Algoritma Stemming Sebagai Pra-Proses Pengecekan Kemiripan Naskah

Rahmadya Trias Handayanto [1,*], Ahmad Wafiq Amrillah [1], Intan Juwita [1], Muhammad Arifin, Setiaji [1], Reyvan Karani [1]

[1] Teknik Komputer; Universitas Islam 45; Jl. Cut Meutia No.83 Bekasi 17113 Telp. (021) 8802015, 8801027, 8808851-52 Fax. (021) 8801192; e-mail: rahmadya.trias@gmail.com, awafiq@gmail.com, intanjuwi4@gmail.com, arifin.rpl1@gmail.com, ajisetiaji464@gmail.com, revyankarani@gmail.com,

* Korespondensi: e-mail: rahmadya.trias@gmail.com

Diterima: 31 Oktober 2018; Review: 14 Nopember 2018; Disetujui: 28 Nopember 2018

Cara sitasi: Handayanto RT, Amrillah AW, Juwita I, Arifin M, Setiaji, Karani R. 2018. Algoritma Stemming Sebagai Pra-Proses Pengecekan Kemiripan Naskah. Bina Insani ICT Journal. 5 (2): 175 – 182.

Abstrak: Proses pencarian kata dasar dari suatu kata dalam Bahasa Indonesia lebih sulit dari pada bahasa Inggris. Proses yang dikenal dengan istilah stemming itu membutuhkan algoritma tertentu dalam mencari kata dasar suatu kata. Berbeda dengan bahasa Inggris yang hanya mengenal akhiran, dalam Bahasa Indonesia dikenal awalan, akhiran dan sisipan sehingga proses stemming jauh lebih rumit. Beberapa algoritma stemming untuk Bahasa Indonesia telah banyak diterapkan untuk proses temu kembali. Penelitian ini mencoba menerapkan algoritma stemming untuk mengecek kemiripan naskah berdasarkan kata dasarnya. Selain itu sebuah aplikasi sederhana dibuat untuk menguji akurasi proses stemming yang diusulkan.

Kata kunci: kata dasar, kemiripan naskah, Matlab, pemrosesan teks, temu kembali.

Abstract: *Searching a root of word in Indonesian is more difficult than English. This process, which is called stemming, need specific algorithms in finding the root. Whereas in English usually the roots are found by separating the suffixes, in Indonesia the roots are found by separating preffixes, suffixes, and inffixes as well, so it adds the complexity of stemming process. Many stemming algorithms have been proposed in information retrieval, but in this study, the stemming was also used for similarity check of papers. In addition, a prototype was created for checking the proposed-algorithm's accuracy.*

Keywords*: information retrieval, Matlab, root word, similarity, text processing.*

1. Pendahuluan

Untuk mempermudah proses perolehan informasi dan meningkatkan akurasinya (*recall* dan *precision*) diperlukan proses stemming, yaitu proses merubah suatu kata menjadi kata dasarnya. Untuk merubahnya diperlukan beberapa algoritma dan operasi-operasi berbasis string. Algoritma stemming telah banyak ditemukan, khususnya yang berbahasa Inggris atau negara eropa lainnya. Namun untuk bahasa Indonesia hanya beberapa yang telah diusulkan [Adriani et al., 2007; Keke et al., 2012; Susyanto, 2018]. Kebutuhan akan proses stemming khususnya untuk bahasa Indonesia sangat penting mengingat jumlah pengguna bahasa ini cukup banyak. Selain itu bahasa Indonesia memiliki tingkat kerumitan di atas bahasa Inggris atau eropa lainnya karena paling banyak menggunakan imbuhan (awalan (*preffix*), sisipan (*inffix*), dan akhiran), sementara bahasa Inggris kebanyakan pada akhiran (*suffix*), hanya beberapa yang ada di awalan seperti "mono-", "pre-", dan lain-lain. Itu pun tidak memiliki tingkat kesulitan dalam memecahnya menjadi kata dasar. Riset perolehan teks untuk Bahasa Indonesia masih terbilang baru dibandingkan dengan Bahasa Inggris yaitu ketika dimulainya proyek yang dilakukan Google dengan Google Translate-nya.

ISSN: 2355-3421 (Print) ISSN: 2527-9777 (Online); 175 - 182

Bahasa Indonesia yang berakar dari bahasa melayu merupakan perpaduan dari berbagai bahasa. Bahasa sansekerta merupakan bahasa yang paling banyak mempengaruhi Bahasa Indonesia. Bahasa Arab dan Eropa mulai diadopsi ke dalam Bahasa Indonesia ketika maraknya proses perdagangan di era sebelum kolonialisasi. Bahasa Indonesia memiliki banyak keunikan disamping adanya banyak imbuhan antara lain tidak mengenalnya waktu (*past, present*, atau *future*). Bahasa Indonesia juga cenderung terbuka dalam mengikuti perkembangan jaman sehingga cenderung berubah dan beradaptasi mengikuti istilah-istilah baru yang muncul, terutama di bidang teknologi informasi. Dengan demikian penelitian yang berkelanjutan sangan diperlukan dalam menggali informasi yang bertukaran di dunia maya saat ini yang menggunakan bahasa Indonesia.

Proses stemming saat ini banyak diterapkan dalam mesin-mesin pencari seperti Google dan Bing. Algoritma Porter masih menjadi andalan dalam mencari suatu kata sesuai dengan yang diharapkan [Manning, 2009]. Di Indonesia, riset stemming untuk Bahasa Indonesia telah banyak dilakukan dengan hasil yang beragam. Misalnya Nazief dan Adriani [Adriani et al., 2007; Keke et al., 2012] mengusulkan sebuah metode yang menggunakan kombinasi stemming dengan sebelumnya mencari ke basis data bahasa Indonesia (kamus). Perbaikan metode ini dilakukan dengan proses perhitungan jumlah karakter ketika akan dilakukan proses stemming disamping adanya perbaikan-perbaikan kecil guna meningkatkan akurasi [Susyanto, 2018].

Metode-metode yang diusulkan di atas bermaksud meningkatkan akurasi proses stemming, khususnya pada mesin temu kambali (*information retrieval*). Tetapi untuk penggunaan lainnya yaitu proses pengecekan plagiarisme, ada kalanya hanya butuh pola saja. Pola tersebut diambil dari kata dasar yang digunakan. Antara dua naskah yang diuji dicek kemiripan polanya berdasarkan kata dasar yang telah ditemukan. Sedikit ketidakakuratan pada proses stemming ini tidak begitu banyak berpengaruh dalam proses pengecekan kemiripan mengingat fungsi utama memang bukan pada penentuan keakuratan kata dasar yang diperoleh melainkan kemiripan pola satu naskah dengan naskah yang diduga plagiasi.

Penerapan linguistic dalam pemrosesan teks masih banyak yang meragukan keabsahannya [Keke et al., 2012]. Selain itu banyaknya sumber daya komputasi yang diperlukan membebani proses pengecekan kemiripan naskah itu sendiri yang sejatinya adalah tugas utama sistem plagiarism checker. Untuk itu penelitian ini hanya menerapkan algoritma ringan yang memisahkan imbuhan dengan kata dasarnya sebagai proses awal sebelum masuk ke mesin pengecekan plagiasi dalam bahasa Indonesia. Sebuah *Graphic User Interface* (GUI) dibuat dengan bahasa Matlab untuk menguji akurasinya. Sebelum sampai pada hasil dan pembahasan, berikutnya akan dibahas metode stemming dan proses pembuatan aplikasinya. Tulisan diakhiri dengan kesimpulan dan saran

2. Metode Penelitian

Algoritma merupakan prosedur yang berisi langkah-langkah dalam memproses sebuah input menjadi output. Istilah ini muncul ketika Muhammad bin Musa Al-khawarizmi, seorang ahli matematika Persia, berkembang di Eropa yang karena lidah orang eropa yang sulit mengucapkan kata dalam bahasa arab, kata Al-khawarizmi menjadi Algorithm [Cormen et al., 2009]. Algoritma selanjutkan diterapkan dalam beragam komputasi dan kian penting seiring ditemukan dan penggunaan luas komputer. Bahkan saat ini algoritma tidak hanya diterapkan dalam bidang teknologi informasi, tetapi juga pada bidang-bidang lainnya dengan teknik berfikirnya yang dikenal dengan istilah *computational thinking*.

2.1. Algoritma Nazief dan Adriani

Salah satu algoritma Stemming untuk bahasa Indonesia yang terkenal adalah yang diusulkan oleh Nazief dan Adriani. Algoritma yang dikembangkan di universitas Indonesia ini memiliki langkah-langkah sebagai berikut: **Pertama**, sebelum proses stemming dicari terlebih dahulu padanannya di kamus, jika sudah ada maka proses selesai. **Kedua**, menghilangkan *inflectional suffixes* seperti kata “-lah”, “-kah”, “-pun”, atau “-tah” dan cek kembali ke kamus apakah sudah ada, jika ada di kamus proses selesai. **Ketiga**, menghapus turunan suffix (“-an” dan “-i”) dan kembali cek ke kamus jika sudah ada proses selesai tetapi jika ada cek lagi. **Keempat,** jika akhiran “-an” dihapus terdapat “k” maka “k” dihapus juga. Cek ke kamus dan berhenti jika ada. **Kelima**, kembalikan akhiran yang dihapus (“-l”, “-an” atau “-kan”) untuk masuk ke proses berikutnya. **Keenam**, menghapus turunan preffix (“be-“, “di-“, “ke-“, “me-“, “pe-”, “se-“, dan “te-“) dan cek di kamus jika ada berhenti, tetapi jika tidak ada lakukan proses recoding.

Beberapa peneliti memperbaiki metode stemming usulan Nazief dan Adriani untuk memperbaiki akurasi proses penentuan kata dasar. Karena karakternya yang mengecek ketersediaan kata dasar pada kamus, algoritma tersebut memiliki kelemahan dari sisi kompleksitas komputasi (*computational complexity*) yang tinggi dan membutuhkan sumber daya komputer yang tinggi (prosesor dan memori).

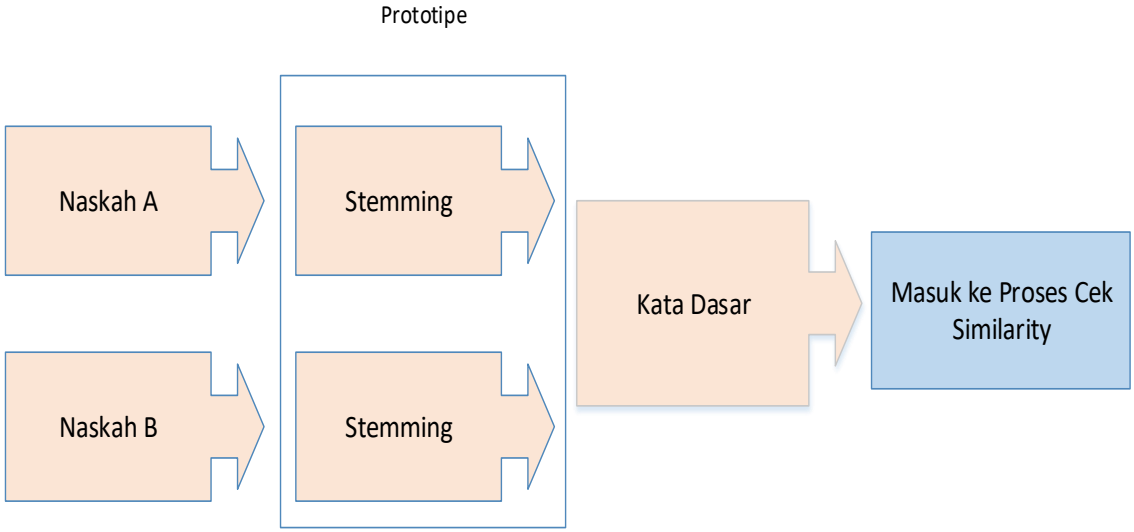
2.2. Algoritma Porter

Salah satu algoritma *stemming* yang lebih dahulu dikembangkan untuk bahasa Inggris diperkenalkan oleh Porter [Kraaij, 1994; Manning, 2009]. Algoritma ini tidak mengandalkan kamus sebagi basis data, Algoritma dengan beberapa tahap untuk penggunaan dalam bahasa Indonesia ini memiliki rincian langkah sebagai berikut: 1). Menghapus partikel “-lah”, “-tah”, “-kah” dan “-pun”. 2). Menghapus kata ganti seperti “-ku”, “-mu”, dan “-nya”. 3). Mengecek apakah ada awalan pertama, jika tidak ada lanjut ke langkah 4 dan jika ada lanjut ke langkah 5. 4). Menghapus awalan kedua, lanjut ke langkah 6. 5). Menghapus akhiran. Jika tidak ada maka kata tersebut adalah kata dasar, jika ada lanjut ke 7. 6). Menghapus akhiran dan kata akhir diasumsikan sebagai kata dasar. 7). Menghapus awalan kedua dan kata akhir diasumsikan sebagai kata dasar (*basic/root word*).

Di dalam bahasa Indonesia dikenal awalan yang lebih dari satu tetapi tidak semua diperkenankan. Urutan awalan yang diperkenankan antara lain “meng-“ dengan “per-“, misalnya dalam kata “mempertahankan” serta antara awalan “di-“ dengan “ber-“ misalnya “diberdayakan”. Saat ini perkembangan bahasa Indonesia mengganti sisipen -per- dengan menghilangkan/meluluhkan unsur “p” dari misalnya mempertahankan menjadi memertahankan, memperhatikan menjadi memerhatikan.

2.3. Rancangan Prototipe

Algoritma *stemming* yang akan diterapkan dalam aplikasi yang diusulkan adalah algoritma Porter dan Nazief & Adriani tanpa kamus. Alasan meniadakan kamus adalah guna mempercepat proses *stemming* karena hanya akan difungsikan sebagai proses awal pengecekan kemiripan naskah. Gambar 1 memperlihatkan metode prototipe yang diusulkan. Walaupun dalam bentuk purwarupa, perancangan tetap menerapkan langkah standar rekayasa perangkat lunak dalam membuat sebuah aplikasi yaitu perencanaan (*planning*), pembuatan kode sumber (*coding*) hingga penerapan (*implementation).*



Sumber: Hasil Penelitian (2018)

Gambar 1. Metodologi Penelitian

Dalam penelitian ini, ruang lingkup dibatasi hanya pada prototipe untuk melakukan proses *stemming* dengan hasil berupa untaian kata dasar tanpa adanya awalan, sisipan, dan akhiran. Dengan menggunakan kata dasar hasil *stemming* diharapkan sistem pengecekan

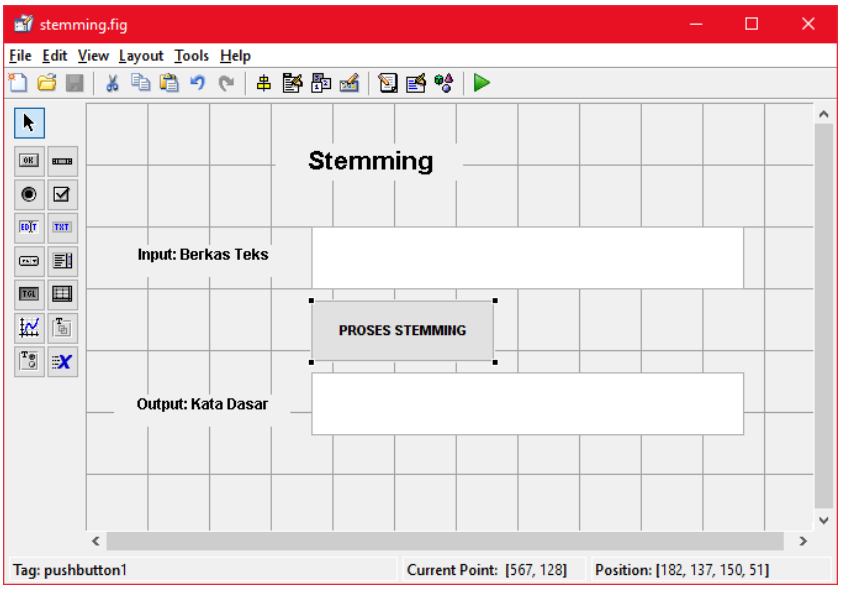
ISSN: 2355-3421 (Print) ISSN: 2527-9777 (Online); 175 - 182

plagiarisme mampu membedakan kalimat yang hanya dikonversi dari kalimat pasif menjadi kalimat aktif karena “meng-“ dan “di-“ dihilangkan oleh proses *stemming*.

Proses yang terjadi pada purwarupa yang dirancang adalah sebagai berikut. Pertama-tama dua naskah dimasukan dalam inputan pertama lewat GUI yang dibangun dengan software Matlab. Sebuah tombol dengan fungsi pencarian file disertakan dalam purwarupa untuk memudahkan pengguna memasukan file yang akan dihilangkan kata imbuhannya. Beberapa fungsi Matlab diperlukan seperti **regexprep** dan **strcat** untuk berturut-turut mencari dan mengganti serta menyisipkan spasi kosong, khususnya di awal kalimat. Selanjutnya modul yang berupa fungsi *stemming* dimasukan dalam tombo fungsi proses dalam bentuk tombol. Hasilnya berupa dua file yaitu file yang akan diuji plagiarisasinya dan file sumber yang juga telah dalam bentuk kata dasar yang akan menjadi rujukan kemiripan suatu teks yang akan diuji.

2.4. Pembuatan *Graphic User Interface* (GUI)

Untuk menguji algoritma *stemming* yang diusulkan, diperlukan GUI untuk mempermudah testing. GUI yang diimplementasikan dalam studi ini menggunakan bahasa pemrograman Matlab versi 2008 ke atas. Gambar 2 memperlihatkan rancangan purwarupa untuk pengujian algoritma stemming sebelum diterapkan, dengan GUI berupa satu *edit text* untuk memasukan dan satu lagi untuk mengeluarkan (*output*) serta satu tombol ekseksi proses *stemming*.



Sumber: Hasil Penelitian (2018)

Gambar 2. Rancangan GUI

Walaupun Matlab di versi awal bekerja dalam mode teks/konsol, dengan GUI kemampuan matematis Matlab tetap terjaga berkat efisiensi dalam penggunaan matriks. Gambar 2 adalah GUI yang akan digunakan untuk membuat algoritma *stemming*. Masukan berkas yang akan dicari kata dasarnya diletakkan di bagian “Input: Berkas Teks”, bisa dengan mengetik atau meng-*copy* dari sumber teks. Algoritma *stemming* diletakan di bagian tombol “PROSES STEMMING” lewat m-file editor (klik kanan dilanjutkan dengan view Callback). Hasil proses yang berupa kata dasar dikirim ke edit teks di sebelah kanan “Output: Kata Dasar”.

2.5. Pembuatan Algoritma *Stemming*

Algoritma *Stemming* membutuhkan fungsi-fungsi pemrosesan teks pada Matlab [Banchs, 2013]. Fungsi-fungsi yang diperlukan adalah “set” dan “get” yang fungsinya berturut-turut untuk mengirimkan string dan menerima string dari edit teks. Fungsi “strcat” diperlukan untuk menyisipkan spasi di awal kalimat karena jika tidak maka kata pertama walau ada awalan dianggap tidak ada. Fungsi yang paling penting adalah fungsi “regexprep” yang berfungsi mencari kata yang sesuai dengan kata kunci (dalam hal ini awalan, sisipan dan akhiran)

dilanjutkan dengan me-*replace* (dalam kasus ini karena menghilangkan maka diganti dengan tanpa karakter dengan dua *single-dash* ‘ ‘).

Dalam mengelola pencarian karakter/string, Matlab menyediakan aturan seperti ditunjukan dalam tabel 1. Fungsi yang memanfaatkan meta-karakter ini adalah dikenal dengan istilah regular expression.

Tabel 1. Meta-karakter untuk pencarian karakter/string

Meta-Karakter	Arti
.	Any character
[]	Any character contained within the brackets
[^]	Any character not contained within the brackets
\w	A word character [a-z_A-Z0-9]
\W	Not a word character [^a-z_A-Z0-9]
\d	A digit [0-9]
\D	Not a digit [^0-9]
\s	Whitespace [\t\r\n\f\v]
\S	Not whitespace [^ \t\r\n\f\v]
()	Group subexpression
|	Match subexpression before or after the |
^	Match expression at the start of string
$	Match expression at the end of string
\<	Match expression at the start of a word
\>	Match expression at the end of a word
*	Match zero or more occurrences
+	Match one or more occurrences
?	Match zero or one occurrence
{n,m}	Match between n and m occurrences

Sumber: Hasil Penelitian (2018)

Algoritma yang diterapkan dalam *stemming* yang dirancang tidak melibatkan basis data (kamus) untuk meningkatkan kecepatan proses. Untuk menghasilkan *stemming* yang sempurna agak sulit karena terkadang secara linguistik ada kata-kata tertentu yang memerlukan pengecualian, ketidakteraturan, dan sebagainya. Tetapi untungnya sebagian besar kata mengikuti aturan-aturan standar dalam awalan, sisipan, dan akhiran. *Pseudocode* untuk proses *stemming* adalah sebagai berikut:

Input: String
Output: Kata Dasar
Konversi seluruh karakter menjadi huruf kecil
Sisipkan satu spasi di awal kata
Cari awalan dengan format <spasi><awalan> dan hilangkan jika jumlah huruf tersisa > 3
Lakukan kembali untuk mencari awalan ke-2
Cari sisipan dengan format <sisipan> dan hilangkan jika jumlah huruf tersisa > 6
Cari akhiran dengan format <akhiran><spasi/titik> dan hilangkan jika jumlah huruf tersisa > 3

Beberapa kata butuh pengecualian karena kurang dari tiga kata walau memiliki imbuhan, misalnya kata “berdoa” yang bisa dihilangkan menjadi “doa”. Khusus awalan, karena kerap dijumpai awalan yang lebih dari satu misalnya “memperbaharui” maka perlu dua kali proses penghilangan awalan (“mem” dan “per”) sebelum menghilangkan sisipan “ha” dan akhiran “i”, sehingga ditemukan kata dasar “baru”.

ISSN: 2355-3421 (Print) ISSN: 2527-9777 (Online); 175 - 182

Beberapa aspek linguistik bisa saja dimasukan seperti setelah menghilangkan “peng” dalam kata “pengecualian” ditambahkan “k” agar hasil kata dasarnya “kecuali”, tetapi karena fungsi *stemming* untuk pengecekan kemiripan, efek dari hilangnya “k” masih dapat ditolerir mengingat yang dicari hanya pola kalimat saja.

3. Hasil dan Pembahasan

Setelah mengikuti prosedur standar perancangan perangkat lunak yang dimulai dari analisa, disain, coding, hingga testing, sebuah prototipe *stemming* dihasilkan (Gambar 3). Secara sederhana testing dicoba dengan menguji sebuah kata dengan imbuhan (awalan, sisipan, dan akhiran) apakah dihasilkan outpun kata dasarnya atau tidak. Jika sistem mampu mengkonversi kata tersebut menjadi kata dasarnya maka algoritma sudah cukup untuk melakukan proses *stemming.*

Sumber: Hasil Penelitian (2018)

Gambar 3. Testing Sistem *Stemming* Awal

Untuk menghasilkan algoritma *stemming* yang lengkap dibutuhkan banyak algoritma tambahan. Untuk jumlah kata yang banyak, misalnya ribuan kata, maka proses pencarian kata dasar menjadi berat jika menggunakan algoritma dengan tambahan kondisi yang banyak, apalagi jika menggunakan pencarian ke basis data berupa kamus. Untuk proses awal (*pre-processing*) sistem pengecekan kemiripan sebagai kebutuhan utama bukanlah kata dasar melainkan pola kalimat, oleh karena itu dengan penghilangan imbuhan sepertinya sudah cukup memadai karena hanya membutuhkan proses pencarian dengan fungsi *regexprep* pada bahasa pemrograman Matlab.

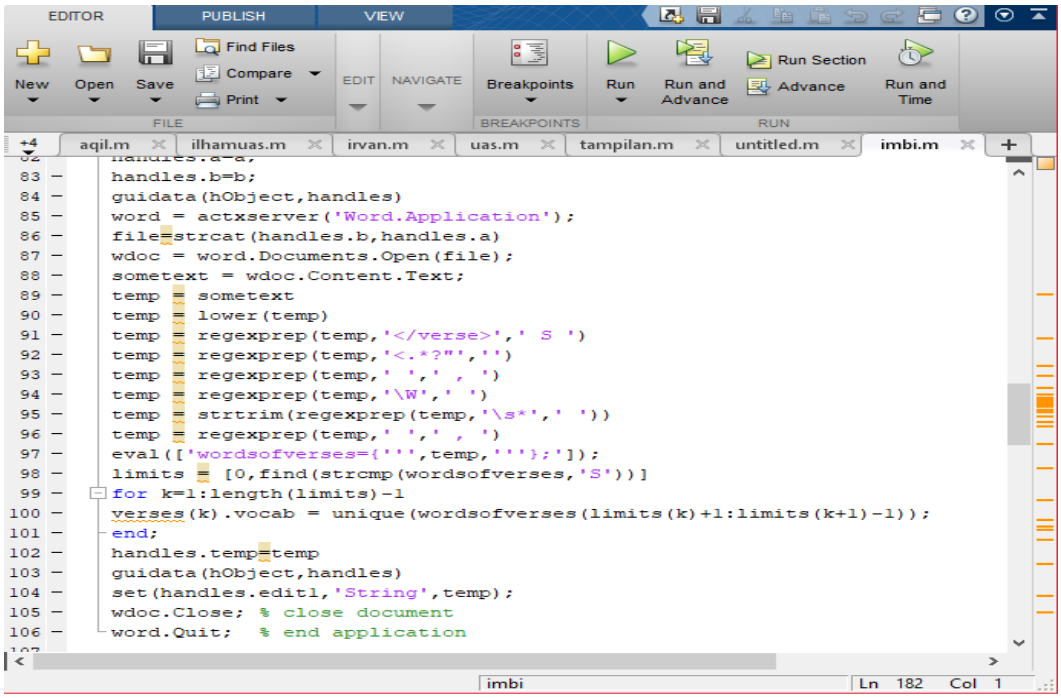
Sebagian besar teks ditulis dalam format *Word Processor*, misalnya Microsoft Word. Oleh karena itu diperlukan User Interface (UI) yang berfungsi memanggil naskah dalam format tersebut. Gambar 4 memperlihatkan tombol mengambil file teks, memproses teks tersebut dengan *stemming* serta dilanjutkan dengan menyimpan file teks tersebut untuk diproses lebih lanjut pada mesin uji kemiripan (*plagiarism check*). Dengan file teks yang sudah tidak memiliki imbuhan (kata dasar) diharapkan akurasi uji kemiripan lebih baik dibanding dengan tanpa proses *stemming*.

Untuk memudahkan implementasi, file executable dibuat dengan proses *Deployment* yang tersedia di Matlab. Dengan demikian penggunaannya lebih praktis dan mudah karena tidak memerlukan Software Matlab yang dilihat dari sisi biaya lisensi cukup tinggi. *Stemming* merupakan proses yang unik karena selalu memerlukan perbaikan (*update*) mengingat perkembangan Bahasa Indonesia (kamus besar bahasa Indonesia) yang cepat disamping tentu saja sulit menghasilkan algoritma *stemming* yang sempurna.

Sumber: Hasil Penelitian (2018)

Gambar 4. Purwarupa Sistem *Stemming*

Selanjutnya file hasil proses *stemming* dipindahkan baik naskah sumber dan naskah yang diuji ke sistem pengecekan plagiasi yang banyak beredar di pasaran saat ini. Beberapa mesin uji kemiripan menggunakan metode N-gram dimana dua (bi-gram) atau lebih deretan kata yang sama dicari ke naskah sumber. Misalnya dengan N sebanyak 3 maka kombinasi dari tiga kata jika terdapat pada naskah sumber maka dikatakan ada kemiripan/sama sebaliknya walaupun terdapat kata yang sama pada naskah sumber tetapi tidak ada dua kata lainnya maka masih dianggap original. Begitu pula untuk usaha merubah kalimat aktif yang berawalan “me-“ menjadi kalimat pasif yang berawalan “di-“ dapat dicegah karena proses *stemming* yang hanya mengambil kata dasarnya saja. Oleh karena itu kemampuan *stemming* dalam menentukan kata dasar sangat penting untuk mendukung algoritma N-gram tersebut. Gambar 5 memperlihatkan bagaimana fungsi *stemming* diimplementasikan dalam bahasa pemrograman (Matlab 2013).



Sumber: Hasil Penelitian (2018)

Gambar 5. Pembuatan Fungsi *Stemming*

ISSN: 2355-3421 (Print) ISSN: 2527-9777 (Online); 175 - 182

4. Kesimpulan

Dengan jumlah pemakan sebanyak lebih dari dua ratus juta jiwa Bahasa Indonesia menjadi sumber informasi di internet. Sumber informasi yang saat ini sudah membengkak dan masuk dalam kategori *big data* memerlukan penanganan yang efektif dan efisien, misalnya dalam hal temu kembali atau pengecekan kemiripan. Jika dalam perolehan informasi memerlukan akurasi tinggi dalam penentuan informasi kata dasar suatu kalimat agar mudah dalam proses indeksasi, dalam pengecekan kemiripan tidak terlalu membutuhkan akurasi tinggi melainkan sekedar menentukan kata dasar yang mungkin dengan proses komputasi yang cepat dan efisien. Hasil testing prototipe menunjukan kemampuan dalam proses *stemming* sehingga bermanfaat dalam proses selanjutnya, yaitu pengecekan plagiasi karena N-gram membutuhkan kata dasar dalam membandingkan kombinasi beberapa kata antara satu naskah yang akan dicek kemiripannya dengan naskah-naskah sumber. Untuk itu penelitian yang akan datang akan fokus kepada akurasi dari proses *stemming* terhadap hasil uji kemiripan. Selain itu perlu dipertimbangkan juga penerepannya dengan aplikasi berbasis web agar lebih mudah digunakan oleh pihak-pihak yang memerlukan proses *stemming.* Proses *stemming* diharapkan dapat membantu proses pengecekan kemiripan dan layak dijadikan proses awal (*preprocessing*) yang penting untuk kelanjutan penelitian ini.

Referensi

Adriani M, Asian J, Nazief B, Williams HE. 2007. Stemming Indonesian : A Confi x-Stripping Approach. 6: 1–33.

Banchs RE. 2013. Text Mining with MATLAB. Barcelona: Springer.

Cormen TH, Leiserson C, Rivest RL, Stein C. 2009. Introduction to Algorithms. Massachusetts: The MIT Press.

Keke D, Chikita R, Prayogo AD. 2012. Sistem Temu Balik Informasi. Yogyakarta.

Kraaij W. 1994. Porter ' s stemming algorithm for Dutch. 167–180.

Manning C. 2009. An Introduction to Information Retrieval. Cambridge: Cambridge University Press.

Susyanto T. 2018. Implementasi dan Analisis Algoritma Stemming Nazief dan Adriani. SINUS.