1. Algoritma Nazief & Adriani
   1. Information Retrieval

Information retrieval atau biasa disebut dengan ‘temu kembali informasi’ adalah ilmu yang memperlajari mengenai prosedur atau langkah-langkah dalam mendapatkan kembali informasi dari berbagai sumber yang relevan atau koleksi sumber yang diinginkan dan diperlukan.

Sistem temu kembali informasi dikatakan efektif apabila dapat menghasilkan informasi atau dokumen-dokumen yang relevan dengan yang diinginkan. Tetapi banyaknya varian morfologi dari tiap bisa berpengaruh terhadap proses temu kembali, oleh karena itu dibutuhkan suatu cara untuk mengembalikan setiap kata pada kata dasarnya, atau biasa disebut juga sebagai *Stemming.*

* 1. *Stemming*

*Stemming* sendiri adalah suatu proses yang dilakukan untuk mendapatkan kata dasar dari suatu kata pada kalimat dengan menghilangkan bagian-bagian atau kata imbuhan dari kata tersebut, baik itu imbuhan di awal (prefiks) maupun imbuhan di akhir (sufiks).

Setiap Bahasa memiliki struktur dan aturannya masing-masing, oleh karena itu tiap Bahasa memiliki cara atau algoritma tersendiri untuk melakukan *stemming*. Untuk Bahasa Indonesia sendiri telah dikembangkan dua algoritma *Stemming* yaitu algoritma Nazief & Adriani dan juga algoritma Porter.

* 1. Algoritma Nazief & Adriani

Algoritma Nazief & Adriani ini dikembangkan oleh Bobby Nazief dan Mirna Adriani. Algoritma ini menggunakan kamus kata dasar dan mendukung recodng, yakni penyusunan kembali pada kata-kata yang mengalami *overstemming* atau stemming berlebih. Berikut langkah-langkah algoritma Nazief & Adriani :

* + - * 1. Kata yang belum di-stemming dicari pada kamus, jika ditemukan, kata tersebut dianggap sebagai kata dasar yang benar dan algoritma d hentikan.
        2. Hilangkan Inflectional suffixes, yaitu dengan menghilangkan particle(“-lah”, ”-kah”, “-tah” atau“-pun”), kemudian hilangkan inflectional possessive pronoun suffixes (“-ku”, “-mu” atau ”-nya”). Cek kata di dalam kamus kata dasar, jika ditemukan, algortima dihentikan, jika tidak lanjut ke langkah 3.
        3. Hapus Derivational Suffix (“-i” atau ”-an”,”). Jika kata ditemukan dalam kamus kata dasar, maka algoritma berhenti. Jika tidak, maka lanjut ke langkah 3a:

a. Jika akhiran “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.

b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.

* + - * 1. Hapus Derivational Prefix (“be-”,”di-”,”ke-”,”me-”,”pe-“,”se-” dan “te-“). Jika kata yang didapat ditemukan didalam database kata dasar, maka proses dihentikan, jika tidak, maka lakukan recoding.

Tahapan ini dihentikan jika memenuhi beberapa kondisi berikut:

a. Terdapat kombinasi awalan dan akhiran yang tidak diijinkan

b. Awalan yang dideteksi sama dengan awalan yang dihilangkan sebelumnya.

c. Tiga awalan telah dihilangkan

* + - * 1. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus, maka algoritma ini mengembalikan kata yang asli sebelum dilakukan stemming.

Contoh :

Kata “Melanjutkan”

Langkah 1 : “Melanjutkan” tidak ada di kamus kata dasar

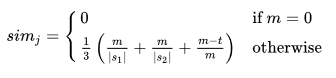
Langkah 2 : hasil proses menjadi “Melanjutkan”

Langkah 3 : hasil proses menjadi “Melanjut”

Langkah 4 : hasil proses menjadi “lanjut” => ditemukan sebagai kata dasar

1. Algoritma *Jaro winkler distance*

Algoritma *jaro winkler distance* adalah algoritma yang digunakan untuk melakukan pemeriksaan kesamaan dari dua kata. Penerapannya sendiri salahsatunya adalah untuk *autocorrection* dan *spelling suggestion.* Jaro winkler distance menggunakan jumlah karakter yang sama dari tiap string kemudian menghitungnya dengan rumus *jaro winkler distance. Jaro winkler distance* dicari dengan menghitung *jaro distance* nya terlebih dahulu*.* Berikut adalah rumus dari *jaro distance :*



|Si| adalah panjang dari string si

m adalah jumlah karakter yang sama

t adalah setengah dari jumlah transposisi

Kemudian *jaro winkler distance* dicari dengan rumus :



Dengan :

L = adalah Panjang dari prefix yang sama di awal maksimal 4 karakter

P = scaling factor

Hal yang pertama kali dilakukan adalah mencari jumlah karakter yang sama dari dua kata / string berbeda. Kemudian dihitung menggunakan rumus *jaro winkler distance* dan didapatkan nilai kemiripan yang bernilai dari 0 sampai dengan 1. Nilai 1 berarti sama, dan 0 berarti tidak ada kemiripan sama sekali. Sebelum di bandingkan, kata yang akan dibandingkan tersebut biasanya dilakukan proses *stemming* terlebih dahulu.

Contoh kata “tidur” dengan “tisur”:

Jumlah huruf yang sama (m) = 4,

Nilai transposisi = 0

Maka akan didapat :

((4/5) + (4/5) + (4)/4) / 3 = 0.86666666666

Simw = 0.86666666666 + 0 x 0.1 (1 - 0.86666666666) = 0.86666666666

Maka kedua kata tersebut tergolong mirip

1. Preprocessing tweet data
   * + - 1. Tokenization

def tokenization(tweet):

words\_array = tweet.text.split(' ')

return words\_array

* + - * 1. Case Folding

def case\_folding(tweet\_words):

temp = []

for word in tweet\_words:

temp.append(word.lower())

return temp

* + - * 1. Stemming

Stemming kata Bahasa Indonesia menggunakan library Sastrawi. Library Sastrawi menerapkan algoritma Nazief & Adriani

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()

stemmer = factory.create\_stemmer()

def stemming(tweet\_words):

temp = []

for word in tweet\_words:

temp.append(stemmer.stem(word))

return temp