakan digunakan pada teks berbahasa inggris dikarenakan teks berbahasa inggris memiliki struktur imbuhan yang tetap dan mudah untuk diolah sementara *stemming* untuk proses bahasa Indonesia memiliki struktur imbuhan yang rumit / kompleks sehingga agak lebih susah untuk diolah.

2.4 Twitter

Twitter adalah sebuah sosial media yang menawarkan layanan *microblogging* yang memungkinkan pengguna mengirimkan *tweet* atau kicauan dibatasi sebanyak 140 karakter dan membaca *tweet* pengguna lain. Twitter banyak dimanfaatkan oleh penggunanya sebagai media

mengutarakan pendapat, berjualan produk, mengkampanyekan seuatu, dsb (Hilal, 2016). Ada beberapa karakteristik yang dimiliki oleh Twitter antara lain:

a. *Length*

Panjang maksimum pesan Twitter adalah 140 karakter. Rata rata *tweet* yang sering ditemukan adalah sepanjang 14 sampai 78 karakter.

b. *Data availability*

Dengan Twitter *Aplication Programming Interface* (API) yang dimiliki, sangat mudah untuk mengumpulkan jutaan *tweet* untuk data set pelatihan.

c. *Language model*

Memposting pesan dari berbagai macam media yang berbeda dapat dilakukan oleh pengguna Twitter. Hal ini memungkinkan frekuensi salah eja dan bahasa gaul di *tweet* jauh lebih tinggi daripada media lain.

d. Domain

Pengguna Twitter dapat memposting pesan *tweet* tentang beragam topik, tidak seperti media lain yang dirancang untuk topik tertentu.

2.5 *Framework* Django

Django adalah sebuah *framework full-stack* untuk membuat aplikasi web dengan bahasa pemrograman Python. *Framework* akan membantu kita membuat web lebih cepat, dibandingkan menulis kode dari nol.

Django awalnya dikembangkan pada tahun 2003 dan 2005 oleh beberapa web developer yang bertugas membuat dan merawat web portal (*newspaper website*). Setelah membuat beberapa *website*, orang-orang tersebut mulai membuat ulang kode-kode yang pernah mereka tulis dengan menerapkan beberapa *design pattern*. Lalu disebarkan sebagai project *open source* dengan nama "Django" pada bulan juli 2005.

Django kemudian semakin berkembang, lalu dirilis versi 1.0 pada bulan september 2008. Sekarang (2018) Django sudah mencapai versi 2.0

2.6 *Levenshtein Distance*

Levenshtein Distance atau *Edit Distance* merupakan suatu pengukuran (metrik) untuk menghitung jumlah perbedaan antara dua kata Perhitungan jarak antara dua kata ditentukan dari jumlah minimum operasi perubahan untuk mengubah kata A menjadi kata B. *Levenshtein Distance* sendiri dikembangkan oleh Vladimir Levenshtein pada tahun 1965. *Levenshtein*

Distance merupakan metode dalam menghitung nilai yang didapat dari hasil operasi modifikasi satu kata dengan kata yang lain dengan bantuan *matrix*. Cara yang digunakan adalah dengan melihat satu persatu karakter dengan karakter lainya, apakah untuk menutupi perbedaan tersebut perlu adanya penambahan huruf, penghapusan huruf, atau penyisipan huruf. Dengan menggunakan fungsi *matrix* (m,n) dimana M mewakili kata yang dibandingkan, sedangkan N sebagai pembanding yang masing masing mewakili setiap huruf sehingga dapat lebih mudah melihat operasi apa yang perlu dilakukan untuk kata tersebut. Nilai yang akan didapat adalah seberapa banyak langkah yang diselesaikan untuk mendapatkan kemiripan kata. Total angka untuk setiap operasinya mengacu kepada *distance*, dimana semakin kecil *distance* semakin besar kemungkinan kata sesuai dengan target kata. Terdapat tiga macam operasi yang dapat dilakukan oleh algoritma ini yaitu:

a. Operasi pengubah karakter

   Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* "yamg" menjadi "yang". Dalam kasus ini karakter "m" diganti dengan huruf "n".

b. Operasi penambahan karakter

   Operasi penambahan karakter berarti enambahkan karakter ke dalam suatu *string*. Contohnya *string* "kepad" menjadi *string* "kepada", dilakukan penambahan karakter "a" di akhir *string*. Penambahan karakter tidak hanya dilakukan di akhir kata, namun bias ditambahkan diawal maupun disisipkan di tengah *string*.

c. Operasi penghapusan karakter

   Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string*. Contohnya string "barur" karakter terakhir dihilangkan sehingga menjadi string "baru". Pada operasi ini dilakukan penghapusan karakter "r".

Operasi operasi yang dilakukan tersebut dapat dilihat pada persamaan (3.1).

$$Dist_{a,b}(i,j) = Min \begin{cases} Dist_{a,b}((i,j-1)+1) = Min \\ Dist_{a,b}((i-1,j)+1) = Min \\ Dist_{a,b}((i-1,j-1)+1_{(a_i \neq b_j)}) = Min \end{cases} \quad (2.1)$$

Keterangan:

a	= string pertama
b	= string kedua
i	= iterasi string pertama
j	= iterasi string kedua
Dist	= jarak

2.7 *Spelling Correction*

Menurut Atmajaya (2009), *Spelling checker* adalah sebuah fasilitas yang berfungsi untuk mengecek kesalahan penulisan ejaan suatu kata berdasarkan bahasa tertentu. Umumnya berkerja dengan menandai kata yang berejaan salah dengan kurva berwarna merah. Fitur tambahan yang dimiliki oleh sebuah *spelling checker* adalah *words suggestion* yang berfungsi membantu *user* dengan memberikan daftar kata-kata yang memiliki ejaan yang mendekati *keyword*. Fitur lain yang biasa dimiliki oleh sebuah *spelling checker* adalah *auto correction*, yang berfungsi mengubah secara otomatis kata yang salah menjadi sebuah kata yang memiliki ejaan yang dekat. Semua tingkat kedekatan ejaan antara kata yang salah dengan kata yang muncul di daftar suggestion ataupun kata yang muncul karena fitur *auto correction* ditentukan dengan banyak metode salah satunya metode algoritma *Levenshtein Distance*. *Spelling checker* atau yang disebut pemeriksa ejaan adalah aplikasi yang memeriksa semua kata dalam sebuah dokumen untuk menghindari kesalahan pengejaan. Pemeriksa ejaan dapat berupa aplikasi mandiri atau sebagai bagian dari aplikasi yang lebih besar, seperti aplikasi pengolah kata, kamus elektronik, atau mesin pencari web. Cara kerja *spelling checker* memindai kata-kata pada suatu naskah dan mengekstraknya, membandingkan kata yang salah dengan memberikan pilihan kepada pengguna terhadap kata-kata yang diketahui ejaannya oleh pemeriksa ejaan tersebut. Ini mungkin hanya akan menampilkan beberapa daftar kata, atau juga mengandung beberapa informasi tambahan, seperti kata hubung serta atribut leksikal dan gramatikal.

Dalam beberapa kasus, pemeriksa ejaan memberikan saran kata yang salah, ini dikarenakan ketidak akuratan kata yang terdapat dalam program tersebut.

2.8 Penelitian Terkait

Sebelumnya penelitian terkait telah dilakukan oleh (Yulikawati & Winarko, 2017). Penelitian tersebut berhasil melakukan pencocokan kata pada pesan Twitter bahasa Indonesia

menggunakan algoritma *Soundex* dengan tingkat keakuratannya rata-rata sebesar 70,83% dengan waktu proses untuk masing-masing sama sekitar 12 detik per *tweet.*

Selanjutnya penelitian terkait yang dilakukan oleh (Irawan, 2016). Pada penelitiannya dengan menggunakan metode *noisy channel* sistem mampu membedakan beberapa variasi dari sebuah kata, namun tidak dapat membedakan kata yang harusnya tidak dinormalisasi dan tidak dapat memperbaiki singkatan. Akurasi yang didapatkan sistem adalah 70,12% dan *F1-Score* sebesar 38,36%.

Selain itu pada penelitian terkait lainya dilakukan oleh (Nasution, Bijaksana, & Al Faraby, 2017). Penelitian ini menggunakan metode yang berbeda yaitu menggunakan metode *word alignment* dan *Word2Vec* untuk menemukan kemiripan kata-kata pada al-quran. Berdasarkan evaluasi menggunakan regresi support *vector regression* (SVR) menghasilkan keakuratan lebih rendah jika dibandingkan menggunakan perhitungan TF-IDF.

Penelitian tentang normalisasi teks juga dilakukan oleh (Naradhipa et al., 2011). Dalam penelitiannya mendeteksi kata tidak baku dengan menggukan 2 metode, yaitu metode *dictionary lookup* dan koreksi dengan *forward reverse dictionary*. Lalu kemudian penelitiannya menambahkan modul umpan balik untuk memperbaharui daya leksikon dan menghasilkan sumber leksikon yang lengkap untuk digunakan dalam pemeriksa ejaan. Dengan menggunakan pendekatan ini hasil penelitian tersebut mendapatkan akurasi mencapai 96,1% dan rata-rata 93,7% untuk mendeteksi kata tidak baku, namun penelitian tersebut belum menerapkan normalisasi kata yang sudah diperiksa di dalamnya.

Penelitian (Han & Baldwin, 2007) telah melakukan studi normalisasi leksikal pada media *Short Message Service* (SMS), bahwa sebagian besar kata kata yang tidak tepat didasarkan pada variasi morfofonemik dan mengusulkan metode bertingkat untuk mendeteksi dan menormalkan kata kata yang tidak tepat. Pada penelitian ini juga konsep menormalisasi dengan membandingkan metode *benchmark* dari literatur, dan mencapai skor *F-score* yang tinggi dengan mengintegrasikan pencarian kamus, kesamaan kata dan pemodelan dukungan konteks. Gambaran perbandingan beberapa penelitian mengenai normalisasi teks diperlihatkan pada Tabel 2.1.

Tabel 2.1 Daftar perbandingan penelitian terkait

No.	Penulis	Fokus Penelitian	Metode
1	Amie Yulikawati	Normalisasi Leksikal Pada Pesan Twitter	metode *phonetic matching* dengan algoritma *soundex*
2	Johanes irawan	Normalisasi teks Twitter bahasa Indonesia	Metode *Noisy Channel Model*
3	Wardhana Z.Nasution	Analisis dan Implementasi Perhitungan Semantics Similarity Pada Ayat Al-Quran	Metode *Word Alignment* dengan pembuatan vektor menggunakan W*ord2Vec*
4	Aqsath Rasyid N. dkk	Spelling checker pada dokumen bahasa Indonesia	Metode *Dictionary Lookup* dan koreksi dengan *Forward Reverse Dictionary*
5	Bo Han and Timothy Baldwin	Normalisasi leksikal pada teks SMS	*Benchmark*