EKSTRAKSI KATA KUNCI PADA ARTIKEL MENGGUNAKAN METODE *TEXTRANK*

SKRIPSI

Oleh :
MUHAMMAD AUFA SHIDDIQ
NIM. 15650085

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2019

EKSTRAKSI KATA KUNCI PADA ARTIKEL MENGGUNAKAN METODE *TEXTRANK*

SKRIPSI

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
MUHAMMAD AUFA SHIDDIQ
NIM. 15650085

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2019

LEMBAR PERSETUJUAN

EKSTRAKSI KATA KUNCI PADA ARTIKEL MENGGUNAKAN METODE *TEXTRANK*

SKRIPSI

Oleh :
MUHAMMAD AUFA SHIDDIQ
NIM. 15650085

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : 14 Juni 2019

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Cahyo Crysdian	M. Imamudin, Lc., MA
NIP. 19740424 200901 1 008	NIP. 19740602 200901 1 010

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

LEMBAR PENGESAHAN

EKSTRAKSI KATA KUNCI PADA ARTIKEL MENGGUNAKAN METODE *TEXTRANK*

SKRIPSI

Oleh :
MUHAMMAD AUFA SHIDDIQ
NIM. 15650085

Telah Dipertahankan di Depan Dewan Penguji
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Pada Tanggal 14 Juni 2019

Susunan Dewan Penguji			**Tanda tangan**
1. Penguji Utama	:	Roro Inda Melani, M.T., M.Sc NIP. 19780925 200501 2 008	( )
2. Ketua Penguji	:	Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
3. Sekretaris Penguji	:	Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )
4. Anggota Penguji	:	M. Imamudin, Lc., MA NIP. 19740602 200901 1 010	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama : Muhammad Aufa Shiddiq
NIM : 15650085
Jurusan : Teknik Informatika
Fakultas : Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan Skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 14 Juni 2019
Yang membuat pernyataan,

Muhammad Aufa Shiddiq
NIM. 15650085

MOTTO

Rahasia keberhasilan adalah kerja keras dan belajar dari kegagalan

HALAMAN PERSEMBAHAN

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ

Dengan mengucap syukur Alhamdulillah, tak terasa hampir 4 tahun telah dilalui, dan kini saatnya mengakhiri perjuangan di Kampus Ulul Albab tercinta. Bukan berhenti menuntut ilmu, tapi untuk mengamalkan ilmu agar bermanfaat bagi sesama. Kupersembahkan karyaku ini untuk semua pihak yang telah terlibat dalam penulisan skripsi ini, baik secara langsung maupun tidak langsung.

Terima kasih kepada kedua orang tua saya Bapak Nuril Huda dan Ibu Zanifatul Abidah yang tidak pernah lelah mendukung dan mendoakan. Selalu mendidik dan mengajarkan segala macam nilai-nilai agama dalam kehidupan. Teruntuk pula kakakku Azizah, S.Si yang selalu memotivasi dalam suka dan duka.

Tak lupa terima kasih kepada sahabat-sahabat seperjuangan keluarga Teknik Informatika Interface 2015 serta keluarga besar Teknik Informatika UIN Maulana Malik Ibrahim Malang yang selalu memberikan support, motivasi disaat down, dan hiburan saat jenuh.

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan studi dengan tepat waktu sekaligus menyelesaikan skripsi dengan baik dan lancar. Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana komputer pada Fakultas Sains dan Teknologi (FSAINTEK) Program Studi Teknik Informatika di Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Didalam pengerjaan skripsi ini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada:

1. Prof. Dr. Abdul Haris, M.Ag, selaku Rektor Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini,M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika sekaligus Dosen Pembimbing I yang telah membimbing dalam penyusunan skripsi ini hingga selesai.
4. M. Imamudin, Lc., MA, selaku Dosen Pembimbing II yang telah membimbing dalam penyusunan skripsi ini hingga selesai.

5. Irwan Budi Santoso, M.Kom dan Roro Inda Melani, M.T., M.Sc, Selaku Dosen penguji yang telah memberikan banyak saran untuk kebaikan penulis.
6. Ayah, Ibu, dan Kakak tercinta yang telah banyak memberikan doa dan dukungan kepada penulis secara moril maupun materil hingga skripsi ini dapat terselesaikan.
7. Teman-teman Teknik Informatika Interface 2015 yang senantiasa memberi motivasi dan berjuang bersama selama menjadi mahasiswa.
8. Semua pihak yang telah banyak membantu dalam penyusunan skripsi ini yang tidak bisa penulis sebutkan semuanya.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat kekurangan dan penulis berharap semoga skripsi ini bisa memberikan manfaat kepada para pembaca khususnya bagi penulis secara pribadi.

Malang, 14 Juni 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Shiddiq, Muhammad Aufa. 2019. **Ekstraksi Kata Kunci pada Artikel menggunakan Metode *TextRank***. Skripsi. Jurusan Teknik Informatika. Fakultas Sains dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing: (I) Dr. Cahyo Crysdian
(II) M. Imamudin, Lc., MA

Kata Kunci: Ekstraksi Kata Kunci, *TextRank*, *Stemming*, *Parts-of-Speech Tagging*, *Multiword expression candidate*, *graph*

Saat ini dalam menentukan kata kunci untuk artikel ilmiah masih dilakukan secara manual. Hal ini menyebabkan proses penentuan kata kunci menjadi tidak efektif dan membutuhkan banyak waktu terlebih lagi jika jumlah artikelnya sangat banyak. Dari permasalahan tersebut, ekstraksi kata kunci diperlukan untuk menemukan kata kunci dari banyak artikel secara otomatis dalam waktu yang singkat. Berbagai penelitian mengenai ekstraksi kata kunci telah dikembangkan dengan memanfaatkan berbagai metode. Salah satu metode yang banyak digunakan dalam proses ekstraksi kata kunci yaitu *TextRank*. Pada penelitian ini ekstraksi kata kunci dikembangkan menggunakan metode *TextRank*. Proses ekstraksi terdiri dari enam tahapan antara lain *Tokenization*, *Stemming*, *Parts-of-Speech Tagging*, pencarian kandidat kata kunci, membuat data *graph* dan perankingan menggunakan *TextRank*. Peneliti melakukan modifikasi pada tahap membuat data *graph* dan perankingan *TextRank* untuk mendapatkan hasil yang lebih baik dengan waktu yang lebih singkat. Hasil pengujian sistem menggunakan data 50 artikel menunjukkan bahwa algoritma yang telah dimodifikasi memperoleh nilai akurasi lebih tinggi serta waktu ekstraksi yang lebih singkat dibandingkan dengan algoritma *TextRank* yang asli.

ABSTRACT

Shiddiq, Muhammad Aufa. 2019. **Keyword Extraction in Articles using TextRank Method**. Undergraduate Theses. Informatics Engineering Department. Faculty of Science and Technology. State Islamic University of Maulana Malik Ibrahim Malang.

Advisors: (I) Dr. Cahyo Crysdian
(II) M. Imamudin, Lc., MA

Keywords: Keyword Extraction, *TextRank*, *Stemming*, *Parts-of-Speech Tagging*, *Multiword expression candidate*, *graph*

Currently in determining keywords for scientific articles is still done manually. This causes the process of determining keywords to be ineffective and requires a lot of time especially if the number of articles is very large. Of these problems, keyword extraction is needed to find keywords from many articles automatically in a short time. Various studies on keyword extraction have been developed using various methods. One method that is widely used in the keyword extraction process is *TextRank*. In this study keyword extraction was developed using the *TextRank* method. The extraction process consists of six process including Tokenization, Stemming, *Parts-of-Speech Tagging*, searching for candidate keywords, creating graph data and ranking using *TextRank*. The researcher modified it at creating a data graph and *TextRank* ranking process to get better results with a shorter time. The results of testing using data from 50 articles shows that the modified algorithms achieve higher accuracy values and shorter extraction times than the original *TextRank* algorithm.

ملخص

صديق، محمد أوفى. ٢٠١٩. استخراج الكلمات الرئيسية في المقالة باستخدام طريقة رتبة النص (TextRank). البحث الجامعي. شعبة المعلوماتية. كلية العلوم والتكنولوجيا. الجامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج.

المشرف: (١) الدكتور جهيو كريسديان

(٢) محمد إمام الدين، الماجستير

الكلمات الرئيسية: استخراج الكلمات الرئيسية ، رتبة النص ، النابعة (Stemming)، توصيف أجزاء الكلام (*Parts-of-Speech Tagging*) ، مرشح التعبير الكلمات المتعددة ، الرسم البياني

في هذا العصر في تحديد الكلمات الرئيسية للمقالة العلمية مازال يقوم يدويا. يؤدي عملية تحديد الكلمات الرئيسية ليس لعا فعالة وتتطلب الكثير من الاوقات خاصةً إذا كان عدد المقالات كبيرًا جدا. من هذه المشكلات ، يلزم استخراج الكلمات الرئيسية للعثور على الكلمات الرئيسية من كثير المقالات تلقائيًا في وقت قصير. تطور العديد من الدراسات حول استخراج الكلمات الرئيسية باستخدام طرق مختلفة. إحدى الطرق الذى يستخدم كثيرا في عملية استخراج الكلمات الرئيسية هي رتبة النص. في هذا البحث، تطور استخراج الكلمات الرئيسية باستخدام طريقة رتبة النص. تتكون عملية الاستخراج من ست مراحل فهي الرمزية (Tokenization)، النابعة (Stemming)، توصيف أجزاء الكلام، البحث عن الكلمات الرئيسية المرشحة ، إنشاء الرسوم البيانية للبيانات والتصنيف باستخدام رتبة النص (TextRank). قام الباحث بتعديله في مرحلة إنشاء رسم بياني للبيانات وتصنيف رتبة النص للحصول على أفضل النتيجة في وقت أقصر.دلت نتائج اختبار النظام باستخدام بيانات ٥٠ مقالات إلى أن الخوارزمية المعدلة تحصل على أعلى قيم الدقة وأقصر أوقات الاستخراج مقارنةً بخوارزمية رتبة النص (TextRank) الأصلية

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di zaman sekarang ini teknologi berkembang sangat cepat. Salah satu bukti berkembangnya teknologi yaitu dirancangnya komputer. Sama seperti manusia, komputer juga dapat memiliki kecerdasan, kecerdasan di dalam komputer dibuat dengan cara menanamkan teknologi *Artificial Intelligence* di dalamnya. Maka dari itu, komputer sangat berguna untuk membantu sebagian pekerjaan manusia dan memudahkan urusannya. Karena sesungguhnya sesudah kesulitan pasti ada kemudahan, sebagaimana firman Allah SWT dalam QS. Al-Insyirah (5-6):

فَاِنَّ مَعَ الْعُسْرِ يُسْرًا ۝ اِنَّ مَعَ الْعُسْرِ يُسْرًا ۝

"Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan." (QS. Al-Insyirah (94): Ayat 5-6).

Dengan berkembangnya teknologi dan internet yang cepat, semakin banyak jumlah dokumen dan artikel yang beredar. Pada penulisan atau pengelolaan jurnal ilmiah, umumnya penulis/pengelola jurnal menentukan sendiri kata kunci dari artikel-artikel yang ada. Saat ini dalam menentukan kata kunci dari artikel ilmiah masih dilakukan secara manual. Hal ini menyebabkan proses penentuan kata kunci menjadi tidak efektif dan membutuhkan banyak waktu terlebih lagi jika jumlah artikelnya sangat banyak. Tidak semua penulis mampu membuat kata kunci

untuk artikel yang dibuatnya dan tidak semua kata kunci dapat dengan benar mewakili isi teks karena subjektifitas manusia sehingga kata kunci yang dihasilkan tidak bersifat general.

Kata kunci atau *keyword* merupakan kata-kata singkat yang dapat menggambarkan isi suatu artikel ataupun dokumen (Figueroa & Chen, 2014). Kata kunci (*keyword*) memiliki beberapa kegunaan. Menurut Zhang, An & Liu (2017), kata kunci memiliki fungsi-fungsi sebagai berikut: Dalam pencarian direktori, pengguna mudah untuk mencari konten yang sesuai atau relevan dengan target. Sebagai abstrak dari teks secara keseluruhan, pengguna dapat membaca dan menentukan apa yang dibahas oleh keseluruhan teks tersebut. Mesin pencari atau *search engine* mencari informasi yang dibutuhkan oleh pengguna berdasarkan kata kunci, yang dapat meningkatkan efisiensi dan membuat hasil pencarian lebih akurat.

Ekstraksi kata kunci adalah sebuah tahapan untuk dapat mengindentifikasi berbagai kumpulan teks pada suatu dokumen dan menemukan kata kunci yang tepat sesuai dengan topik pembahasan dari dokumen yang diolah (Li & Wang, 2014). Ekstraksi kata kunci memiliki peranan penting dalam bidang *Text Mining*. Dalam prakteknya, ekstraksi kata kunci manual memakan waktu dan hasilnya tidak dapat memenuhi kebutuhan yang berbeda dari pengguna yang berbeda. Maka dari itu, ekstraksi kata kunci otomatis diperlukan, yang telah menarik banyak perhatian dalam penelitian beberapa tahun terakhir (Wen, *et. al*, 2016).

Untuk membuat algoritma ekstraksi kata kunci menjadi lebih akurat, banyak penelitian yang telah dikembangkan dengan memanfaatkan berbagai metode. Salah satu metode yang banyak digunakan dalam proses ekstraksi kata kunci yaitu *TextRank*. Berdasarkan beberapa penelitian sebelumnya, akurasi dari algoritma *TextRank* terbukti cukup akurat dalam mengekstraksi kata kunci. Metode *TextRank* menggunakan pemodelan graf yang tidak membutuhkan data latih dalam pemrosesannya sehingga bisa lebih cepat.

Salah satu penelitian yang telah dilakukan sebelumnya, Ramadhiana (2017) mencoba meningkatkan algoritma *TextRank* dengan memodifikasi tahap *preprocessing*. dengan menerapkan *POS Tagging* dengan menggunakan metode *Hidden Markov Model* (HMM) dan *Multiword Expression Candidate*. Hasil pengujian menunjukkan bahwa akurasi yang didapat mencapai 33,54% dengan waktu ekstraksi minimal 6 (enam) detik per artikel. Dari hasil pengujian tersebut, nilai akurasi yang diperoleh tergolong masih cukup rendah dan juga waktu ekstraksi dirasa masih cukup lama untuk satu artikel.

Berdasarkan latar belakang di atas, maka penulis berniat mengajukan penelitian untuk membuat sebuah sistem untuk proses ekstraksi kata kunci pada artikel berbahasa Indonesia dengan mengimplementasikan algoritma *TextRank*. Algoritma *TextRank* yang diimplementasikan berpacu pada penelitian yang dilakukan oleh Ramadhiana (2017). Penulis melakukan modifikasi pada tahap *POS Tagging*, pembuatan *graph* dan perankingan. Hasil dari penelitian ini diharapkan

mampu memperoleh tingkat akurasi yang lebih tinggi dan juga waktu ekstraksi yang lebih singkat dari penelitian sebelumnya.

1.2 Pernyataan Masalah

1. Seberapa efektif pengimplementasian algoritma *TextRank* yang telah dimodifikasi pada sistem ekstraksi kata kunci artikel berbahasa Indonesia?
2. Seberapa efisien pengimplementasian algoritma *TextRank* yang telah dimodifikasi pada sistem ekstraksi kata kunci artikel berbahasa Indonesia?

1.3 Tujuan Penelitian

1. Untuk mengukur efektifitas pengimplementasian algoritma *TextRank* yang telah dimodifikasi pada sistem ekstraksi kata kunci artikel berbahasa Indonesia. Dalam hal ini pengukuran efektifitas dilakukan menggunakan parameter akurasi dengan rumus *F-measure* yang terdiri dari nilai *Precision* dan *Recall.*
2. Untuk mengukur efisiensi pengimplementasian algoritma *TextRank* yang telah dimodifikasi pada sistem ekstraksi kata kunci artikel berbahasa Indonesia. Dalam hal ini pengukuran efisiensi dilakukan menggunakan parameter waktu yang diperlukan proses ekstraksi.

1.4 Batasan Masalah

1. Teks yang diteliti adalah artikel ilmiah berbahasa Indonesia dalam ruang lingkup *Computer Science* terutama yang memiliki kata kunci yang telah

ditentukan manual oleh penulisnya sebagai evaluasi sistem.

2. Kata kunci terbentuk dari satu kata atau multi kata/frasa yang terdiri dari dua atau tiga kata.
3. Jenis format file dokumen artikel ilmiah yang diolah yaitu .PDF

1.5 Manfaat Penelitian

1. Mengetahui seberapa akurat ekstraksi kata kunci dari dokumen berbahasa Indonesia menggunakan algoritma *TextRank*.
2. Memberikan rekomendasi kata kunci yang berbentuk satu kata atau multi kata/frasa secara otomatis dan cepat.
3. Memudahkan penulis artikel untuk menemukan kata kunci dari artikel yang ditulisnya.

1.6 Sistematika Penulisan

Dalam penulisan skripsi ini terdiri dari 5 (lima) bab sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang penjelasan penulis mengenai latar belakang, pernyataan masalah, tujuan dan manfaat penelitian.

BAB II TINJAUAN PUSTAKA

Bab ini berisi berbagai landasan teori yang digunakan untuk memahami permasalahan yang ada pada penelitian ini. Dimana teori-teori tersebut seperti

teori umum tentang ekstraksi kata kunci, *POS tagging*, *multiword expression*, dan teori dasar dari algoritma *TextRank*.

BAB III PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan dibahas mengenai analisis permasalahan penelitian dan penjelasan tentang rancangan struktur program dan antarmuka sekaligus implementasi dari rancangan struktur program dan antarmuka aplikasi ekstraksi kata kunci yang dibuat.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi penjelasan pengimplementasian sistem seperti gambaran antarmuka aplikasi yang dibuat dan pengujian aplikasi apakah berhasil dijalankan dengan baik serta menemukan *error* yang muncul didalam sistem yang dibuat. Pada bab ini juga membahas hasil pengujian berupa persentase akurasi keberhasilan sistem serta tingkat efisiensi dari sistem.

BAB V PENUTUP

Pada bab ini akan dijabarkan beberapa kesimpulan dari perancangan sistem dan saran untuk pengembangan penelitian lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Ekstraksi Kata Kunci

Hu & Wu (2006) melakukan penelitian untuk ekstraksi kata kunci dari novel. Mereka menggunakan algoritma *Position Weight (PW)* untuk memanfaatkan fitur linguistik yang menyatakan nilai dari suatu kata berdasarkan posisi dari kata tersebut dalam dokumen. Terdapat tiga metode yang digunakan yaitu *Term Frequency Inverse Term Frequency (TFITF)*, *Position Weight Inverse Position Weight (PWIPW)*, dan *CHI-Square (χ2)*. Hasil pengujian menunjukkan bahwa algoritma *PW* memiliki potensi yang besar untuk ekstraksi kata kunci, karena menghasilkan hasil yang lebih baik daripada pendekatan lain yang ada.

Sari & Purwarianti (2014) membangun sebuah sistem ekstraksi kata kunci otomatis dengan tiga tahapan, yakni praproses, translasi, dan pencocokan kandidat kata kunci dengan daftar kata kunci. Penelitian ini menggunakan 3 metode pembobotan, yaitu *TF*, *TFxIDF* dan *WIDF*. Pengujian dilakukan dengan menggunakan 33 data artikel yang diambil dari kumpulan jurnal koleksi PDII LIPI. Hasil pengujian menunjukkan bahwa metode *TFxIDF* mendapatkan hasil terbaik. Peneliti menggunakan algoritma *levensthein* untuk menyempurnakan hasil kata kunci.

Li & Wang (2014) melakukan penelitian untuk ekstraksi kata kunci menggunakan metode *TextRank*. Mereka mencoba meningkatkan algoritma *TextRank* dengan menggunakan domain pengetahuan untuk artikel ilmiah bahasa Cina. Pada penelitiannya ini, algoritma *TextRank* dimodifikasi pada bagian *preprocessing*nya dimana pada penentuan kandidat kata kuncinya menggunakan metode *Document Frequency Accessor Variety* (DF-AV) karena menurutnya jika menggunakan *POS tagging* tidak cukup akurat untuk diaplikasikan pada jurnal ilmiah bahasa Cina. Kemudian pada penelitiannya ini digunakan pengetahuan terhadap kata kunci yang sudah dikenali pada beberapa domain pengetahuan dengan menghitung panjang kata kunci, komponen kata kuncinya, dan juga frekuensi tertinggi kata kunci untuk menggantikan fungsi *thesaurus* seperti pada domain pengetahuan beberapa penelitian sebelumnya. Akurasi yang didapatkan ternyata mampu lebih tinggi dari algoritma TF-IDF.

Figueroa & Chen (2014) menggunakan metode *HybridRank* yaitu metode gabungan *TextRank* dan KEA untuk ekstraksi frase kunci dari abstrak jurnal berbahasa Inggris. Pada penelitian ini menggunakan koleksi dokumen dari *IEEE Xplore* sebanyak 1606 dokumen dan koleksi dokumen *Hulth* 2003 sebanyak 2000 dokumen yang didalamnya mengandung bagian abstrak. Pada penelitiannya ini dapat menghasilkan daftar keyphrase dengan kualitas terbaik untuk artikel pendek berupa abstrak tersebut.

Wen, Zhang & Yuan (2016) menggunakan model graph untuk mencari kandidat kata kunci berdasarkan metode *TextRank*. Kemudian menggunakan metode Word2Vec untuk menghitung similaritas antar kalimat yang digambarkan sebagai bobot transisi antar dua kata. Perhitungan dilakukan terhadap setiap kata kemudian mengambil N kandidat kata kunci yang memiliki nilai bobot yang tinggi sebagai outputnya. Hasil pengujian menunjukkan bahwa algoritma *TextRank* yang dimodifikasi menggunakan metode *Word2Vec* dapat meningkatkan performa dari ekstraksi kata kunci secara umum.

Sama seperti penelitian sebelumnya, Zhao, *et. al* (2017) menggunakan metode *Word2Vec* dan *TextRank*. Mereka menggunakan metode tersebut untuk mencari kandidat kata kunci dalam teks pendek media sosial. Hasil pengujian menunjukkan bahwa pendekatan metode tersebut memiliki efek yang lebih baik pada teks singkat media sosial, dan waktu yang dihabiskan lebih singkat. Atas dasar ini, teks yang dikirim di Twitter dari pengguna digunakan untuk mengekstraksi kata kunci dan digunakan sebagai landasan untuk mengetahui profil pengguna.

Zhang, An, & Liu (2017) melakukan penelitian dan implementasi terhadap *Keyword Extraction Algorithm* (KEA) menggunakan algoritma TF-IDF dan *TextRank*. Data dokumen diperoleh dari halaman web menggunakan aplikasi crawler. Mereka melakukan modifikasi terhadap metode TF-IDF dan *TextRank*. Hasil pengujian menunjukkan bahwa tingkat akurasi dari algoritma yang ditingkatkan untuk mengekstraksi kata kunci lebih baik daripada algoritma

TextRank yang telah banyak digunakan. Algoritma tersebut memiliki stabilitas yang baik dan mencapai tingkat yang diharapkan dari algoritma ekstraksi kata kunci.

Ramadhiana (2017) melakukan penelitian untuk ekstraksi kata kunci menggunakan metode *TextRank* untuk mengekstraksi dokumen teks berbahasa Indonesia. Beliau memodifikasi tahap *preprocessing* pembentukan kandidat kata kunci dari algoritma *textrank* tersebut menggunakan aturan *multiword expression candidate*. Tahapan keseluruhan metode yang digunakan pada penelitian tersebut yaitu *preprocessing*(*text cleaning, tokenizing, case folding, stopword removal, POS tagging, candidates multiword extraction*), ekstraksi kata kunci dan tahapan terakhir yaitu *postprocessing* untuk pemfilteran kata kunci yang terlalu umum. Hasil dari penelitian yang dilakukan menunjukkan bahwasanya *textrank* dengan *multiword expression candidate* memiliki waktu ekstraksi yang lebih cepat dan persentase akurasi *recall* yang sedikit lebih tinggi dibandingkan algoritma *textrank* biasa pada top-15 kata kunci.

2.2 Parts-of-Speech Tagging

Parts-of-Speech (POS) *tagging* merupakan proses pemberian label pada kata yang ada pada suatu kalimat dengan kelas katanya masing-masing. Label-label kata tersebut terdiri dari kata benda (*noun*), kata kerja (*verb*), kata keterangan (*adverb*), kata sifat (*adjective*) dan lain sebagainya. Untuk tahapan ekstraksi kata kunci, *POS tagging* dapat digunakan untuk mengambil kata yang hanya merupakan kata benda (*noun*) atau hanya merupakan kata sifat ataupun mengambil dan

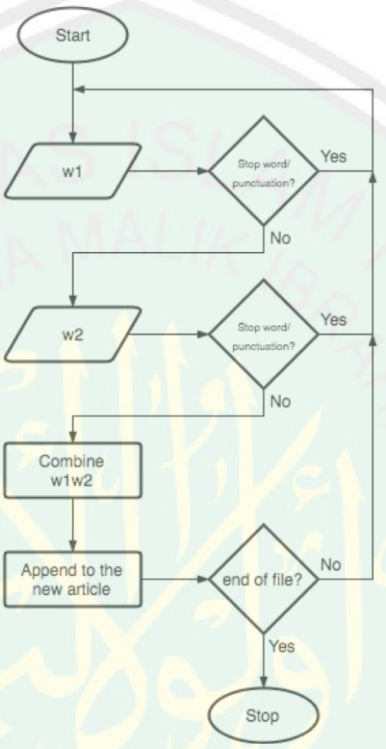
memfilter semua jenis kata untuk perhitungan kata sebagai kata kunci begitupun fungsinya pada aplikasi *text mining* lainnya (Wicaksono & Purwarianti, 2010).

Wicaksono & Purwarianti (2010) melakukan penelitian terhadap Parts-of-Speech Tagger berbasis *Hidden Markov Model* (HMM) untuk meningkatkan akurasi dari proses *POS Tagging*. Beberapa metode telah digunakan. Metode pertama menggunakan *affix tree* yang mencakup kata suffix dan prefix. Yang kedua adalah menggunakan HMM sebagai salah satu fitur untuk *POS Tagging*. Metode terakhir adalah menggunakan leksikon tambahan (dari KBBI-Kateglo) untuk membatasi tag kandidat yang dihasilkan oleh *affix tree*. Hasil pengujian menunjukkan bahwa akurasi terbaik adalah 96,50% dengan 99,4% untuk kata-kata dalam kosakata dan 80,4% untuk kata *Out-of-Vocabulary* (OOV). Percobaan menunjukkan bahwa *affix tree* dan leksikon tambahan efektif dalam meningkatkan akurasi proses *POS Tagging*, sementara penggunaan *POS tag* yang berhasil tidak memberikan banyak perbaikan pada penanganan OOV.

2.3 Multiword Expression

Multiword expression atau biasa disebut frasa merupakan gabungan dua atau lebih kata yang terhubung dan membentuk makna baru. Gunawan, Amalia, & Charisma (2016) menggunakan suatu aturan metode *multiword expression candidate* dimana metode ini memanfaatkan pemfilteran *stopword* dan tanda baca. Tahapan utama dari metode ini yaitu kata hasil dari proses tokenisasi selanjutnya dilakukan pemfilteran *stopword* dan tanda baca. Kata yang termasuk *stopword* dan

tanda baca akan diganti dengan tanda baca tertentu seperti tanda baca titik (.). Kemudian kata-kata yang didahului tanda baca ataupun *stopword* tersebut digabungkan menjadi satu frasa baru. Alur dari proses ekstrasi *multiword expression candidate* dapat dilihat pada **Gambar 2.1**.



Gambar 2.1 Flowchart Metode *Multiword Expression Candidates* (Gunawan, et al. 2016)

2.4 TextRank

TextRank merupakan algoritma peringkat berbasis *graph*. Algoritma ini pada dasarnya adalah metode untuk menentukan pentingnya *vertex* dalam *graph*, berdasarkan informasi global yang diambil secara rekursif dari seluruh graph. Ide dasar yang diterapkan oleh model peringkat berbasis graph adalah "voting" atau

"rekomendasi". Ketika satu *vertex* terhubung dengan yang lain, pada dasarnya memberikan dukungan untuk *vertex* yang lain (Mihalcea & Tarau, 2004).

Algoritma *TextRank* menganggap bahwa jika sebuah kata terhubung ke kata lain melalui *edge* maka kata tersebut memberikan dukungan untuk kata yang terhubung dengannya. Oleh karena itu nilai bobot dari sebuah kata tergantung pada jumlah dukungan yang diperolehnya dan nilai bobot kata-kata lain yang memilihnya (Li & Zhao, 2016).

Pada umumnya, *graph* terarah dinyatakan sebagai *G = (V, E)*, Graph terdiri dari himpunan *vertex V* dan himpunan *edge E*, dimana *E* merupakan subset dari *V***V*. Untuk *vertex Vi*, *In(Vi)* merupakan himpunan *vertex* yang terhubung masuk ke *vertex Vi* (*predecessor*), dan *Out(Vi)* merupakan himpunan *vertex* yang terhubung keluar *vertex Vi* (*successor*). Persamaan untuk menghitung skor *Vi* dinyatakan sebagai berikut:

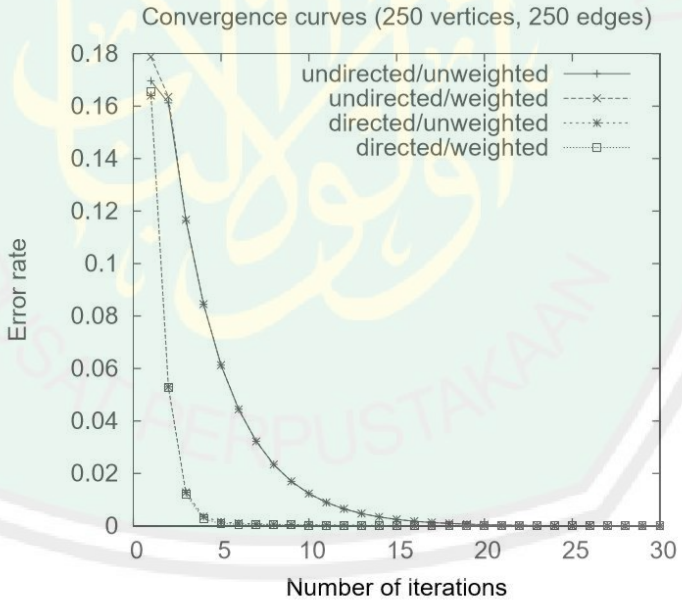
$$S(Vi) = (1 - d) + d \times \sum_{j \in In(Vi)} \frac{1}{|Out(Vj)|} S(Vj) \qquad (2.1)$$

dimana *d* merupakan nilai *damping factor* yang dapat ditentukan nilainya antara 0 sampai 1 (Mihalcea & Tarau, 2004). Untuk menghitung skor *Vi*, diperlukan nilai awal dari tiap *vertex* dalam *graph*. Kemudian secara rekursif menghitung skor tiap *vertex* hingga nilainya konvergen. Setelah setiap titik konvergen, skor akhir dari *vertex* mewakili bobot *vertex* dalam *graph* (Li & Zhao, 2016).

Walaupun pada dasarnya algoritma *TextRank* diaplikasikan pada *graph* berarah, proses perhitungan secara rekursif dapat diaplikasikan pada *graph* tak berarah. Dapat diartikan bahwa jumlah vertex yang masuk sama dengan jumlah vertex yang keluar. Mihalcea & Tarau (2004) juga mengenalkan persamaan *TextRank* yang lain untuk perhitungan bobot pada *edge*. Persamaan ini dapat diaplikasikan untuk perhitungan bobot pada *vertex*, baik *graph* yang berarah maupun tidak berarah.

$$\mathrm{WS}(\mathrm{V}i) = (1-d) + d \times \sum_{Vj \in In(Vi)} \frac{wji}{\sum_{Vk \in Out(Vj)} wjk} WS(\mathrm{V}j) \quad (2.2)$$

Gambar 2.2 menunjukkan perhitungan algoritma TextRank untuk tiap jenis graph dan bobot.



Gambar 2.2 Kurva Konvergensi pada 250 vertex dan 250 edge (Mihalcea & Tarau, 2004)

BAB III

PERANCANGAN DAN IMPLEMENTASI SISTEM

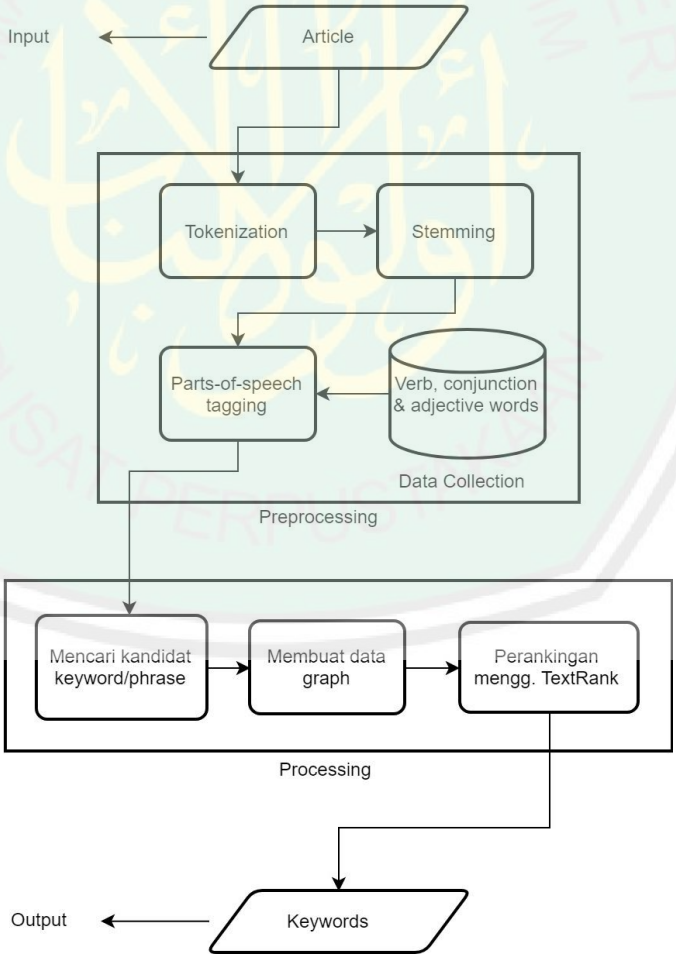
3.1 Data

Data yang digunakan pada penelitian ini yaitu artikel ilmiah di bidang *Computer Science* berbahasa Indonesia. Jumlah keseluruhan artikel yang digunakan yaitu 50 artikel. Keseluruhan artikel didapat dari jurnal CESS (*Journal of Computer Engineering, System and Sains*) yang dipublikasikan oleh Universitas Negeri Medan. Artikel diunduh secara manual dari website Garuda (Garba Rujukan Digital) (http://garuda.ristekdikti.go.id/journal). Keseluruhan artikel berekstensi file PDF dan telah memiliki kata kunci manual yang ditentukan oleh masing-masing penulisnya. Kata kunci manual akan dimanfaatkan sebagai data uji sistem.

3.2 Desain Sistem

Desain sistem ekstraksi kata kunci terdiri dari dua tahapan, yaitu tahap *preprocessing* dan tahap *processing*. Tahap *preprocessing* adalah tahapan untuk mengolah paragraf input menjadi data yang siap diolah pada tahap *processing*. Pada tahap *preprocessing* terdapat tiga proses, yaitu *Tokenization*, *Stemming*, dan *Parts-of-Speech Tagging*. Otuput dari tahap *preprocessing* adalah data kata dari paragraf yang sudah memiliki tipe kata berdasarkan kata dalam kalimat. Tahap *processing* adalah tahap memproses data kata yang telah diproses sebelumnya untuk mencari kandidat kata kunci dalam paragraf yang memiliki skor tertinggi. Tahap *processing*

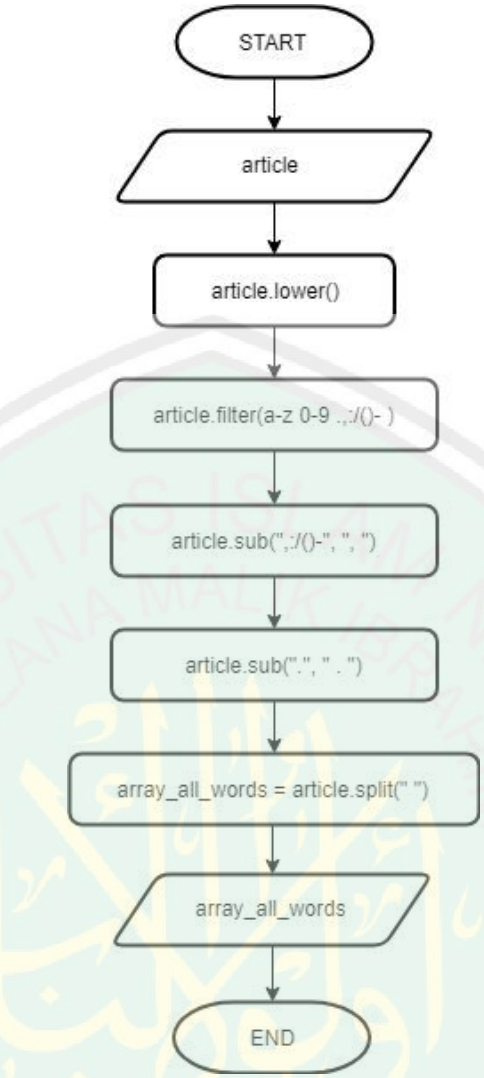
terdiri dari tiga proses, yaitu mencari kandidat kata kunci/frasa kunci, membuat data graph kata kunci, dan perankingan kata kunci menggunakan metode *TextRank*. Pada penelitian ini penulis mengimplementasikan dua algoritma *TextRank* yang berbeda, yakni *Modified TextRank* dan *Original TextRank*. *Modified TextRank* merupakan algoritma *TextRank* yang telah diubah oleh penulis sedangkan *Original TextRank* merupakan algoritma *TextRank* yang asli berdasarkan penelitian yang dilakukan oleh Ramadhiana (2017). Perbedaan algoritma hanya terletak pada tahap pembuatan data graph dan perankingan. Desain sistem ekstraksi kata kunci ditunjukkan pada **Gambar 3.1**.



Gambar 3.1. Desain Sistem

3.2.1 *Tokenization*

Tokenization adalah proses memecah kalimat atau paragraf menjadi himpunan kata. Proses ini dilakukan dengan melakukan iterasi berdasarkan jumlah kalimat output dari proses sebelumnya. Proses *Tokenization* diawali dengan mengubah semua huruf pada himpunan kata menjadi huruf kecil. Kemudian menghilangkan beberapa karakter khusus yang tidak diperlukan, lalu mengubah karakter khusus yang diperbolehkan menjadi tanda baca koma dan spasi kecuali karakter titik. Pengubahan karakter khusus ini bertujuan untuk memisahkan kata yang bersebelahan dengan karakter khusus dengan kata sebelum atau sesudahnya dalam proses mencari kandidat frasa kunci. Kemudian memotong kalimat menggunakan tanda baca spasi sebagai titik potongnya. Output dari proses *Tokenization* adalah data *array* kata dari semua kalimat. Proses *Tokenization* ditunjukkan pada **Gambar 3.4**.

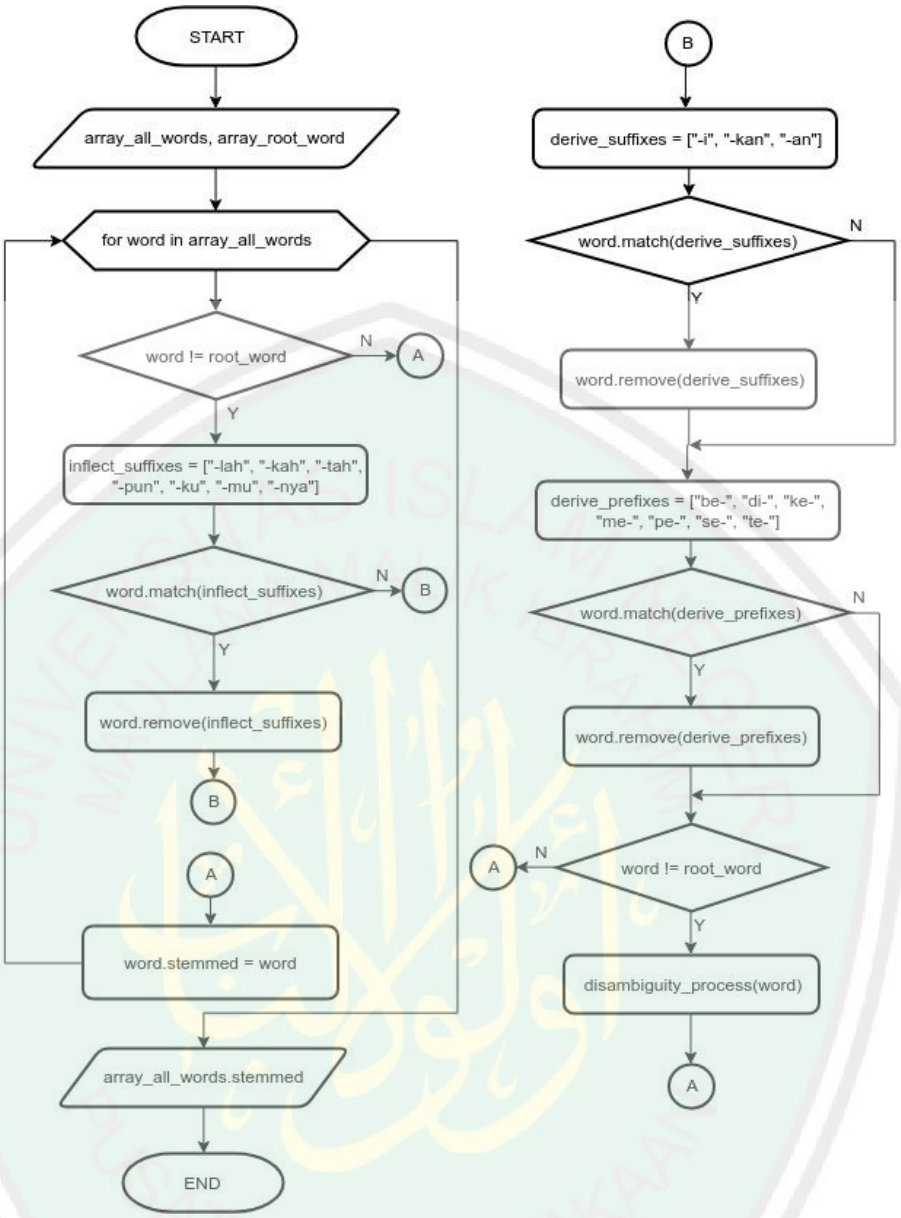


Gambar 3.2. *Flowchart Tokenization*

3.2.2 *Stemming*

Stemming adalah proses menghilangkan imbuhan dari kata untuk mendapatkan kata dasarnya. Pada sistem ini menggunakan algoritma *stemming* Nazief & Adriani. Proses *stemming* pada penelitian ini diawali dengan proses evaluasi seluruh kata pada kamus kata dasar. Jika kata yang dievaluasi ditemukan pada kamus tersebut, maka kata yang dievaluasi merupakan kata dasar dan proses *stemming* berhenti. Namun jika kata yang dievaluasi tidak ditemukan, maka

selanjutnya dilakukan tahap penghilangan *Inflection Suffixes*. Imbuhan yang merupakan *Inflection Suffixes* adalah ("-lah", "-kah", "-tah", "-pun", "-ku", "-mu", "-nya"). Langkah selanjutnya yaitu mengecek kata pada kamus kata dasar, jika kata tersebut terdapat dalam kamus maka algoritma berhenti. Jika kata tidak terdapat dalam kamus dan tidak terdapat *Inflection Suffixes* maka lanjut ke tahap menghilangkan *Derivation Suffixes. Derivation Suffixes* merupakan imbuhan yang terdapat di akhir kata. Imbuhan yang merupakan *Derivation Suffixes* adalah ("-i", "-an", "-kan",). Alur proses *Stemming* ditunjukkan pada **Gambar 3.3**.



Gambar 3.3. *Flowchart Stemming*

Kemudian setelah dilakukan penghapusan imbuhan pada kata yang dievaluasi, kata tersebut di evaluasi kembali pada kamus kata dasar. Apabila ditemukan pada kamus kata dasar, maka proses *stemming* berhenti. Jika kata tidak terdapat dalam kamus dan tidak terdapat *Derivation Suffixes* maka lanjut ke tahap menghilangkan *Derivation Prefixes. Derivation Prefixes* merupakan imbuhan yang terdapat di awal

kata. Imbuhan yang merupakan *Derivation Prefixes* adalah ("be-", "di-", "ke-", "pe-", "me-", "se-", "te-"). Kemudian setelah dilakukan penghapusan *Derivation Prefixes*, kata di evaluasi kembali pada kamus kata dasar. Jika kata yang dievaluasi ditemukan pada kamus kata dasar, maka proses *stemming* berhenti. Jika kata tidak terdapat dalam kamus dan tidak terdapat *Derivation Prefixes*, maka masuk ke dalam proses prediksi tabel disambiguitas. Proses ini menghilangkan imbuhan yang tidak terdapat dalam daftar *Inflection Suffixes*, *Derivation Suffixes*, maupun *Derivation Prefix*. Jika dari semua proses di atas tidak menemukan kata dasarnya maka proses *stemming* berhenti dan mengembalikan kata asalnya.

3.2.3 *Parts-of-Speech Tagging*

Parts-of-Speech Tagging (POS Tagging) adalah proses melabeli kata dengan tipe katanya masing-masing. Input dari proses ini adalah kata dasar dari proses *stemming* beserta kata asalnya dan kamus kata dasar bahasa Indonesia. Kamus kata dasar bahasa Indonesia diperoleh dari situs http://bahtera.org/. Bahtera adalah kamus bahasa Indonesia yang menjadi rujukan sesuai Kamus Besar Bahasa Indonesia. Kata dasar yang terdapat di Bahtera sebanyak 28.526 kata. Contoh isi kamus kata dasar bahasa Indonesia terdapat pada **Tabel 3.1**.

Tabel 3.1. Contoh isi kamus kata dasar bahasa Indonesia

id_katadasar	katadasar	tipe_katadasar
1	abad	Nomina
2	abadi	Adjektiva
3	abadiah	Nomina
4	abah	Nomina

Sedangkan untuk jenis-jenis tipe kata terdapat dalam **Tabel 3.2**. Khusus untuk penelitian ini penulis menambahkan satu tipe baru yaitu Verba-Definitif, Separator dan Titik. Verba-Definitif merupakan verba yang menjelaskan definisi dari subjek yang dimaksud. Penambahan tipe ini bertujuan agar meningkatkan akurasi dari sistem ekstraksi kata kunci. Sedangkan Separator merupakan tanda baca yang terdapat dalam kalimat. Separator digunakan sebagai salah satu pembatas untuk mencari frasa dalam kalimat. Dan titik digunakan sebagai pembatas antar kalimat.

Tabel 3.2. Jenis-jenis tipe kata

No	Tipe kata
1	Nomina
2	Adjektiva
3	Verba
4	Adverbia
5	Preposisi
6	Interjeksi
7	Konjungsi
8	Pronomina
9	Numeralia
10	Lain-lain
11	Verba-Definitif
12	Separator
13	Titik

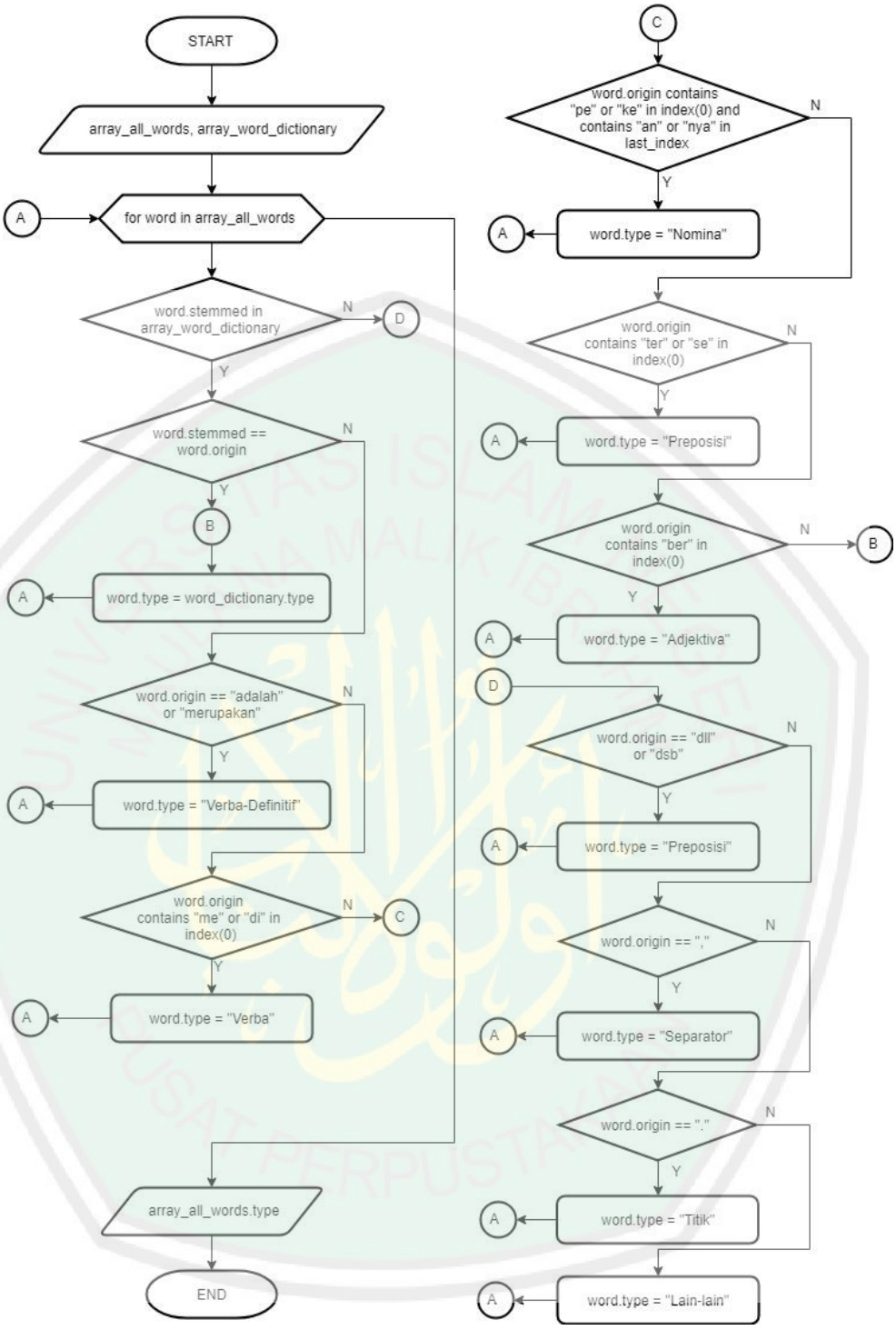
Proses *POS tagging* terdiri dari beberapa langkah sebagai berikut:

1. Melakukan iterasi berdasarkan jumlah kata dalam paragraf input.
2. Jika kata dasar terdapat dalam tabel tipe kata maka lanjut ke langkah 3, jika tidak maka lanjut ke langkah 9.
3. Jika kata dasar sama dengan yang terdapat di tabel tipe kata maka

mengembalikan tipe kata aslinya, jika tidak maka lanjut ke langkah 4.

4. Jika kata asal sama dengan "adalah" atau "merupakan" maka mengembalikan tipe "Verba-Definitif", jika tidak maka lanjut ke langkah 5.
5. Jika kata asal terdapat imbuhan "me-" atau "di-" maka mengembalikan tipe "Verba", jika tidak maka lanjut ke langkah 6.
6. Jika kata asal terdapat imbuhan ("pe-" atau "ke-") dan ("-an" atau "-nya") maka mengembalikan tipe "Nomina", jika tidak maka lanjut ke langkah 7.
7. Jika kata asal terdapat imbuhan "ter-" atau "se-" maka mengembalikan tipe "Preposisi", jika tidak maka lanjut ke langkah 8.
8. Jika kata asal terdapat imbuhan "ber-" maka mengembalikan tipe "Adjektiva", jika tidak maka mengembalikan tipe kata aslinya.
9. Jika kata asal sama dengan "dll" atau "dsb" maka mengembalikan tipe "Preposisi", jika tidak maka lanjut ke langkah 10.
10. Jika kata asal sama dengan titik (.) maka mengembalikan tipe "Titik", jika tidak maka lanjut ke langkah 11.
11. Jika kata asal sama dengan koma (,) maka mengembalikan tipe "Separator", jika tidak maka mengembalikan tipe "Lain-lain".
12. Jika proses iterasi berhenti maka proses berhenti dan mengembalikan array kata dasar, kata asal dan tipe kata.

Proses *POS Tagging* ditunjukkan pada **Gambar 3.6**.



Gambar 3.4. *Flowchart Parts-of-Speech Tagging*

3.2.4 Mencari kandidat kata kunci/frasa kunci

Proses ini menggunakan metode *multiword expression candidate* yang digunakan oleh Gunawan, Amalia, & Charisma (2016) untuk ekstrasi frasa dalam

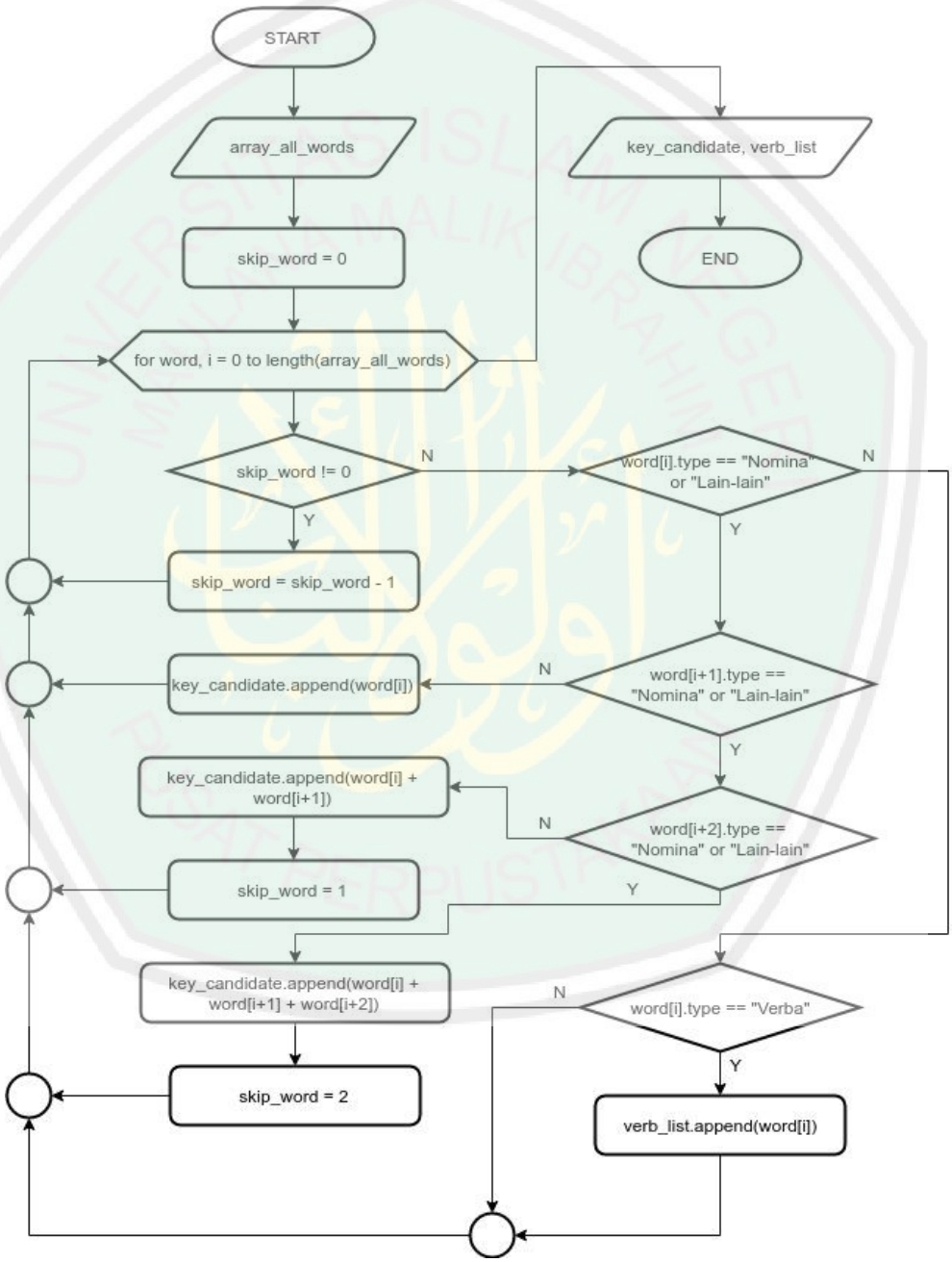
paragraf. Frasa yang diperoleh dari metode tersebut terdapat beberapa frasa yang tidak memiliki makna atau tidak sesuai. Maka dari itu, dalam penelitian ini metode dimodifikasi pada bagian pembentukan frasa dimana kandidat kata kunci dan frasa kunci dibentuk berdasarkan tipe kata "Nomina" dan "Lain-lain". Penggunaan dua tipe tersebut diharapkan mampu menghasilkan frasa yang memiliki makna dan lebih baik dari penelitian yang telah ada.

Kata verba diperoleh berdasarkan tipe kata "Verba". Kata yang bertipe selain "Nomina", "Verba" dan "Lain-lain" dilabeli sebagai stopwords. Kata kunci terdiri dari satu kata, sedangkan frasa kunci terdiri dari dua atau tiga kata. Proses ini mengembalikan dua data *array* yakni *key_candidate* dan *verb_list*. *Key_candidate* berisi daftar kandidat kata dan frasa kunci, sedangkan *verb_list* berisi daftar kata verba yang terdapat dalam paragraf. Diagram alur proses mencari kandidat kata kunci/frasa kunci ditunjukkan pada **Gambar 3.5**. Berikut ini langkah-langkahnya:

1. Melakukan iterasi terhadap semua kata dalam paragraf.
2. Jika bertemu dengan kata yang bertipe "Nomina" atau "Lain-lain", maka kata tersebut disimpan dan lanjut ke langkah 3. Jika tidak maka lanjut ke langkah 5.
3. Jika kata setelahnya bertipe "Nomina" atau "Lain-lain", maka kata tersebut disimpan dan lanjut ke langkah 4. Jika tidak maka kata sebelumnya ditambahkan ke *array key_candidate*.
4. Jika kata setelahnya bertipe "Nomina" atau "Lain-lain", maka tiga kata tersimpan

ditambahkan ke *array* key_candidate. Jika tidak maka dua kata sebelumnya ditambahkan ke *array key_candidate*.

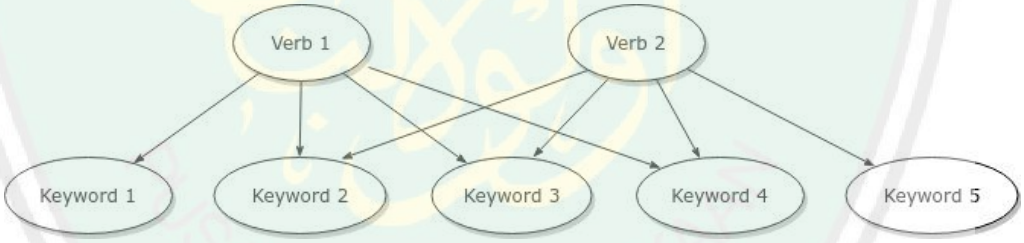
5. Jika bertemu dengan kata “Verba”, maka mengembalikan kata tersebut sebagai Verba dan ditambahkan ke *array verb_list*. Jika tidak maka iterasi dilanjutkan.



Gambar 3.5. *Flowchart* mencari kandidat kata kunci

3.2.5 Membuat data *graph*

Pembuatan data *graph* terdiri dari dua macam, yakni *graph* berarah dan *graph* tak berarah. *Graph* berarah digunakan untuk algoritma *Modified TextRank* sedangkan *graph* tak berarah digunakan untuk algoritma *Original TextRank*. Pada *graph* tak berarah, *graph* dibangun dari *array key_candidate* dan *verb_list* yang diperoleh dari proses sebelumnya. Setiap kata dan frasa diubah ke bentuk *node* dan menambahkan *arc* berdasarkan *node* yang memiliki hubungan dengan *node* tersebut. Proses ini bertujuan untuk mencari hubungan *predecessor* dan *successor*. *Predecessor* disini adalah *node* yang berisi kata bertipe Verba dan *successor* merupakan *node* yang berisi kandidat kata atau frasa kunci. Model dari *graph* kandidat kata kunci ditunjukkan pada **Gambar 3.6**.



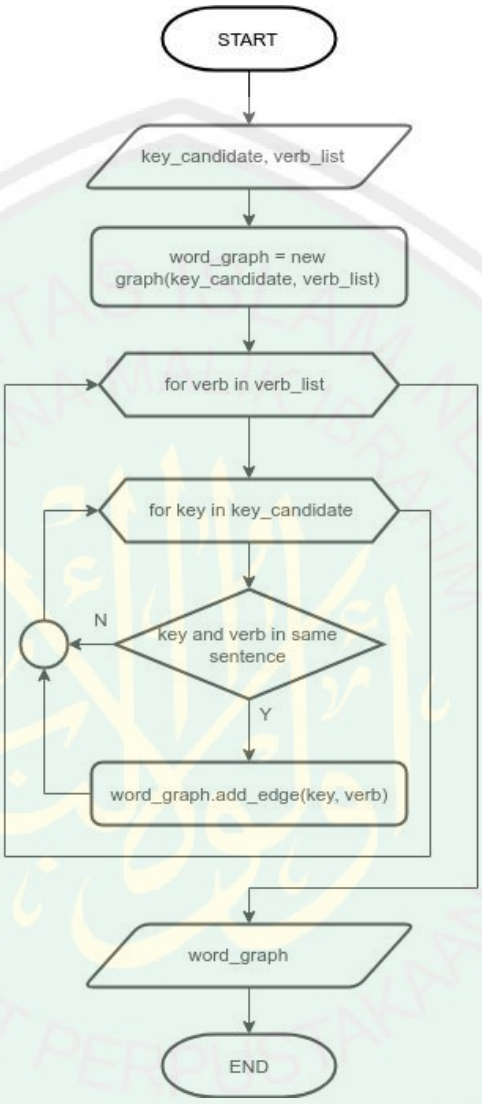
Gambar 3.6. *Graph* kandidat kata kunci

Berikut ini langkah-langkah membuat data *graph*:

1. Membuat objek *graph* baru dengan parameter *key_candidate* dan *verb_list*.
2. Melakukan iterasi item *verb* terhadap *array verb_list*.
3. Melakukan iterasi item *key* terhadap *array key_candidate*.
4. Jika *key* dan *verb* dalam satu kalimat yang sama, maka tambahkan *arc* dari *key*

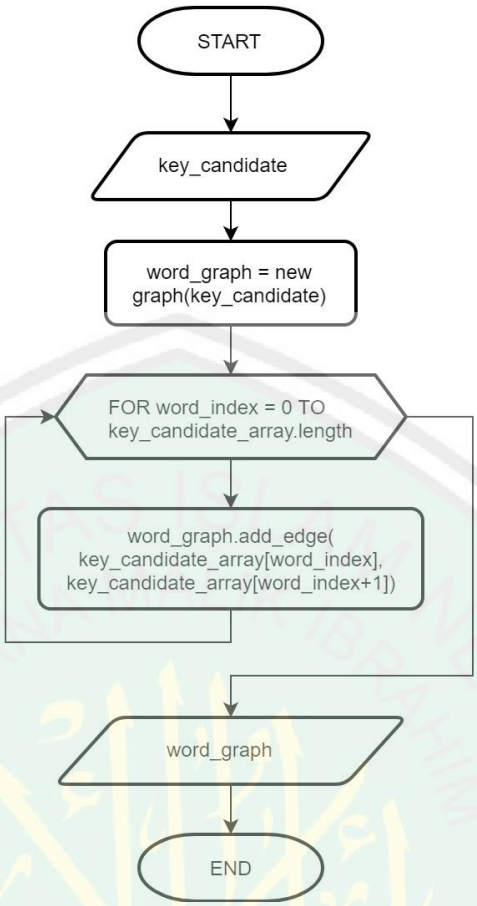
ke *verb*. Jika tidak maka iterasi dilanjutkan.

Berikut ini merupakan *flowchart* proses membuat data *graph* berarah.



Gambar 3.7. *Flowchart* membuat data *graph* berarah

Proses pembentukan *graph* tak berarah menghasilkan *graph* yang lebih sederhana karena hanya terdapat *node* kandidat kata kunci. *Node* yang terhubung merupakan kata yang muncul bersamaan dengan batas dua kata. *Flowchart* membuat data *graph* tak berarah ditunjukkan pada **Gambar 3.8**.



Gambar 3.8. *Flowchart* membuat data *graph* tak berarah

3.2.6 Perankingan

Perankingan dilakukan dengan menggunakan rumus *TextRank*. Berikut ini adalah rumus *TextRank*.

$$S(Vi) = (1-d) + d \times \sum_{j \in In(Vi)} \frac{1}{|Out(Vj)|} S(Vj) \quad (3.1)$$

Dimana S(Vi) merupakan skor dari *node* Vi dan d merupakan nilai *damping factor* dengan nilai default 0.85. Dalam penelitian ini *node* Vi merupakan *node* kandidat kata kunci dan *node* Vj merupakan node kata verba. In(Vi) merupakan *node* kata

verba (Vj) yang mengarah ke *node* Vi, Out(Vj) merupakan *node* kandidat kata kunci yang berhubungan dengan *node* Vj. S(Vj) merupakan skor dari *node* Vj.

Pada penelitian yang dilakukan oleh Mihalcea & Tarau (2004), perhitungan skor dilakukan ke setiap kata dengan nilai awal S(Vj) 1. Proses perhitungan dilakukan hingga nilai skor tiap kata konvergen. Perhitungan ini memerlukan iterasi yang banyak sehingga membutuhkan waktu yang lama. Dalam penelitian ini penulis melakukan modifikasi terhadap inisialisasi nilai S(Vj) dan iterasinya. Nilai S(Vj) diberi nilai yang berbeda-beda bergantung pada posisi kata dan jenis katanya. Berikut ini merupakan tabel nilai S(Vj).

Tabel 3.3. Tabel nilai S(Vj).

No	Penjelasan	Nilai
1	Kata Vj bertipe "Verba-Definitif"	2
2	Posisi kata Vj terdapat pada indeks leksikal 0 dalam kalimat	1
3	Posisi kata Vj terdapat pada indeks leksikal 1 dalam kalimat	0,5
4	Nilai default	1

Proses iterasi dalam penelitian ini hanya dilakukan satu kali pada *node* kandidat kata kunci. Dengan melakukan modifikasi ini diharapkan waktu eksekusi dari algoritma yang dibuat lebih cepat dari penelitian yang telah ada dan juga tingkat akurasi yang dihasilkan meningkat.

Langkah-langkah perankingan dijelaskan sebagai berikut:

1. Melakukan iterasi setiap *node* di dalam *graph*.

2. Jika *node i* berupa kandidat kata atau frasa kunci, maka lanjut ke langkah 3, jika tidak maka iterasi dilanjutkan.
3. Inisialisasi variabel *score j* dengan nilai 0 dan *weight j* dengan nilai 1.
4. Melakukan iterasi setiap *node* predecessor dari *node i*.
5. Jika *predecessor i* bertipe "Verba-Definitif" maka nilai *weight j* ditambahkan 2. Jika tidak maka lanjut ke langkah 6.
6. Jika posisi kata *predecessor i* terdapat pada indeks leksikal 0 dalam kalimat, maka skor ditambahkan 1. Jika tidak maka lanjut ke langkah 7.
7. Jika posisi kata *predecessor i* terdapat pada indeks leksikal 1 dalam kalimat, maka skor ditambahkan 0,5. Jika tidak maka lanjut ke langkah 8.
8. Menghitung nilai *score j* dengan persamaan sebagai berikut.

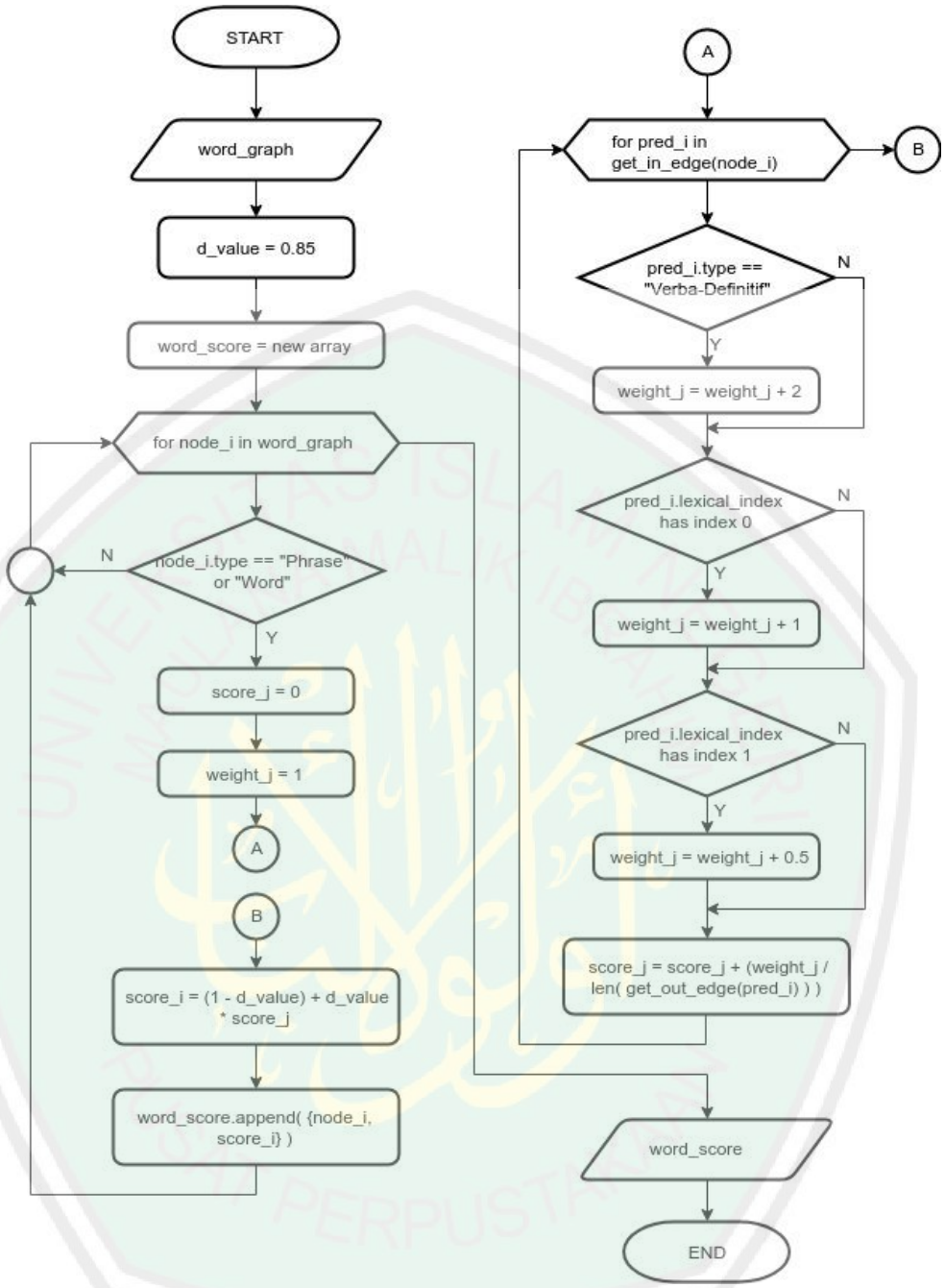
$$score\_j = score\_j + (weight\_j / len(get\_out\_edge(pred\_i))) \quad (3.2)$$

9. Jika iterasi dari *predecessor i* selesai, maka dilanjutkan dengan menghitung nilai *score i* dengan persamaan sebagai berikut.

$$score\_i = (1 - d\_value) + d\_value * score\_j \quad (3.3)$$

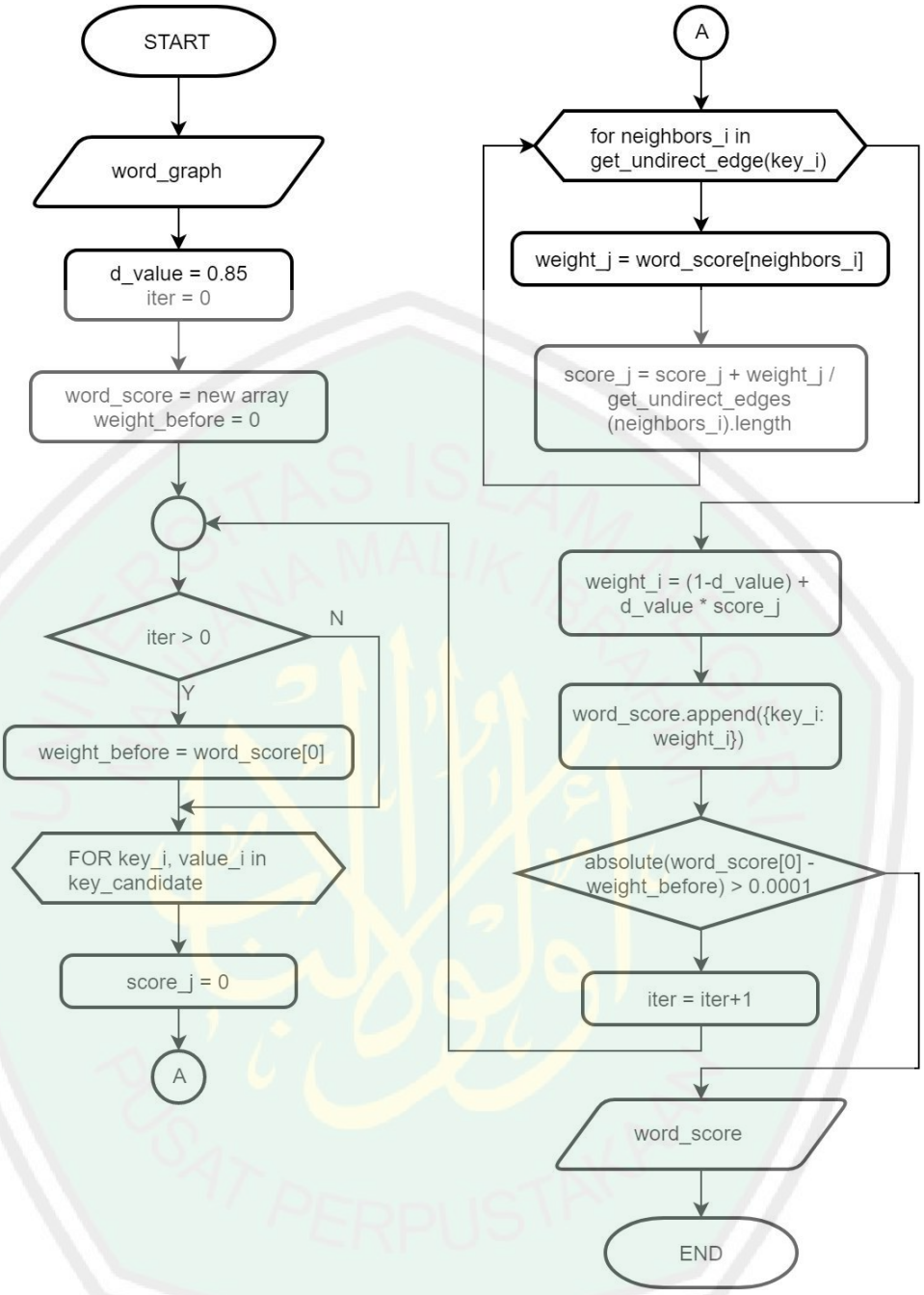
10. Menambahkan nilai *score i* ke dalam *array* word score.

Flowchart perankingan *Modified TextRank* ditunjukkan pada **Gambar 3.9**.



Gambar 3.9. *Flowchart* perankingan *Modified TextRank*

Untuk algoritma Original TextRank, persamaan yang digunakan yaitu **Persamaan 2.2.** Nilai default awal wj yaitu 1. Apabila nilai wj sudah dihitung maka nilai wj sama dengan nilai akhir wj. *Flowchart* perankingan *Original TextRank* ditunjukkan pada **Gambar 3.10**.



Gambar 3.10. *Flowchart* perankingan *Original TextRank*

3.3 Desain *Database*

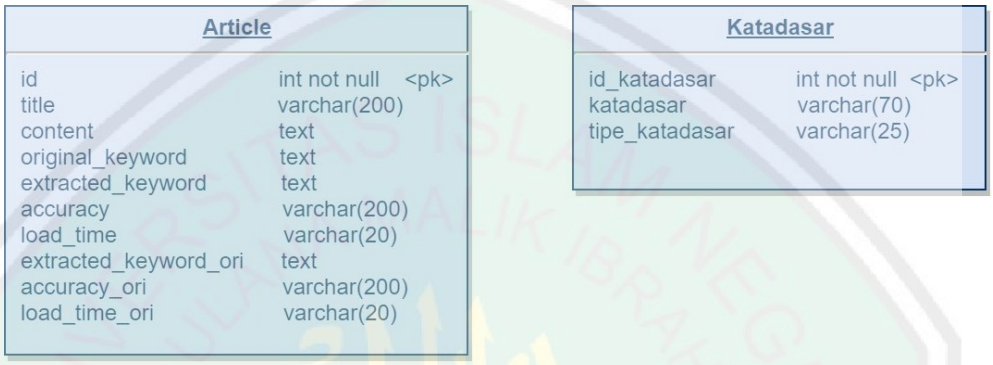
Database aplikasi *Keyword Extractor* terdiri dari dua tabel yaitu tabel *Article* dan tabel *Katadasar*. Tabel *Article* berisi daftar artikel yang tersimpan dalam sistem. Tabel ini terdiri dari 10 kolom antara lain:

1. *id,* menggunakan tipe data *integer* sebagai *primary key.*
2. *title,* bertipe data *varchar* untuk menyimpan judul artikel.
3. *content,* bertipe data *text* untuk menyimpan isi dari artikel.
4. *original_keyword,* bertipe data *text* untuk menyimpan kata kunci asli dari artikel.
5. *extracted_keyword,* bertipe data *text* untuk menyimpan kata kunci hasil ekstrasi dari sistem.
6. *accuracy,* bertipe data *varchar* untuk menyimpan akurasi dari hasil ekstrasi.
7. *load_time,* bertipe data *varchar* untuk menyimpan waktu ekstrasi.
8. *extracted_keyword_ori,* bertipe data *text* untuk menyimpan kata kunci hasil ekstrasi dari algoritma *textrank original*.
9. *accuracy_ori,* bertipe data *varchar* untuk menyimpan akurasi hasil ekstrasi dari algoritma *textrank original*.
10. *load_time_ori,* bertipe data *varchar* untuk menyimpan waktu ekstrasi dari algoritma *textrank original*.

Tabel *Katadasar* berisi kamus kata dasar yang diperoleh dari situs http://bahtera.org/. Tabel ini terdiri dari tiga kolom antara lain:

1. *id_katadasar* menggunakan tipe data *integer* sebagai *primary key*.
2. *katadasar* bertipe data *varchar* untuk menyimpan kata dasar.
3. *tipe_katadasar* bertipe data *varchar* untuk menyimpan tipe kata dasar.

Desain *database* aplikasi *Keyword Extractor* dapat dilihat pada **Gambar 3.11**.



Gambar 3.11. Desain *Database*

3.4 Desain Tampilan

Aplikasi *Keyword Extractor* dibangun berbasis web. Desain tampilan atau *User Interface* (UI) aplikasi *Keyword Extractor* terdiri dari empat halaman utama, antara lain Beranda, Buat Artikel Baru, Detail Artikel dan Hasil Ekstraksi kata kunci.

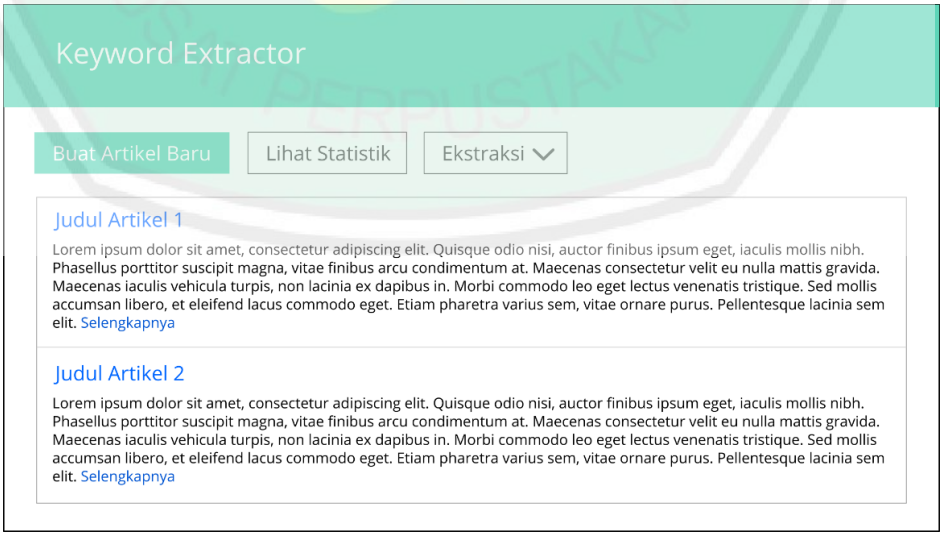
3.4.1 Beranda

Halaman beranda merupakan halaman awal aplikasi *Keyword Extractor*. Layout halaman beranda terdiri dari *header* yang berisi nama aplikasi dan *content* yang berisi daftar artikel yang tersimpan dalam *database*. Jika aplikasi belum terisi artikel, maka akan akan muncul *button* Buat Artikel Baru untuk menambahkan

artikel baru ke aplikasi. Berikut ini merupakan tampilan awal ketika belum terisi artikel.

Gambar 3.12. Tampilan beranda awal

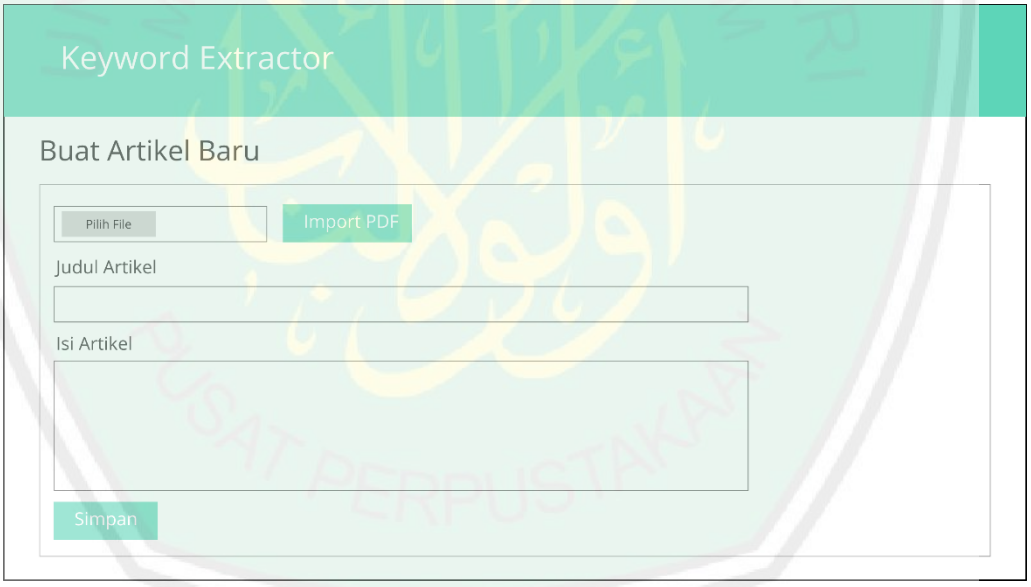
Jika tampilan halaman beranda sudah terisi artikel, maka akan tampil daftar artikel yang terdapat dalam database. Setiap *item* artikel menampilkan judul artikel dan sebagian isi artikel. Judul artikel tersebut berupa hyperlink yang mengarah ke halaman detail artikel.



Gambar 3.13. Tampilan beranda

3.4.2 Buat Artikel

Halaman ini berfungsi untuk menambahkan artikel baru ke dalam sistem. Terdapat dua opsi untuk menambahkan artikel. Pertama, menggunakan fitur Import PDF. Pengguna memilih file PDF yang berisi artikel kemudian menekan Import PDF, secara otomatis judul, isi dan kata kunci akan terisi sesuai dengan isi dari file PDF yang diimpor. Kedua, pengguna dapat mengisi manual kolom judul, isi dan kata kunci artikel. Apabila pengguna selesai mengetik judul, isi dan kata kunci, pengguna dapat menekan button simpan dan aplikasi dialihkan ke halaman beranda. Desain halaman buat artikel baru ditunjukkan pada **Gambar 3.14**.



Gambar 3.14. Tampilan Buat Artikel Baru

3.4.3 Detail Artikel

Halaman ini menampilkan judul dan isi artikel secara keseluruhan dari artikel yang dipilih di halaman beranda. Terdapat tiga opsi yang disediakan, yaitu

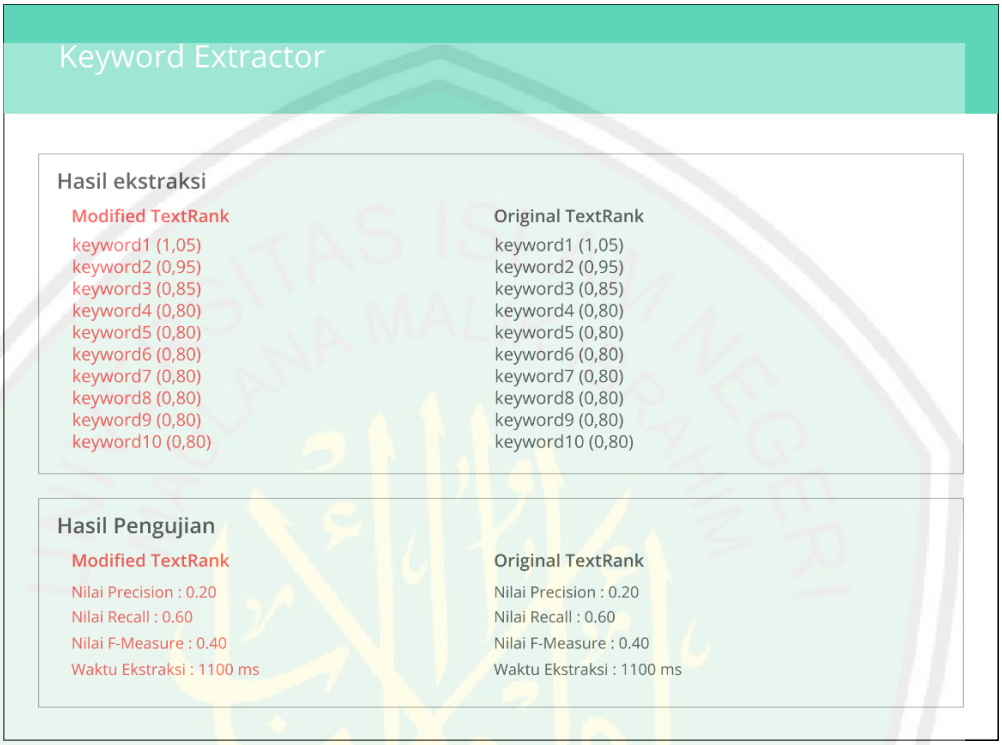
analisa kata kunci, edit artikel dan hapus artikel. Opsi analisa kata kunci berfungsi untuk mengekstrak kata kunci dalam artikel, opsi edit artikel berfungsi untuk mengubah judul atau isi artikel, dan opsi hapus artikel untuk menghapus artikel. Desain halaman detail artikel ditunjukkan pada **Gambar 3.15**.

Gambar 3.15. Tampilan Detail Artikel

3.4.4 Hasil Ekstraksi

Halaman ini menampilkan hasil ekstraksi kata kunci yang dihasilkan sistem beserta hasil pengujiannya. Hasil ekstraksi terdiri dari dua kolom. Pertama kolom kata kunci hasil ekstrasi dari algoritma yang dibangun dan kolom kedua menampilkan kata kunci hasil ekstrasi dari algoritma *textrank original*. Masing-masing kolom menampilkan sepuluh kata kunci terbaik beserta skor yang diperoleh dari hasil ekstraksi. Kata kunci diurutkan dari kata kunci yang memiliki skor yang tertinggi hingga terendah. Panel hasil pengujian juga terdiri dari dua kolom yaitu kolom pengujian hasil ekstrasi dari algoritma yang dibangun dan kolom pengujian

hasil ekstrasi dari algoritma *textrank original*. Hasil pengujian terdiri dari nilai *precision*, nilai *recall*, nilai *f-measure* dan waktu ekstraksi yang diperlukan. Desain halaman hasil ekstraksi ditunjukkan pada **Gambar 3.16**.



Gambar 3.16. Tampilan hasil ekstraksi

3.4.5 Statistik

Halaman ini menampilkan statistik hasil ekstraksi dari keseluruhan artikel yang terdiri dari rata-rata akurasi, akurasi tertinggi, rata-rata waktu ekstraksi dan total waktu ekstraksi yang diperlukan untuk seluruh artikel dari masing-masing pengujian. Di bawahnya terdapat tabel yang berisi seluruh artikel yang ada di *database*. Tabel tersebut terdiri dari enam kolom antara lain id, judul artikel, akurasi *modified textrank*, waktu akurasi *modified textrank*, akurasi *original textrank* dan

waktu akurasi *original textrank*. Desain halaman statistik ditunjukkan pada **Gambar 3.17**.

Gambar 3.17. Tampilan halaman statistik

3.5 Implementasi Sistem

Rancangan sistem yang telah dibuat diimplementasikan menggunakan bahasa pemrograman Python 3.7. Untuk pengintegrasian antara sistem dengan desain antarmuka digunakan *framework* Django 2.2 yang merupakan *framework* web berbahasa pemrograman Python. Sistem juga membutuhkan *database* untuk menyimpan data artikel dan kamus kata dasar. *Database* yang digunakan yaitu MySQL versi 5.0.

3.5.1 Import File Artikel

Artikel yang akan diinputkan ke dalam sistem berupa file PDF. Maka dari itu dibutuhkan *library* dan kode program untuk mendapatkan isi artikel dari file

PDF. *Library* yang digunakan yaitu *tika*. Output dari proses ini terdiri dari tiga teks, yaitu judul artikel, isi (abstrak-penutup) dan kata kunci artikel. Teks tersebut kemudian disimpan di *database*. Kode program dari proses import file artikel dapat dilihat pada **Gambar 3.18**.

```
import tika
tika.initVM()
from tika import parser

class ImportPdf:
    def __init__(self, url):
        parsed = parser.from_file('D:\\College\\Script C\\data\\UNIMED Articles\\'+url)
        self.page_content = parsed["content"]
    def get_title(self):
        temp_title = self.page_content[170:400].split(' ')
        title = ''
        for word in temp_title:
            if word.isupper() and len(word) > 1: title += word+' '
        return title

    def get_content(self):
        temp_abs = self.page_content[300:3000]
        temp_isi = self.page_content[1000:len(self.page_content)]
        abstrak = temp_abs[temp_abs.find("Abstrak"):temp_abs.find("Kata Kunci")]
        abstrak.replace("Abstrak —", "")
        abstrak = abstrak.replace("\n", " ")
        isi = temp_isi[temp_isi.find("PENDAHULUAN"):temp_isi.find("REFERENSI")]
        isi = isi.replace("PENDAHULUAN", "")
        content = { 'abstrak': abstrak, 'isi': isi }
        return content

    def get_keyword(self):
        temp_key = self.page_content[200:4000]
        keywords = temp_key[temp_key.find("Kata Kunci"):temp_key.find("I. PENDAHULUAN")]
        keywords = keywords.replace("Kata Kunci—", "")
        keywords = re.sub(r'[^a-z0-9 ,-]', ' ', keywords, flags = re.IGNORECASE|re.MULTILINE)
        return keywords
```

Gambar 3.18. Kode program proses *Tokenizing*

Contoh teks artikel yang akan diekstrak ditunjukkan pada **Gambar 3.19**. Contoh teks tersebut nantinya akan digunakan pada tiap tahap ekstraksi kata kunci untuk contoh implementasi.

Input Teks:

Kata kunci (keyword) merupakan kata-kata singkat yang dapat menggambarkan isi suatu artikel ataupun dokumen. Kata kunci memberikan kemudahan bagi setiap pembaca artikel untuk dapat secara cepat mengetahui inti dari artikel tersebut. Ekstraksi kata kunci adalah sebuah tahapan untuk dapat mengindentifikasi berbagai kumpulan teks pada suatu dokumen dan menemukan kata kunci yang tepat sesuai dengan topik pembahasan dari dokumen yang diolah.

Gambar 3.19. Contoh input teks

Implementasi sistem pada tahap *preprocessing* terdiri dari tiga proses, antara lain *Tokenizing, Stemming* dan *Part-of-Speech Tagging*. Sedangkan pada tahap processing terdiri tiga proses yang terdiri dari mencari kandidat kata kunci, membuat data *graph* dan perankingan.

3.5.2 *Tokenizing*

Pada proses ini paragraf artikel dipecah menjadi sekumpulan kata. Kode program dari *Tokenizing* dapat dilihat pada **Gambar 3.20.**

```
import re
    def word_token(text):
        result = re.sub(r'[^a-z0-9 .,:/\(\)-]', ' ', result, flags = re.IGNORECASE|re.MULTILINE)
        result = re.sub(r'[,:/\(\)]', ' , ', result, flags = re.IGNORECASE|re.MULTILINE)
        result = re.sub(r'[.]', ' . ', result, flags = re.IGNORECASE|re.MULTILINE)
        result = re.sub(r'( +)', ' ', result, flags = re.IGNORECASE|re.MULTILINE)
        return result.strip().split(' ')
```

Gambar 3.20. Kode program proses *Tokenizing*

Hasil dari proses *Tokenizing* dapat dilihat pada **Tabel 3.4**.

Tabel 3.4. Hasil proses *Tokenizing*.

Kata	kunci	,	keyword	,
merupakan	kata-kata	singkat	yang	dapat
menggambarkan	isi	suatu	artikel	ataupun
dokumen	.	Kata	kunci	memberikan
kemudahan	bagi	setiap	pembaca	artikel
untuk	dapat	secara	cepat	mengetahui
inti	dari	artikel	tersebut	.
Ekstraksi	kata	kunci	adalah	sebuah
tahapan	untuk	dapat	mengindentifikasi	berbagai
kumpulan	teks	pada	suatu	dokumen
dan	menemukan	kata	kunci	yang
tepat	sesuai	dengan	topik	pembahasan
dari	dokumen	yang	diolah	.

3.5.3 *Stemming*

Pada proses ini kata-kata yang didapat dari proses sebelumnya akan dihilangkan imbuhannya untuk mendapatkan kata dasarnya. Proses *stemming* pada sistem ini menggunakan *library Sastrawi* yang menerapkan algoritma *stemming* Nazief & Adriani. Untuk dokumentasi *library Sastrawi* dapat dilihat di laman (https://pypi.org/project/Sastrawi/).

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
        # Stemming with Sastrawi Library
    def stemming(text):
                factory = StemmerFactory()
                stemmer = factory.create_stemmer()
                return stemmer.stem_token(text)
```

Gambar 3.21. Kode program proses *Stemming*

Hasil dari proses *Stemming* dapat dilihat pada **Tabel 3.5**.

Tabel 3.5. Hasil proses *Stemming*.

Kata	kunci	,	keyword	,
rupa	kata	singkat	yang	dapat
gambar	isi	suatu	artikel	atau
dokumen	.	Kata	kunci	beri
mudah	bagi	tiap	baca	artikel
untuk	dapat	cara	cepat	tahu
inti	dari	artikel	sebut	.
Ekstraksi	kata	kunci	adalah	buah
tahap	untuk	dapat	identifikasi	bagai
kumpul	teks	pada	suatu	dokumen
dan	temu	kata	kunci	yang
tepat	sesuai	dengan	topik	bahas
dari	dokumen	yang	olah	.

3.5.4 *Part-of-Speech Tagging*

Part-of-Speech Tagging merupakan proses melabeli kata dengan tipe katanya masing-masing. Tahapan awal yaitu dengan mencocokkan kata input dengan kata yang ada di tabel Katadasar. Kemudian tiap kata akan diubah ke tipe data dict. Dalam satu dict terdiri dari tiga variabel, antara lain:

1. *type*, digunakan untuk menyimpan tipe kata.
2. *origin*, digunakan untuk menyimpan kata awal.
3. *stem*, digunakan untuk menyimpan kata dasar.

Kode program dari *Part-of-Speech Tagging* dapat dilihat pada **Gambar 3.22.**

```
from .models import Katadasar
class WordTagging:
    def __init__(self):
        self.word_dict = {}
        self.add_root_word()
```

```
    def add_root_word(self):
        word_root = Katadasar.objects.values_list('katadasar', flat=True)
        word_type = Katadasar.objects.values_list('tipe_katadasar', flat=True)
        self.word_dict = dict(zip(word_root, word_type))
    def identify_word(self, stemmed_word, origin_word ):
        word_type = {'tipe_katadasar': 'Lain-lain'}
        if stemmed_word in self.word_dict:
            if origin_word.lower() == 'adalah' or origin_word.lower() == 'merupakan':
                return 'Verba-Definitif'
            elif stemmed_word == origin_word.lower():
                return self.word_dict[stemmed_word]
            elif origin_word.lower().find('me') == 0 or origin_word.lower().find('di') == 0:
                return 'Verba'
            elif origin_word.lower().find('pe') == 0 and re.match(r'^(.*)(an|nya)$', origin_word.lower()) or
origin_word.lower().find('ke') == 0:
                return 'Nomina'
            elif origin_word.lower().find('ter') == 0 or origin_word.lower().find('se') == 0:
                return 'Preposisi'
            elif origin_word.lower().find('ber') == 0:
                return 'Adjektiva'
            else: return self.word_dict[stemmed_word]
        elif origin_word.lower() == 'dll' or origin_word.lower() == 'dsb' or origin_word.lower() == 'etc':
            return 'Konjungsi'
        elif origin_word.lower() == ',': return 'Separator'
        elif origin_word.lower() == '.': return 'Titik'
        elif re.match(r'[0-9]', origin_word.lower()): return 'Numeralia'
        else: return word_type['tipe_katadasar']
    def identify_text(self, stemmed_sentence, origin_sentence):
        type_list = []
        for i, stemmed_word in enumerate(stemmed_sentence):
            type_temp = self.identify_word(stemmed_word, origin_sentence[i])
            type_list.append({'type': type_temp, 'color':self.word_color[type_temp]})
        return type_list
    def make_word_dict(self):
        temp_word_data = []
        for j, word in enumerate(self.origin_list):
            temp_word_data.append({
                'origin': self.origin_list[j], 'stem': self.stemmed_list[j],  'type': self.word_type_list[j]['type'], })
        return temp_word_data
```

Gambar 3.22. Kode program proses *Part-of-Speech Tagging*

Hasil dari proses *Part-of-Speech Tagging* dapat dilihat pada **Gambar 3.23**.

[{'origin': 'kata', 'stem': 'kata', 'type': 'Nomina'}, {'origin': 'kunci', 'stem': 'kunci', 'type': 'Nomina'}, {'origin': ',', 'stem': ',', 'type': 'Separator'}, {'origin': 'keyword', 'stem': 'keyword', 'type': 'Lain-lain'}, {'origin': ',', 'stem': ',', 'type': 'Separator'}, {'origin': 'merupakan', 'stem': 'rupa', 'type': 'Verba-Definitif'}, {'origin': 'kata-kata', 'stem': 'kata', 'type': 'Nomina'}, {'origin': 'singkat', 'stem': 'singkat', 'type': 'Adjektiva'}, {'origin': 'yang', 'stem': 'yang', 'type': 'Konjungsi'}, {'origin': 'dapat', 'stem': 'dapat', 'type': 'Adverbia'}, {'origin': 'menggambarkan', 'stem': 'gambar', 'type': 'Verba'}, {'origin': 'isi', 'stem': 'isi', 'type': 'Nomina'}, {'origin': 'suatu', 'stem': 'suatu', 'type': 'Numeralia'}, {'origin': 'artikel', 'stem': 'artikel', 'type': 'Nomina'}, {'origin': 'ataupun', 'stem': 'atau', 'type': 'Konjungsi'}, {'origin': 'dokumen', 'stem': 'dokumen', 'type': 'Nomina'}, {'origin': '.', 'stem': '.', 'type': 'Titik'}, {'origin': 'kata', 'stem': 'kata', 'type': 'Nomina'}, {'origin': 'kunci', 'stem': 'kunci', 'type': 'Nomina'}, {'origin': 'memberikan', 'stem': 'beri', 'type': 'Verba'}, {'origin': 'kemudahan', 'stem': 'mudah', 'type': 'Nomina'}, {'origin': 'bagi', 'stem': 'bagi', 'type': 'Preposisi'}, {'origin': 'setiap', 'stem': 'tiap', 'type': 'Preposisi'}, {'origin': 'pembaca', 'stem': 'baca', 'type': 'Verba'}, {'origin': 'artikel', 'stem': 'artikel', 'type': 'Nomina'}, {'origin': 'untuk', 'stem': 'untuk', 'type': 'Preposisi'}, {'origin': 'dapat', 'stem': 'dapat', 'type': 'Adverbia'}, {'origin': 'secara', 'stem': 'cara', 'type': 'Preposisi'}, {'origin': 'cepat', 'stem': 'cepat', 'type': 'Adjektiva'}, {'origin': 'mengetahui', 'stem': 'tahu', 'type': 'Verba'}, {'origin': 'inti', 'stem': 'inti', 'type': 'Nomina'}, {'origin': 'dari', 'stem': 'dari', 'type': 'Preposisi'}, {'origin': 'artikel', 'stem': 'artikel', 'type': 'Nomina'}, {'origin': 'tersebut', 'stem': 'sebut', 'type': 'Preposisi'}, {'origin': '.', 'stem': '.', 'type': 'Titik'}, {'origin': 'ekstraksi', 'stem': 'ekstraksi', 'type': 'Nomina'}, {'origin': 'kata', 'stem': 'kata', 'type': 'Nomina'}, {'origin': 'kunci', 'stem': 'kunci', 'type': 'Nomina'}, {'origin': 'adalah', 'stem': 'adalah', 'type': 'Verba-Definitif'}, {'origin': 'sebuah', 'stem': 'buah', 'type': 'Preposisi'}, {'origin': 'tahapan', 'stem': 'tahap', 'type': 'Nomina'}, {'origin': 'untuk', 'stem': 'untuk', 'type': 'Preposisi'}, {'origin': 'dapat', 'stem': 'dapat', 'type': 'Adverbia'}, {'origin': 'mengindentifikasi', 'stem': 'mengindentifikasi', 'type': 'Lain-lain'}, {'origin': 'berbagai', 'stem': 'bagai', 'type': 'Adjektiva'}, {'origin': 'kumpulan', 'stem': 'kumpul', 'type': 'Verba'}, {'origin': 'teks', 'stem': 'teks', 'type': 'Nomina'}, {'origin': 'pada', 'stem': 'pada', 'type': 'Preposisi'}, {'origin': 'suatu', 'stem': 'suatu', 'type': 'Numeralia'}, {'origin': 'dokumen', 'stem': 'dokumen', 'type': 'Nomina'}, {'origin': 'dan', 'stem': 'dan', 'type': 'Konjungsi'}, {'origin': 'menemukan', 'stem': 'temu', 'type': 'Verba'}, {'origin': 'kata', 'stem': 'kata', 'type': 'Nomina'}, {'origin': 'kunci', 'stem': 'kunci', 'type': 'Nomina'}, {'origin': 'yang', 'stem': 'yang', 'type': 'Konjungsi'}, {'origin': 'tepat', 'stem': 'tepat', 'type': 'Adjektiva'}, {'origin': 'sesuai', 'stem': 'sesuai', 'type': 'Lain-lain'}, {'origin': 'dengan', 'stem': 'dengan', 'type': 'Preposisi'}, {'origin': 'topik', 'stem': 'topik', 'type': 'Nomina'}, {'origin': 'pembahasan', 'stem': 'bahas', 'type': 'Nomina'}, {'origin': 'dari', 'stem': 'dari', 'type': 'Preposisi'}, {'origin': 'dokumen', 'stem': 'dokumen', 'type': 'Nomina'}, {'origin': 'yang', 'stem': 'yang', 'type': 'Konjungsi'}, {'origin': 'diolah', 'stem': 'olah', 'type': 'Verba'}, {'origin': '.', 'stem': '.', 'type': 'Titik'}]

Gambar 3.23. Hasil proses *Part-of-Speech Tagging*

3.5.5 Mencari kandidat kata kunci/frasa kunci

Pada tahapan ini data *dict* yang diperoleh dari tahapan *POS Tagging* akan dibentuk kandidat kata kunci, frasa kunci dan verba. Kode program dari tahapan ini dapat dilihat pada **Gambar 3.24.**

```
import networkx as nx


# Processing steps
class Processing:
    def __init__(self):
        self.word_phrase_list = {}
        self.word_phrase_list_array = []
        self.word_graph = nx.DiGraph()
        self.word_graph_undirect = nx.Graph()
        self.word_score_list = {}

    def add_key_candidate(self, keys, key_type, sentence_id, lexical_index):
        if keys in self.word_phrase_list:
            has_sentence = self.word_phrase_list[keys]['has_sentence']
            has_sentence.append(sentence_id)
            temp_lexical_index =  self.word_phrase_list[keys]['lexical_index']
            temp_lexical_index.append(lexical_index)
            self.word_phrase_list.update({
                keys:{'type':key_type, 'has_sentence': has_sentence, 'lexical_index': temp_lexical_index}
            })
        else:
            self.word_phrase_list.update({
                keys: {'type':key_type, 'has_sentence': [sentence_id], 'lexical_index': [lexical_index]}
            })
        self.word_phrase_list_array.append(keys)

    def process_word(self, sentence, basic=False):
        skip_word = 0
        lexical_index = 0
        sentence_id = 0
        for i, word in enumerate(sentence):
            if skip_word == 0:
                if word['type'] == 'Nomina' or word['type'] == 'Lain-lain':
                    keysid1 = word['origin']
                    if i < len(sentence)-1:
                        if sentence[i+1]['type'] == 'Nomina' or sentence[i+1]['type'] == 'Lain-lain':
                            if word['type'] != 'Lain-lain' and sentence[i+1]['type'] == 'Lain-lain': continue
                            keysid2 = sentence[i+1]['origin']
                            keys2 = keysid1 + ' ' + keysid2
```

```
                    if i < len(sentence)-2:
                        if sentence[i+2]['type'] == 'Nomina' or sentence[i+2]['type'] == 'Lain-lain':
                            keysid3 = sentence[i+2]['origin']
                        keys3 = keys2 + ' ' + keysid3
                        self.add_key_candidate(keys3, 'Phrase', sentence_id, lexical_index)
                            skip_word = 2
                    else:  self.add_key_candidate(keys2, 'Phrase', sentence_id, lexical_index)
                        skip_word = 1
                else:
                    self.add_key_candidate(keys2, 'Phrase', sentence_id, lexical_index)
                    skip_word = 1
            else:  self.add_key_candidate(keysid1, 'Word', sentence_id, lexical_index)
        else: self.add_key_candidate(keysid1, 'Word', sentence_id, lexical_index)


    elif word['type'] == 'Verba':
        if not basic:
            keys = word['origin']
            self.add_key_candidate(keys, 'Verba', sentence_id, lexical_index)
        lexical_index += 1

    elif word['type'] == 'Verba-Definitif':
        if not basic:
            keys = word['origin']
            self.add_key_candidate(keys, 'Verba-Definitif', sentence_id, lexical_index)
        lexical_index += 1
    elif word['type'] == 'Titik':
        sentence_id += 1
        lexical_index = 0
else:
    skip_word-=1
    continue
```

Gambar 3.24. Kode program mencari kandidat kata kunci

Data kandidat kata kunci bertipe data *dict*. Kandidat kata kunci dan verba digunakan sebagai *key*, dan *value* terdiri dari tiga variabel, antara lain:

1. *type*, digunakan untuk menyimpan tipe kata.
2. *has_sentence*, digunakan untuk menyimpan indeks kalimat dari kata.

3. *lexical_index*, digunakan untuk menyimpan kata indeks leksikal dari kata.

Hasil dari proses pencarian kandidat kata kunci dapat dilihat pada **Tabel 3.6**.

Tabel 3.6. Hasil proses pencarian kandidat kata kunci**.**

Kata kunci & verba	Tipe	Indeks kalimat	Indeks leksikal
kata kunci	Phrase	0, 1, 2	0, 0, 4
keyword	Word	0	0
merupakan	Verba-Definitif	0	0
kata-kata	Word	0	1
Isi	Word	0	2
artikel	Word	0, 1, 1	2, 2, 3
dokumen	Word	0, 2, 2	2, 3, 4
kemudahan	Word	1	1
inti	Word	1	3
ekstraksi kata kunci	Phrase	2	0
tahapan	Word	2	1
teks	Word	2	3
sesuai	Word	2	4
topik pembahasan	Phrase	2	4
menggambarkan	Verba	0	1
memberikan	Verba	1	0
pembaca	Verba	1	1
mengetahui	Verba	1	2
adalah	Verba-Definitif	2	0
mengidentifikasi	Verba	2	1
kumpulan	Verba	2	2
menemukan	Verba	2	3
diolah	Verba	2	4

3.5.6 Membuat data graph

Dalam sistem ini terdapat dua opsi pembentukan data *graph*, yang pertama yaitu *graph* berarah yang dirancang oleh penulis, yang kedua merupakan *graph* tak berarah yang dirancang berdasarkan penelitian yang dilakukan oleh Ramadhiana

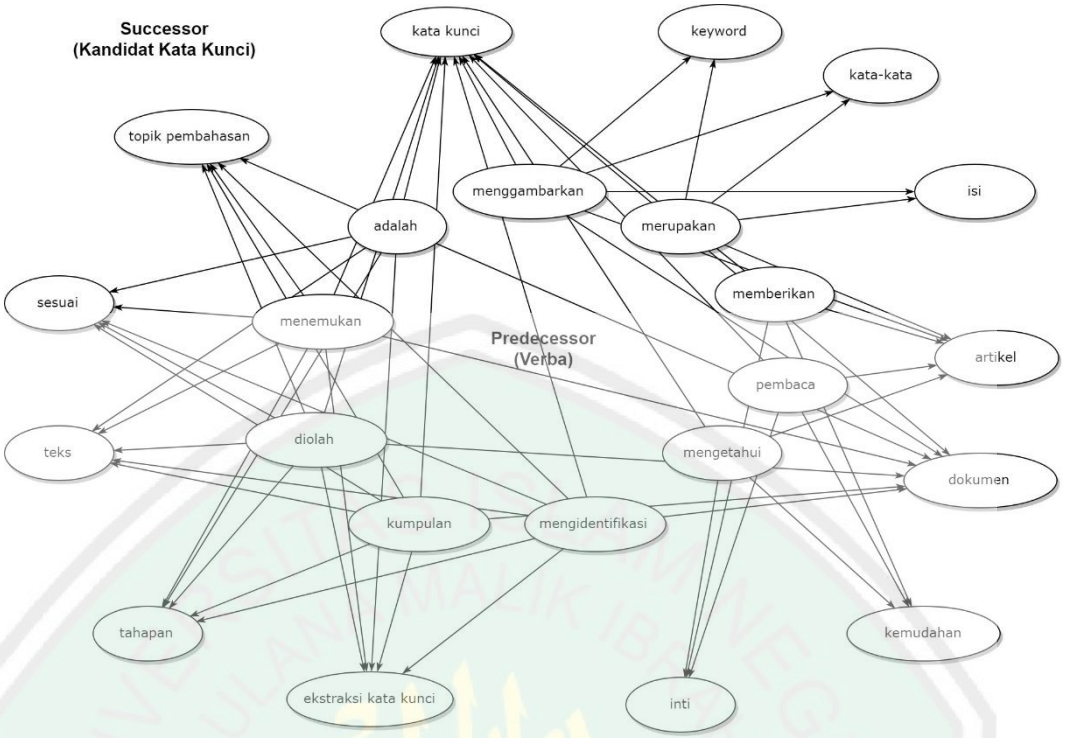
(2017). Graph yang kedua dibangun untuk algoritma TextRank asli. Proses pembentukan *graph* menggunakan *library networkx* (). Kode program dari proses pembentukan data *graph* berarah ditunjukkan pada **Gambar 3.25**.

```
def make_word_graph(self):
    self.word_graph.add_nodes_from(self.word_phrase_list)

def make_graph_edges(self):
    for key_i, value_i in self.word_phrase_list.items():
        if value_i['type'] == 'Verba' or value_i['type'] == 'Verba-Definitif':
            for k, sentence_id in enumerate(value_i['has_sentence']):
                for key_j, value_j in self.word_phrase_list.items():
                    if sentence_id in value_j['has_sentence'] and (value_j['type'] == 'Phrase' or value_j['type'] ==
'Word'):
                        self.word_graph.add_edge(key_i, key_j)
```

Gambar 3.25. Kode program pembentukan data *graph* berarah

Proses pembentukan *graph* berarah mengasilkan *graph* yang lebih kompleks dari graph tak berarah. Hal ini dikarenakan terdapat *node* Verba sebagai *predecessor* dari kandidat kata kunci. Verba terhubung dengan kandidat kata kunci yang masih satu kalimat dengan verba tersebut. Hasil dari proses pembentukan *graph* berarah dapat dilihat pada **Gambar 3.26**.



Gambar 3.26. Hasil proses pembentukan *graph* berarah

Kode program proses pembentukan data *graph* tak berarah ditunjukkan pada **Gambar 3.27**.

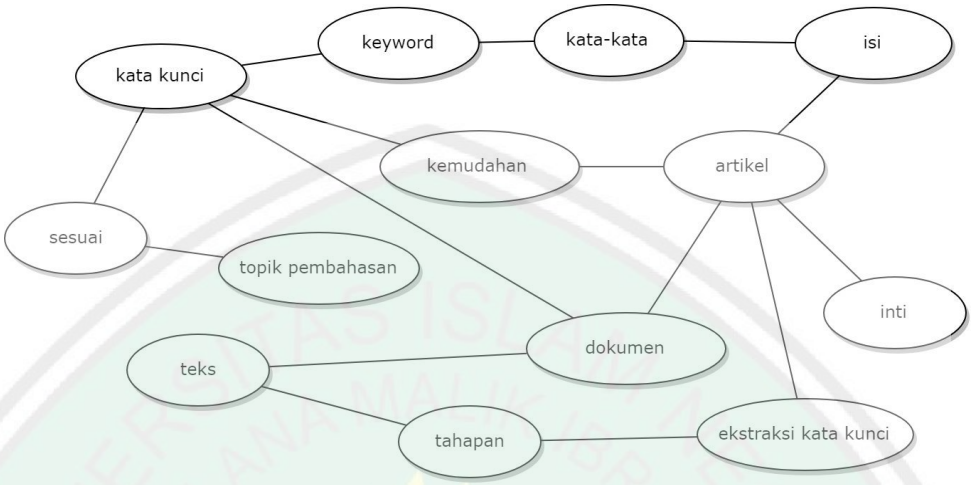
```
def make_undirect_graph(self):
    self.word_graph_undirect.add_nodes_from(self.word_phrase_list)

def make_undirect_edges(self):
    for word_index in range(len(self.word_phrase_list_array)-2):
        self.word_graph_undirect.add_edge(
            self.word_phrase_list_array[word_index],
            self.word_phrase_list_array[word_index+1]
        )
```

Gambar 3.27. Kode program pembentukan data *graph* tak berarah

Proses pembentukan *graph* tak berarah menghasilkan *graph* yang lebih sederhana karena hanya terdapat *node* kandidat kata kunci. *Node* yang terhubung

merupakan kata yang muncul bersamaan dengan batas dua kata. Hasil *graph* tak berarah dapat dilihat pada **Gambar 3.28**.



Gambar 3.28. Hasil proses pembentukan *graph* tak berarah

3.5.7 Perankingan

Proses perankingan ini juga memiliki dua opsi berdasarkan proses pembentukan data graph sebelumnya. Untuk graph berarah menggunakan **Persamaan 3.1**, sedangkan untuk graph tak berarah menggunakan **Persamaan 2.2**. Untuk membedakan antara dua proses tersebut, penulis memberi masing-masing algoritma. Untuk algoritma yang penulis rancang, diberi nama *Modified TextRank*, sedangkan untuk algoritma TextRank asli diberi nama *Original TextRank*. Algoritma *Original TextRank* yang diimplementasikan pada sistem ini berdasarkan penelitian yang dilakukan Ramadhiana (2017). Kode program dari proses perankingan menggunakan *Modified TextRank* ditunjukkan pada **Gambar 3.29**.

```
def calc_modif_textrank(self, tr_iter=0):
        d_value = 0.85
    for key_i, value_i in self.word_phrase_list.items():
        if value_i['type'] == 'Phrase' or value_i['type'] == 'Word':
            score_j = 0
            for pred_i in self.get_in_edges(key_i):
                weight_j = 1
                if self.word_phrase_list[pred_i]['type'] == 'Verba-Definitif': weight_j += 2
                if 0 in self.word_phrase_list[pred_i]['lexical_index']: weight_j += 1
                if 1 in self.word_phrase_list[pred_i]['lexical_index']: weight_j += 0.5
                score_j += weight_j / len(self.get_out_edges(pred_i))
            score_i = (1-d_value) + d_value * score_j
            self.word_score_list.update({key_i: score_i})
```

Gambar 3.29. Kode program perankingan *Modified TextRank*

Sedangkan untuk kode program dari perankingan menggunakan *Original TextRank* ditunjukkan pada **Gambar 3.30**.

```
def calc_origin_textrank(self, tr_iter=0):
        d_value = 0.85
        weight_before = 0.0
    if tr_iter > 0: weight_before = self.get_word_score_list()[0][1]
    for key_i, value_i in self.word_phrase_list.items():
        score_j = 0, weight_j = 1
        for neighbors_i in self.get_undirect_edges(key_i):
            if tr_iter > 0: weight_j = self.word_score_list[neighbors_i]
            score_j += weight_j / len(self.get_undirect_edges(neighbors_i))
        weight_i = (1-d_value) + d_value * score_j
        self.word_score_list.update({key_i: weight_i})
    if abs(self.get_word_score_list()[0][1] - weight_before) > 0.0001:
        self.calc_basic_textrank(tr_iter+1)
```

Gambar 3.30. Kode program perankingan *Original TextRank*

Implementasi algoritma *Modified TextRank* dan *Original TextRank* ke kode program menghasilkan proses perhitungan yang otomatis. Hasil perhitungan

dari sitem menghasilkan nilai yang sama dengan nilai hasil perhitungan manual. Hasil dari perhitungan skor *Modified TextRank* dapat dilihat pada **Tabel 3.7**.

Tabel 3.7. Hasil perhitungan skor *Modified TextRank***.**

No	Kata kunci	Jumlah predecessor	Skor kata
1	kata kunci	10	2.9175
2	dokumen	7	1,9613
3	artikel	5	1,8854
4	ekstraksi kata kunci	5	1,1821
5	tahapan	5	1,1821
6	teks	5	1,1821
7	sesuai	5	1,1821
8	topik pembahasan	5	1,1821
9	kemudahan	3	1,1062
10	inti	3	1,1062
11	keyword	2	0,9291
12	kata-kata	2	0,9291
13	isi	2	0,9291

Pada proses perhitungan skor *Modified TextRank*, nilai S(Vj) ditentukan berdasarkan tipe kata dan indeks leksikal dari *predecessor* yang telah dijelaskan di **Tabel 3.3**. Rincian dari perhitungan skor *Modified TextRank* yakni sebagai berikut:

1. Skor kata "kata kunci":

$$S(Vi) = (1-d) + d \times \sum_{j \in In(Vi)} \frac{1}{|Out(Vj)|} S(Vj)$$

$$= (1-0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6} + \frac{(1+1)}{4} + \frac{(1+0{,}5)}{4} + \frac{1}{4} + \frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 2.9175$$

2. Skor kata "dokumen":

$$S(Vi) = (1-0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6} + \frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}9613$$

3. Skor kata "artikel":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6} + \frac{(1+1)}{4} + \frac{(1+0{,}5)}{4} + \frac{1}{4}\right) = 1{,}8854$$

4. Skor kata "ekstraksi kata kunci":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}1821$$

5. Skor kata "tahapan":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}1821$$

6. Skor kata "teks":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}1821$$

7. Skor kata "sesuai":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}1821$$

8. Skor kata "topik pembahasan":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{7} + \frac{(1+0{,}5)}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7}\right) = 1{,}1821$$

9. Skor kata "kemudahan":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+1)}{4} + \frac{(1+0{,}5)}{4} + \frac{1}{4}\right) = 1{,}1062$$

10. Skor kata "inti":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+1)}{4} + \frac{(1+0{,}5)}{4} + \frac{1}{4}\right) = 1{,}1062$$

11. Skor kata "keyword":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6}\right) = 0{,}9291$$

12. Skor kata "kata-kata":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6}\right) = 0{,}9291$$

13. Skor kata "isi":

$$S(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{(1+2+1)}{6} + \frac{(1+0{,}5)}{6}\right) = 0{,}9291$$

Untuk hasil perhitungan skor dari *Original TextRank* dapat dilihat pada **Tabel 3.8**.

Tabel 3.8. Hasil perhitungan skor *Original TextRank*.

No	Kata kunci	Jumlah relasi	Skor kata
1	kata kunci	4	1.6388
2	dokumen	3	1.2093
3	artikel	5	2.0061
4	ekstraksi kata kunci	2	0.8690
5	tahapan	2	0.8893
6	teks	2	0.8706
7	sesuai	2	0.9796
8	topik pembahasan	1	0.5663
9	kemudahan	2	0.8392
10	inti	1	0.4910
11	keyword	2	0.8778
12	kata-kata	2	0.8931
13	isi	2	0.8706

Nilai default awal *wj* yaitu 1. Apabila nilai *wj* sudah dihitung maka nilai *wj* sama dengan nilai akhir *wj*. Rincian perhitungan skor pada *Original TextRank* yakni sebagai berikut:

1. Skor kata "kata kunci":

$$WS(Vi) = (1 - d) + d \times \sum_{Vj \in In(Vi)} \frac{wji}{\sum_{Vk \in Out(Vj)} wjk} WS(Vj)$$

$$= (1 - 0{,}85) + 0{,}85 \times \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{2}\right) = 1{,}7083$$

2. Skor kata "keyword":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{1{,}7083}{4} + \frac{1}{2}\right) = 0{,}9380$$

3. Skor kata "kata-kata":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{0{,}9380}{2} + \frac{1}{2}\right) = 0{,}9736$$

4. Skor kata "isi":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{0{,}9736}{2} + \frac{1}{5}\right) = 0{,}7338$$

5. Skor kata "artikel":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{0{,}7338}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{1} + \frac{1}{2}\right) = 2.4452$$

6. Skor kata "dokumen":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{2.4452}{5} + \frac{1{,}7083}{4} + \frac{1}{2}\right) = 1.3537$$

7. Skor kata "kemudahan":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{2.4452}{5} + \frac{1{,}7083}{4}\right) = 0.9287$$

8. Skor kata "inti":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{2.4452}{5}\right) = 0.5656$$

9. Skor kata "ekstraksi kata kunci":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{2.4452}{5} + \frac{1}{2}\right) = 0.9906$$

10. Skor kata "tahapan":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{0.9906}{2} + \frac{1}{2}\right) = 0.9960$$

11. Skor kata "teks":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{0.9960}{2} + \frac{1.3537}{3}\right) = 0.9568$$

12. Skor kata "sesuai":

$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{1{,}7083}{4} + \frac{1}{1}\right) = 1.3630$$

13. Skor kata "topik pembahasan":

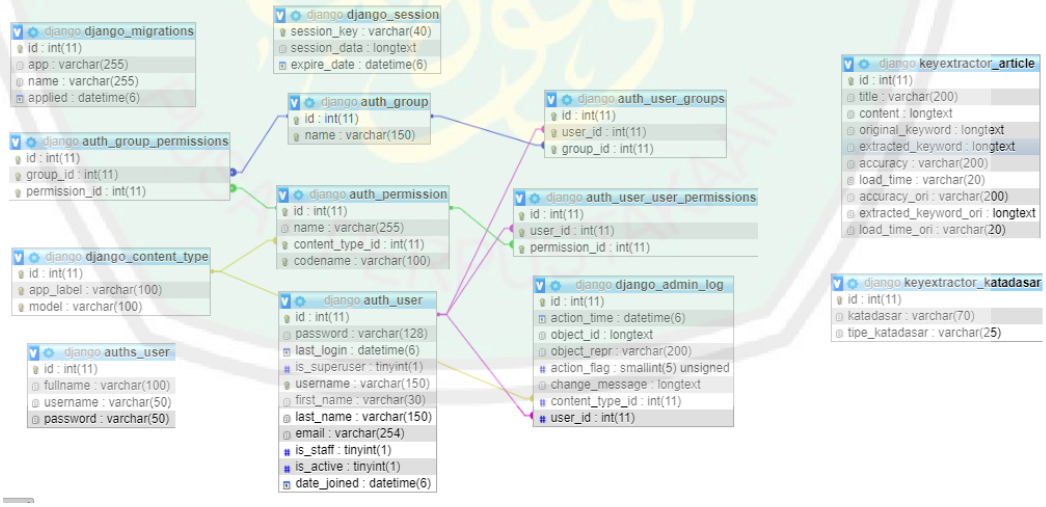
$$WS(Vi) = (1 - 0{,}85) + 0{,}85 \times \left(\frac{1.3630}{5}\right) = 0.7292$$

Proses perhitungan skor pada *Original TextRank* dilakukan secara berulang-ulang hingga didapat nilai *error rate* 0,0001.

Dari kedua proses perhitungan skor di atas, hanya akan dipilih 10 kata kunci dengan nilai skor tertinggi sebagai output dari sistem ekstraksi kata kunci.

3.6 Implementasi Database

Langkah awal proses pengimplementasian desain *database* yaitu membuat model di *framework Django* dengan cara menulis kode program yang berisi objek tabel beserta atributnya. Kemudian proses implementasi ke *database* MySQL dilakukan secara otomatis oleh *framework Django* dengan perintah *makemigrations* dan *migrate*. Hasil dari implementasi *database* pada MySQL dapat dilihat pada **Gambar 3.31**.



Gambar 3.31. ERD Aplikasi Keyword Extractor

Tabel yang digunakan pada aplikasi *Keyword Extractor* hanya dua tabel, yakni tabel *keyextractor_article* dan *keyextractor_katadasar*. Selain kedua tabel

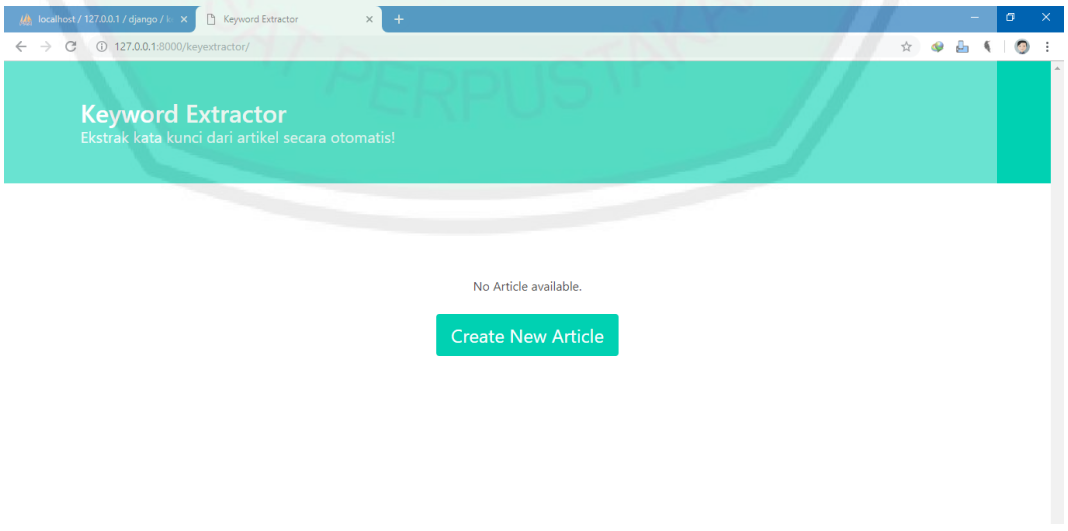
tersebut merupakan tabel bawaan dari *framework Django* dan tidak digunakan pada aplikasi *Keyword Extractor*.

3.7 Implementasi Desain Antarmuka

Aplikasi yang dibangun diberi nama *Keyword Extractor*. Aplikasi ini dibangun berbasis web menggunakan bahasa pemrograman *python* dan *framework Django*. Pembuatan antarmuka sistem menggunakan bahasa HTML5, JavaScript dan JQuery. Kemudian untuk memperindah tampilan antarmuka web digunakan *framework CSS* Bulma. Aplikasi ini terdiri dari empat halaman utama, yakni Beranda, Buat/Edit Artikel, Detail & Analisis Artikel, dan Statistik.

3.7.1 Halaman Beranda

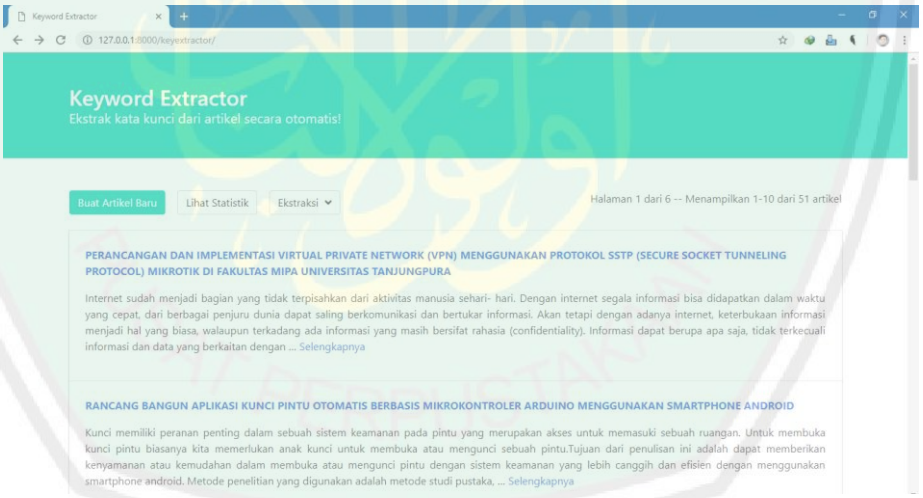
Halaman beranda merupakan tampilan awal dari aplikasi *Keyword Extractor*. Halaman ini menampilkan daftar artikel yang tersimpan di *database*. **Gambar 3.32** menunjukkan halaman beranda ketika belum tersimpan artikel.



Gambar 3.32. Halaman beranda awal ketika belum tersimpan artikel.

Pada halaman beranda terdapat *button* Buat Artikel Baru, Lihat Statistik dan Ekstraksi. *Button* Buat Artikel Baru berfungsi untuk navigasi ke halaman Buat Artikel, *button* Lihat Statistik berfungsi untuk navigasi ke halaman Statistik, dan *button* Ekstraksi merupakan *dropdown* yang terdiri dari 3 submenu, antara lain:

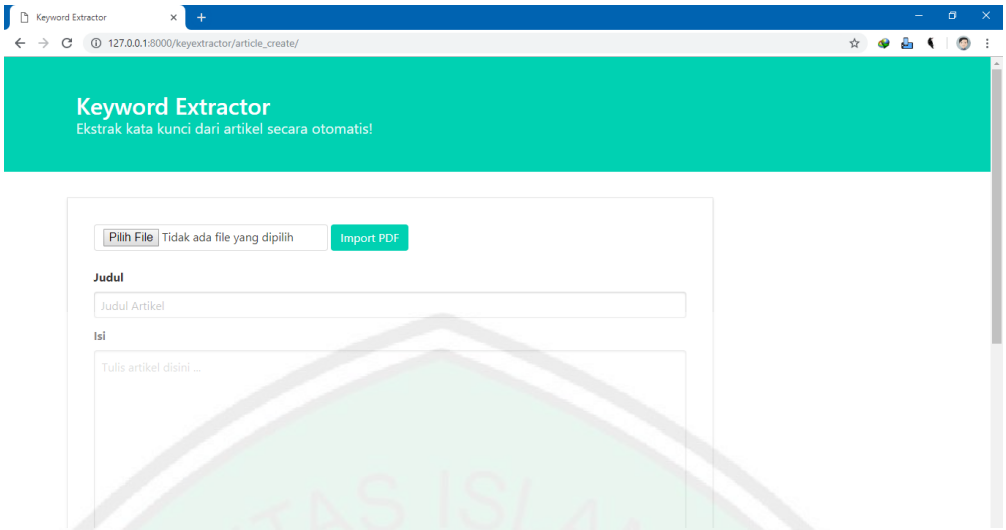
1. Ekstrak menggunakan *Modified TextRank*, berfungsi untuk mengekstrak kata kunci dari semua artikel menggunakan *Modified TextRank*.
2. Ekstrak menggunakan *Original TextRank*, berfungsi untuk mengekstrak kata kunci dari semua artikel menggunakan *Original TextRank*.
3. Reset semua ekstraksi, berfungsi untuk menghapus semua ekstraksi yang telah dilakukan.



Gambar 3.33. Halaman beranda

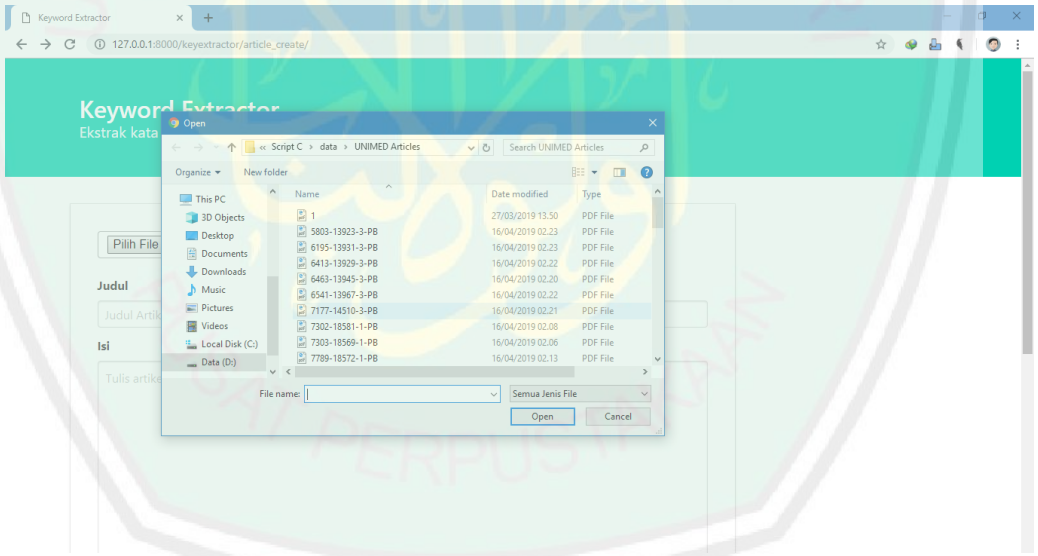
3.7.2 Halaman Buat/Edit Artikel

Halaman ini memiliki dua fungsi, yakni untuk membuat artikel baru dan mengedit artikel. Terdapat empat form input di dalamnya, antara lain import file pdf, judul artikel, isi artikel dan kata kunci artikel.



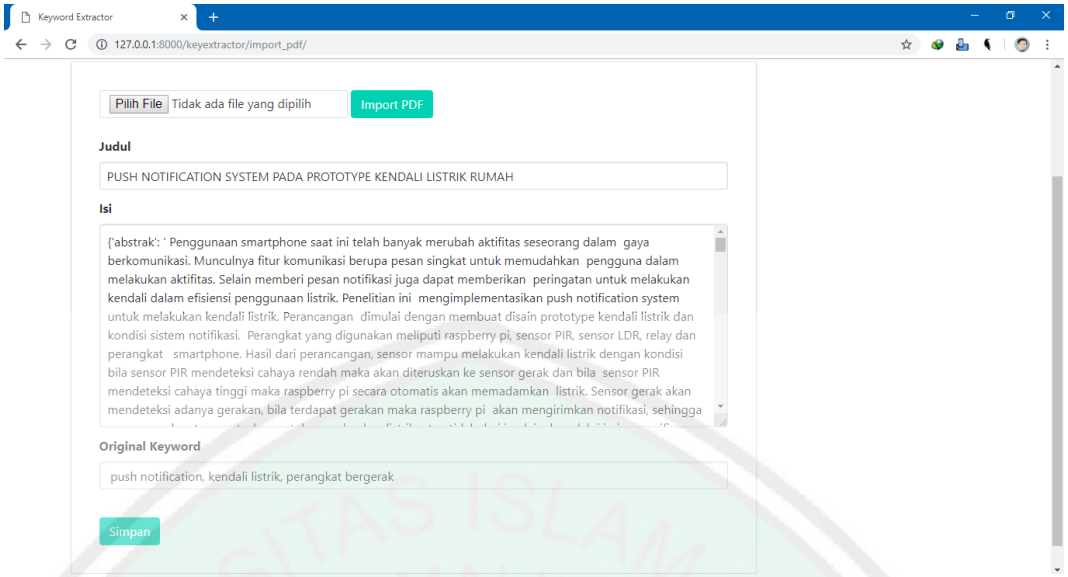
Gambar 3.34. Halaman buat/edit artikel

Pengguna dapat menulis manual artikel atau menambahkan artikel dengan cara memilih file pdf yang berisi artikel kemudian menekan *button* Import PDF.



Gambar 3.35. Memilih file pdf

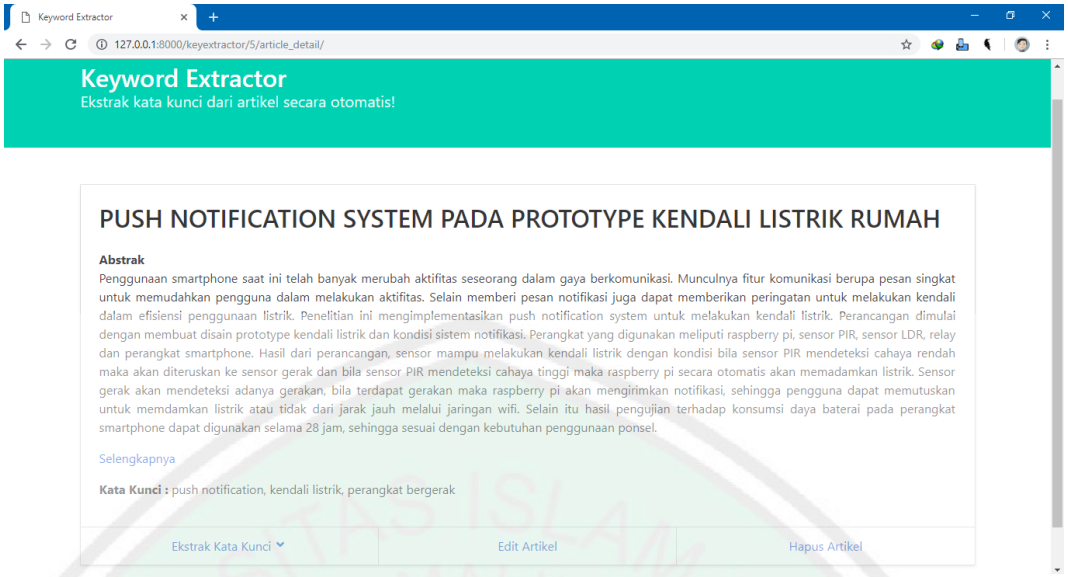
Setelah proses import file pdf selesai, masing-masing form input akan terisi artikel sesuai dengan file yang di-import. Pengguna dapat mengedit artikel atau langsung menyimpannya dengan cara menekan *button* Simpan.



Gambar 3.36. Halaman buat artikel ketika sudah terisi

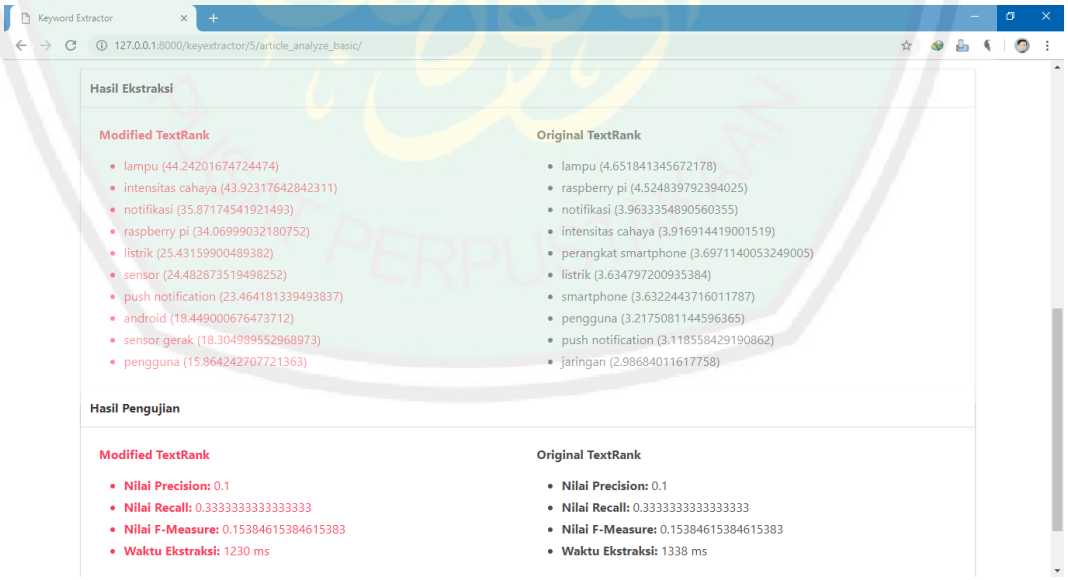
3.7.3 Halaman Detail & Analisis Artikel

Halaman ini menampilkan isi artikel secara keseluruhan, kata kunci dan menu. Menu ekstrak kata kunci berfungsi untuk mengekstrak kata kunci dari artikel. Terdapat dua pilihan ekstraksi, yakni ekstrak menggunakan *Modified TextRank* dan ekstrak menggunakan *Original TextRank*. Menu edit artikel berfungsi untuk navigasi ke halaman edit artikel. Menu hapus artikel berfungsi untuk menghapus artikel dari *database*. Halaman detail artikel ditunjukkan pada **Gambar 3.37**.



Gambar 3.37. Halaman detail artikel

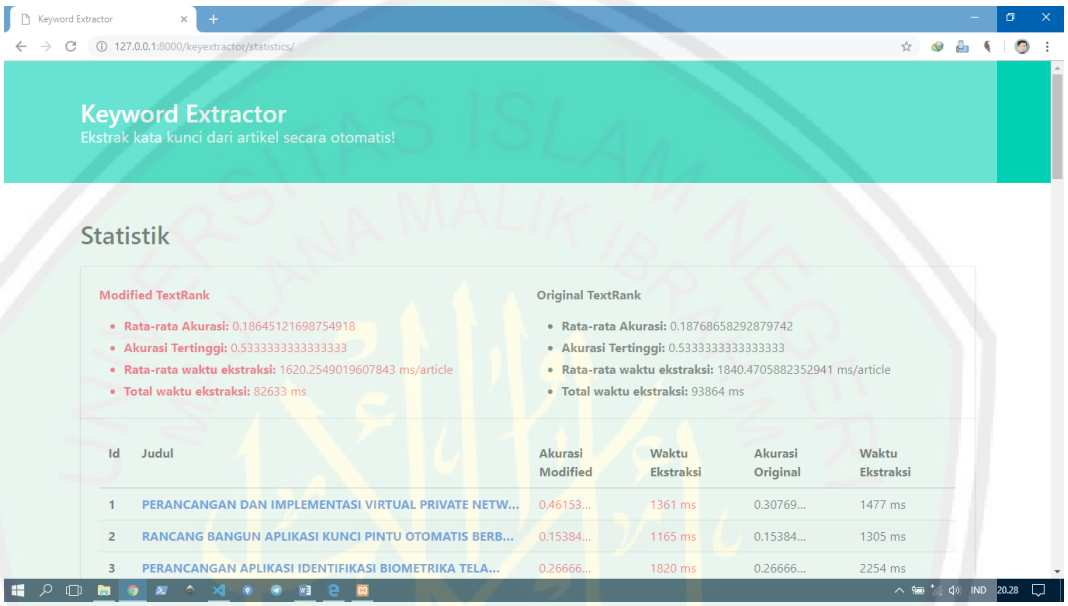
Ketika artikel sudah ter-ekstrak dengan kedua metode, di bawah panel artikel terdapat panel hasil ekstraksi dan hasil pengujian. Panel hasil ekstraksi menampilkan kata kunci hasil ekstraksi dari kedua metode beserta skornya. Panel hasil pengujian menampilkan hasil pengujan dari kedua metode.



Gambar 3.38. Hasil ekstraksi dan pengujian

3.7.4 Halaman Statistik

Halaman ini menampilkan statistik hasil ekstraksi dari keseluruhan artikel dan tabel yang berisi seluruh artikel yang ada di *database*. Halaman statistik ditunjukkan pada **Gambar 3.39**.



Gambar 3.39. Halaman Statistik

BAB IV

HASIL DAN PEMBAHASAN

Bab ini tersusun dari skenario pengujian sistem yang telah dibangun, hasil pengujian sistem dan pembahasan mengenai hasil uji coba sistem beserta integrasi penelitian dengan Islam. Uji coba pada penelitian ini dilakukan untuk mengetahui tingkat efektifitas dan tingkat efisiensi sistem ekstraksi kata kunci yang telah dibangun.

4.1 Skenario Pengujian

Langkah pertama uji coba yaitu menginputkan 50 artikel yang didapat dari jurnal CESS (*Journal of Computer Engineering, System and Sains*) ke dalam aplikasi melalui fitur import pdf. Kemudian melakukan proses ekstraksi ke semua artikel melalui menu Ekstraksi. Ekstraksi yang pertama menggunakan metode *Modified TextRank* dilanjutkan dengan metode *Original TextRank*. Setiap ekstraksi satu artikel dilakukan pengujian akurasi kata kunci dan efisiensi sistem dengan cara menghitung waktu yang dibutuhkan untuk ekstraksi kata kunci dalam satu artikel. Pengujian efektifitas dan efisiensi dilakukan kepada metode *Modified TextRank* dan *Original TextRank* dalam satu artikel.

Untuk mengukur efektifitas dari sistem yaitu dengan cara mengukur akurasi dari algoritma yang dibangun, pengujian dilakukan dengan menggunakan rumus *F-measure* yang terdiri dari nilai *Precision* dan *Recall*. Nilai *Precision*

merupakan nilai perbandingan antara jumlah kata kunci yang benar dengan jumlah kata kunci yang berhasil diekstraksi oleh sistem. Nilai K_1 merupakan jumlah kata kunci hasil ekstraksi sistem, sedangkan nilai K_2 merupakan jumlah kata kunci dalam artikel (Zhang, *et. al*, 2017). Nilai *Precision* berguna untuk mengetahui persentase jumlah kata kunci yang benar dari hasil ekstraksi sistem. Berikut ini merupakan persamaan untuk menghitung *Precision.*

$$P = \frac{|K_1 \cap K_2|}{|K_1|} \tag{4.1}$$

Nilai *Recall* merupakan nilai perbandingan antara jumlah kata kunci yang benar dengan jumlah kata kunci dalam artikel. Nilai *Recall* berguna untuk mengetahui persentase jumlah kata kunci yang benar dari hasil ekstraksi sistem terhadap kata kunci dalam artikel. Berikut ini merupakan persamaan untuk menghitung *Recall.*

$$R = \frac{|K_1 \cap K_2|}{|K_2|} \tag{4.2}$$

Setelah mendapatkan nilai *Precision* dan *Recall*, kemudian menghitung nilai *F-measure*.

$$F = \frac{2 \times P \times R}{P + R} \tag{4.3}$$

Untuk mengukur tingkat efisiensi dari algoritma yang dibangun, penulis mengukur waktu yang diperlukan untuk proses ekstraksi secara keseluruhan, mulai dari proses pengambilan data artikel dari database hingga proses perankingan

selesai. Proses pengukuran dilakukan dengan membuat variabel *start_time* yang diinisialisasi dengan waktu ketika proses pengambilan data artikel, dan variabel *end_time* yang diinisialisasi dengan waktu ketika proses perankingan selesai. Kemudian variabel end_time dikurangi dengan variabel start_time, hasil dari pengurangan tersebut merupakan jumlah waktu yang diperlukan untuk proses ekstraksi secara keseluruhan.

Proses perancangan, pembangunan dan uji coba sistem dilakukan pada *hardware* dan *software* dengan spesifikasi sebagai berikut:

1. *Hardware*
    - Processor AMD E1 Dual Core 1.4 Ghz
    - 4.0 GB RAM
    - VGA AMD Radeon HD 7310 Graphics
    - Hard Drive 320 GB
2. *Software*
    - Sistem Operasi Windows 10 64-bit
    - XAMPP Control Panel v3.2.2
    - Visual Studio Code Version 1.23.0
    - Browser Google Chrome versi 73.0

4.2 Hasil Pengujian

Hasil pengujian dari aplikasi *Keyword Extractor* terdiri dari empat bagian, antara lain hasil pengujian menambahkan artikel ke sistem menggunakan fitur import pdf, pengujian fitur ekstraksi kata kunci, pengujian efektifitas dan efisiensi dari algoritma yang dibangun. Artikel-artikel yang telah dikumpulkan kemudian ditambahkan ke dalam sistem satu per satu menggunakan fitur import pdf. Keseluruhan artikel berhasil diinputkan ke dalam sistem dan tersimpan di *database*. **Gambar 4.1** menunjukkan daftar artikel yang terdapat dalam aplikasi.

Id	Judul
1	PERANCANGAN DAN IMPLEMENTASI VIRTUAL PRIVATE NETW...
2	RANCANG BANGUN APLIKASI KUNCI PINTU OTOMATIS BERB...
3	PERANCANGAN APLIKASI IDENTIFIKASI BIOMETRIKA TELA...
4	SISTEM PENDUKUNG KEPUTUSAN DALAM PEMBERIAN BEASIS...
5	PUSH NOTIFICATION SYSTEM PADA PROTOTYPE KENDALI L...
6	ANALISIS QOS PADA PEMBAGIAN BANDWIDTH DENGAN METO...
7	SISTEM PENUNJANG KEPUTUSAN PEMBERIAN KREDIT MENGG...
8	ANALISIS DALAM MENENTUKAN PRODUK BRI SYARIAH TERB...
9	ANALISISIS PEMILIHAN REKOMENDASI PRODUK TERBAIK P...
10	PENERAPAN ALGORITMA FP-GROWTH PADA SISTEM INFORMA...
11	ENKRIPSI DATA DENGAN ALGORITMA KRIPTROGRAFI NOEKE...
12	PERANCANGAN WEB SISTEM INFORMASI KEBIJAKAN DIVERS...
13	PERANCANGAN SISTEM INFORMASI GUDANG OBAT PADA RUM...
14	MEMANFAATKAN ALGORITMA K-MEANS DALAM MENENTUKAN P...
15	MEMBANGUN APLIKASI PERHITUNGAN PPH21 MOBILE ANDRO...
16	PENERAPAN METODE PERBANDINGAN EKSPONENSIAL (MPE) ...
17	RANCANG BANGUN FILE TRANSFER PROTOCOL (FTP) DENGA...
18	PENENTUAN NILAI PARAMETER METODE EXPONENTIAL SMOO...
19	MULTIMEDIA INTERAKTIF BERBASIS KARAKTER SEBAGAI U...
20	IMPLEMENTASI ALGORITMA SAW (SIMPLE ADDITIVE WEIGH...
21	RANCANG BANGUN JARINGAN MENGGUNAKAN MODE PPOE CLI...
22	PEMANFAATAN BASIS DATA ORACLE PADA SISTEM INFORMA...
23	DATA MINING: PENERAPAN RAPIDMINER DENGAN K-MEANS ...
24	MANAJEMEN RISIKO KEAMANAN SISTEM INFORMASI MENGGU...
25	PEMANFAATAN MODEL ENTERPRISE ARCHITECTURE PADA E-...
26	SPK: ALGORITMA MULTI-ATTRIBUTE UTILITY THEORY (MA...
27	IMPLEMENTASI PROMETHEE II DALAM PEMILIHAN PESTISI...
28	PENERAPAN LOCATION BASED SERVICE UNTUK OPTIMASI S...
29	FUNGSI ALGORITMA RSA UNTUK MEMODIFIKASI DAN MENIN...
30	PERANCANGAN SISTEM INFORMASI PERSEDIAAN BARANG PA...
31	ENKRIPSI SURAT ELEKTRONIK MENGGUNAKAN METODE XXTE...
32	PENERAPAN CLOUD ACCOUNTING DALAM MENUNJANG EFEKTI...
33	SISTEM INFORMASI MONITORING TUGAS AKHIR (SIMTA) B...
34	KLASIFIKASI MUTU MUTIARA BERDASARKAN BENTUK DAN U...
35	PURWARUPA SISTEM PAKAR INDENTIFIKASI JAMUR LAYAK ...
36	PENERAPAN TEKNOLOGI VIRTUAL REALITY PHOTOGRAPHY P...
37	MODEL PENGAMBILAN KEPUTUSAN DENGAN TEKNIK METODE ...
38	PERBANDINGAN NORMALISASI DATA UNTUK KLASIFIKASI W...
39	PENILAIAN TATA KELOLA TEKNOLOGI INFORMASI UNIVERS...
40	PENGONTROLAN ROBOT HUMANOID MENGGUNAKAN METODE SP...
41	TELEGRAMBOT: USING TELEGRAM TO CRAWLING MALWARE T...
42	SISTEM INFORMASI PEMESANAN KAMAR MESS PT. KAI PER...
43	PERBANDINGAN PERFORMA ALGORITMA KOLONI SEMUT DENG...
44	TEKNOLOGI GLOBAL POSITIONING SISTEM (GPS) UNTUK P...
45	PENGUKURAN KUALITAS WEBSITE ELEVENIA MENGGUNAKANN...
46	IMPLEMENTASI ALGORITMA APRIORI UNTUK MENENTUKAN P...
47	PENERAPAN EUCLIDEAN PROBABILITY DALAM PENDETEKSIA...
48	SMART CITY INFRASTRUKTUR: PERANCANGAN INTEGRASI S...
49	RANCANG BANGUN PEMBAGIAN BANWIDTH DAN MONITORING ...
50	EVALUASI KINERJA DOSEN UNIV.SARI MUTIARA INDONESI...

Gambar 4.1. Daftar artikel dalam aplikasi *Keyword Extractor*

Artikel yang telah tersimpan di database kemudian dilakukan ekstraksi kata kunci dengan metode *Modified TextRank* dan *Original TextRank*. Hasil dari

proses ekstraksi yaitu 10 kata kunci yang memiliki skor tertinggi. Hasil kata kunci serta waktu proses ekstraksi yang diperlukan dari lima artikel pada **Tabel 4.1** yakni sebagai berikut:

Tabel 4.1. Hasil ekstraksi pada 5 artikel

No	Judul	Kata Kunci Manual	*Original TextRank*		*Modified TextRank*	
			Kata kunci	Waktu	Kata kunci	Waktu
1	PERANCANGAN DAN IMPLEMENTASI VIRTUAL PRIVATE NETWORK (VPN) MENGGUNAKAN PROTOKOL SSTP (SECURE SOCKET TUNNELING PROTOCOL) MIKROTIK DI FAKULTAS MIPA UNIVERSITAS TANJUNGPURA	VPS, SSTP, VPN	sstp, internet, aplikasi, penelitian, fmipa, jaringan, server, mikrotik, vps, pengguna	1418 ms	Internet, penelitian, sstp, jaringan, fmipa, aplikasi, pengguna, server, data, vpn	1192 ms
2	RANCANG BANGUN APLIKASI KUNCI PINTU OTOMATIS BERBASIS MIKROKONTROLER ARDUINO MENGGUNAKAN SMARTPHONE ANDROID	Kunci pintu otomatis, Mikrokontroler, QRCode scanner	perangkat, sistem, mikrokontroler, smartphone android, tahap, gbr, bluetooth, android, aplikasi, pintu	1277 ms	smartphone android, aplikasi, mikrokontroler, perangkat, sistem, program, proses, flowchart, bluetooth, permasalahan	1099 ms
3	PERANCANGAN APLIKASI IDENTIFIKASI BIOMETRIKA TELAPAK TANGAN MENGGUNAKAN METODE FREEMAN CHAIN CODE	Aplikasi, Identifikasi, Biometrika, Telapak Tangan, Chain Code	citra, tangan, gbr, chain code, aplikasi, gambar, jpg siti, data, freeman chain code, sistem	2114 ms	citra, chain code, aplikasi, data, tangan, sistem, gambar, pengguna, freeman chain code, penelitian	1751 ms

4	SISTEM PENDUKUNG KEPUTUSAN DALAM PEMBERIAN BEASISWA DENGAN MENGGUNAKAN METODE WEIGHTED PRODUCT	Beasiswa, Weighted Product	beasiswa, keputusan, alternatif, kriteria, wp, metode, tabel, gambar, bobot, sistem	1513 ms	beasiswa, keputusan, alternatif, kriteria, penelitian, sistem, peserta, atribut, metode, calon	1666 ms
5	PUSH NOTIFICATION SYSTEM PADA PROTOTYPE KENDALI LISTRIK RUMAH	push notification, kendali listrik, perangkat bergerak	lampu, raspberry pi, notifikasi, intensitas cahaya, perangkat smartphone, listrik, smartphone, pengguna, push notification, jaringan,	1364 ms	intensitas cahaya, lampu, notifikasi, raspberry pi, push notification, listrik, android, smartphone, sensor, pengguna,	1110 ms

Setelah kata kunci didapat, kemudian dilakukan pengujian efektifitas dan efisiensi dari sistem dengan cara menghitung akurasi dan mengukur waktu ekstraksi. Hasil dari pengujian 50 artikel dapat dilihat pada **Tabel 4.2**.

Tabel 4.2. Hasil uji akurasi dan waktu ekstraksi

No Artikel	*Original TextRank*				*Modified TextRank*			
	Recall	Precision	F-Measure (Akurasi)	Waktu Ekstraksi (ms)	Recall	Precision	F-Measure (Akurasi)	Waktu Ekstraksi (ms)
1	66,67%	20,00%	**30,77%**	**1418**	66,67%	20,00%	**30,77%**	**1192**
2	33,33%	10,00%	**15,38%**	**1277**	33,33%	10,00%	**15,38%**	**1099**
3	40,00%	20,00%	**26,67%**	**2114**	40,00%	20,00%	**26,67%**	**1751**
4	50,00%	10,00%	**16,67%**	**1513**	50,00%	10,00%	**16,67%**	**1666**
5	33,33%	10,00%	**15,38%**	**1364**	33,33%	10,00%	**15,38%**	**1110**
6	80,00%	40,00%	**53,33%**	**2607**	80,00%	40,00%	**53,33%**	**2130**
7	16,67%	10,00%	**12,50%**	**1711**	0,00%	0,00%	**0,00%**	**1502**
8	33,33%	10,00%	**15,38%**	**1902**	33,33%	10,00%	**15,38%**	**1599**
9	66,67%	20,00%	**30,77%**	**1527**	33,33%	10,00%	**15,38%**	**1285**
10	33,33%	10,00%	**15,38%**	**1068**	33,33%	10,00%	**15,38%**	**980**
11	33,33%	10,00%	**15,38%**	**1973**	33,33%	10,00%	**15,38%**	**1605**

12	0,00%	0,00%	**0,00%**	**3131**	25,00%	10,00%	**14,29%**	**2841**
13	25,00%	10,00%	**14,29%**	**1372**	50,00%	20,00%	**28,57%**	**1374**
14	33,33%	10,00%	**15,38%**	**1853**	66,67%	20,00%	**30,77%**	**1515**
15	20,00%	10,00%	**13,33%**	**2787**	20,00%	10,00%	**13,33%**	**2743**
16	50,00%	10,00%	**16,67%**	**1752**	50,00%	10,00%	**16,67%**	**1545**
17	25,00%	10,00%	**14,29%**	**1755**	25,00%	10,00%	**14,29%**	**1595**
18	75,00%	30,00%	**42,86%**	**1457**	75,00%	30,00%	**42,86%**	**1338**
19	0,00%	0,00%	**0,00%**	**2451**	0,00%	0,00%	**0,00%**	**2174**
20	50,00%	20,00%	**28,57%**	**1775**	50,00%	20,00%	**28,57%**	**1582**
21	0,00%	0,00%	**0,00%**	**2591**	0,00%	0,00%	**0,00%**	**2167**
22	50,00%	20,00%	**28,57%**	**1686**	50,00%	20,00%	**28,57%**	**1612**
23	25,00%	10,00%	**14,29%**	**1902**	25,00%	10,00%	**14,29%**	**1625**
24	25,00%	10,00%	**14,29%**	**3356**	25,00%	10,00%	**14,29%**	**3238**
25	66,67%	20,00%	**30,77%**	**1894**	66,67%	20,00%	**30,77%**	**1575**
26	50,00%	20,00%	**28,57%**	**1370**	50,00%	20,00%	**28,57%**	**1232**
27	66,67%	20,00%	**30,77%**	**1741**	33,33%	10,00%	**15,38%**	**1518**
28	33,33%	10,00%	**15,38%**	**1494**	66,67%	20,00%	**30,77%**	**1277**
29	0,00%	0,00%	**0,00%**	**2333**	0,00%	0,00%	**0,00%**	**2078**
30	33,33%	10,00%	**15,38%**	**1650**	0,00%	0,00%	**0,00%**	**1401**
31	66,67%	20,00%	**30,77%**	**1947**	66,67%	20,00%	**30,77%**	**1726**
32	33,33%	10,00%	**15,38%**	**1315**	33,33%	10,00%	**15,38%**	**1149**
33	50,00%	20,00%	**28,57%**	**1396**	25,00%	10,00%	**14,29%**	**1150**
34	66,67%	20,00%	**30,77%**	**1565**	66,67%	20,00%	**30,77%**	**1357**
35	40,00%	20,00%	**26,67%**	**2623**	40,00%	20,00%	**26,67%**	**2271**
36	0,00%	0,00%	**0,00%**	**1145**	0,00%	0,00%	**0,00%**	**1001**
37	33,33%	10,00%	**15,38%**	**2075**	66,67%	20,00%	**30,77%**	**1532**
38	0,00%	0,00%	**0,00%**	**1503**	0,00%	0,00%	**0,00%**	**1314**
39	0,00%	0,00%	**0,00%**	**1663**	0,00%	0,00%	**0,00%**	**1391**
40	0,00%	0,00%	**0,00%**	**2585**	20,00%	10,00%	**13,33%**	**2420**
41	28,57%	20,00%	**23,53%**	**1884**	28,57%	20,00%	**23,53%**	**1565**
42	0,00%	0,00%	**0,00%**	**1430**	40,00%	20,00%	**26,67%**	**1256**
43	20,00%	10,00%	**13,33%**	**2286**	20,00%	10,00%	**13,33%**	**1734**
44	66,67%	20,00%	**30,77%**	**2764**	66,67%	20,00%	**30,77%**	**1739**
45	50,00%	20,00%	**28,57%**	**3147**	50,00%	20,00%	**28,57%**	**1950**
46	0,00%	0,00%	**0,00%**	**1332**	0,00%	0,00%	**0,00%**	**1190**
47	100,00%	30,00%	**46,15%**	**1590**	66,67%	20,00%	**30,77%**	**1315**
48	25,00%	10,00%	**14,29%**	**1513**	25,00%	10,00%	**14,29%**	**1298**
49	50,00%	20,00%	**28,57%**	**1825**	50,00%	20,00%	**28,57%**	**1711**
50	33,33%	10,00%	**15,38%**	**1615**	33,33%	10,00%	**15,38%**	**1366**

Pengujian akurasi dilanjutkan dengan menggunakan *Confusion Matrix*. Klasifikasi yang digunakan yakni klasifikasi *binary* yang hanya memiliki dua

keluaran kelas yakni *True* dan *False*. Keluaran *True* didapat apabila jumlah kata kunci hasil ekstraksi sistem yang benar sama dengan jumlah kata kunci aslinya. Hasil klasifikasi dapat dilihat pada **Tabel 4.3**.

Tabel 4.3. Hasil klasifikasi kata kunci

No	*Original TextRank*	*Modified TextRank*	No	*Original TextRank*	*Modified TextRank*	No	*Original TextRank*	*Modified TextRank*
1	FALSE	FALSE	18	FALSE	FALSE	35	FALSE	FALSE
2	FALSE	FALSE	19	FALSE	FALSE	36	FALSE	FALSE
3	FALSE	FALSE	20	FALSE	FALSE	37	FALSE	FALSE
4	FALSE	FALSE	21	FALSE	FALSE	38	FALSE	FALSE
5	FALSE	FALSE	22	FALSE	FALSE	39	FALSE	FALSE
6	FALSE	FALSE	23	FALSE	FALSE	40	FALSE	FALSE
7	FALSE	FALSE	24	FALSE	FALSE	41	FALSE	FALSE
8	FALSE	FALSE	25	FALSE	FALSE	42	FALSE	FALSE
9	FALSE	FALSE	26	FALSE	FALSE	43	FALSE	FALSE
10	FALSE	FALSE	27	FALSE	FALSE	44	FALSE	FALSE
11	FALSE	FALSE	28	FALSE	FALSE	45	FALSE	FALSE
12	FALSE	FALSE	29	FALSE	FALSE	46	FALSE	FALSE
13	FALSE	FALSE	30	FALSE	FALSE	47	TRUE	FALSE
14	FALSE	FALSE	31	FALSE	FALSE	48	FALSE	FALSE
15	FALSE	FALSE	32	FALSE	FALSE	49	FALSE	FALSE
16	FALSE	FALSE	33	FALSE	FALSE	50	FALSE	FALSE
17	FALSE	FALSE	34	FALSE	FALSE			

Hasil klasifikasi menunjukkan bahwa algoritma *Original TextRank* mendapatkan satu artikel yang kata kunci hasil ekstraksinya terklasifikasi benar. Artinya algoritma *Original TextRank* mendapatkan nilai akurasi kebenaran sebesar 2%. Sementara untuk algoritma *Modified TextRank* tidak ada hasil ekstraksi kata kuncinya yang terklasifikasi benar.

Dari hasil uji akurasi dan waktu ekstraksi diperoleh tingkat efektifitas dan efisiensi dari sistem yang dibangun. Tingkat efektifitas dari sistem dilihat dari

seberapa besar akurasi dari kata kunci yang dihasilkan sistem. Nilai akurasi terdiri dari *Precision*, *Recall* dan *F-Measure*. Hasil uji efektifitas sistem pada 50 artikel dapat dilihat pada **Tabel 4.4**.

Tabel 4.4. Tingkat efektifitas dari algoritma yang dibangun

	Original TextRank	***Modified TextRank***
Rata-rata F-Measure (Akurasi)	**18,38%**	**18,91%**
Rata-rata Recall	35,57%	36,27%
Rata-rata Precision	12,60%	13,00%
F-Measure (Akurasi) tertinggi	53,33%	53,33%
Recall tertinggi	100,00%	80,00%
Precision tertinggi	40,00%	40,00%
Jumlah *zero accuracy*	10	9

Untuk tingkat efisiensi dari sistem yang dibangun, dilihat dari waktu ekstraksi yang diperlukan. Dapat dilihat bahwa waktu ekstraksi yang diperlukan untuk mengekstraksi 50 artikel menggunakan algoritma *Modified TextRank* lebih kecil daripada menggunakan algoritma *Original TextRank*.

Tabel 4.5. Tingkat efisiensi dari algoritma yang dibangun

	Original TextRank	***Modified TextRank***
Rata-rata waktu ekstraksi	1880,54 ms/article	1615,68 ms/article
Waktu ekstraksi terendah	1068 ms	980 ms
Total waktu ekstraksi	94027 ms	80784 ms

4.3 Pembahasan

Setelah diperoleh hasil pengujian efektifitas dan efisiensi sistem, dapat dilihat bahwa algoritma *Modified TextRank* yang dibangun oleh penulis memperoleh efektifitas akurasi dengan rata-rata nilai *F-Measure* sebesar 18,91%,

rata-rata nilai *Recall* sebesar 36,27%, dan rata-rata nilai *Precision* sebesar 13%. Dari **Tabel 4.4**, dapat disimpulkan bahwa nilai rata-rata akurasi dari *Modified TextRank* lebih tinggi dibandingkan dengan algoritma *Original TextRank*.

Untuk nilai *F-Measure* tertinggi, kedua algoritma memperoleh nilai yang sama, yakni 53,33%, dan juga nilai *Precision* tertinggi, keduanya memperoleh nilai yang sama, yakni 40%. Namun untuk nilai *Recall*, algoritma *Original TextRank* memperoleh nilai 100%, 20% lebih tinggi daripada algoritma *Modified TextRank*. Untuk jumlah *zero accuracy*, yakni jumlah artikel yang memiliki nilai akurasi 0, algoritma *Modified TextRank* berjumlah 9 artikel dan algoritma *Original TextRank* berjumlah 10 artikel.

Nilai akurasi hasil dari pengujian sistem kurang dari 20%. Hal ini dikarenakan jumlah kata kunci manual yang dimiliki oleh artikel hanya berkisar antara 3-5 kata kunci. Sedangkan kata kunci yang dihasilkan sistem berjumlah 10 kata kunci. Perbedaan jumlah antara kata kunci manual dengan kata kunci hasil ekstraksi berpengaruh pada nilai *Precision*. Dapat dilihat pada **Tabel 4.4** bahwa nilai *Precision* kurang dari 15%. Dan juga artikel yang dikumpulkan oleh penulis memiliki isi yang berbeda-beda, dan juga kata kunci manual di artikel terkadang jarang ditemukan di bagian isi artikel. Namun untuk nilai Recall tidak berpengaruh pada rendahnya nilai akurasi. Nilai *Recall* diperoleh berdasarkan jumlah kata kunci manual yang sama dengan kata kunci sistem.

Tingkat efisiensi dihitung dari waktu yang diperlukan untuk proses ekstraksi. Pada **Tabel 4.5**, waktu yang diperlukan untuk proses ekstraksi dari 50 artikel menggunakan algoritma *Modified TextRank* lebih sedikit dibandingkan dengan menggunakan algoritma *Original TextRank*. Dapat disimpulkan bahwa algoritma yang dibangun oleh penulis lebih efisien daripada algoritma *Original TextRank*. Algoritma *Modified TextRank* membutuhkan waktu lebih sedikit daripada algoritma *Original TextRank* dikarenakan pada perhitungan skor dengan rumus *TextRank* hanya dilakukan satu kali perulangan. Tidak seperti algoritma *Original TextRank* yang membutuhkan beberapa kali perulangan hingga nilai *error rate* 0,0001 yang menyebabkan proses ekstraksi membutuhkan waktu sedikit lebih lama daripada algoritma *Modified TextRank*.

Dari hasil implementasi sistem yang dibangun, pengguna dapat mengekstraksi kata kunci dari artikel dengan mudah. Pengguna dapat mengekstraksi kata kunci dari satu artikel atau beberapa artikel dalam satu klik. Waktu yang diperlukan untuk proses ekstraksi kata kunci per satu artikel yaitu kurang lebih 1-3 detik. Hasil pengujian akurasi kata kunci hasil ekstraksi sistem memperoleh nilai *Recall* mencapai 100%. Hal ini menunjukkan bahwa kata kunci hasil ekstraksi sistem dapat digunakan sebagai kata kunci dari artikel. Dapat diperoleh kesimpulan bahwa sistem ini dapat memudahkan pengguna untuk mencari kata kunci dari artikel hanya dalam satu klik dan dalam waktu yang singkat.

Allah swt. berfirman bahwa sesungguhnya sesudah kesulitan pasti ada kemudahan (QS. Al-Insyirah: 5-6).

فَاِنَّ مَعَ الْعُسْرِ يُسْرًا ۝ اِنَّ مَعَ الْعُسْرِ يُسْرًا ۝

"Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan." (QS. Al-Insyirah (94): Ayat 5-6).

Allah Swt. menceritakan bahwa sesungguhnya sesudah kesulitan pasti ada kemudahan, kemudian berita ini diulangi-Nya lagi. Al-Hasan Al-Bashri mengatakan, "Para sahabat dahulu berkata bahwa satu kesulitan tidak mungkin mengalahkan dua kemudahan". Ibnu Katsir menjelaskan perkataan tersebut dengan kaedah ilmu bahasa Arab. "Kesulitan (*al-'usru*) menggunakan *isim ma'rifah* di dua keadaan, maka kesulitan pertama dan kedua dianggap satu atau dianggap sama. Sedangkan kemudahan (*yusrun*) menggunakan *isim nakirah*, sehingga kemudahan itu berbilang, bukan hanya satu. Oleh karenanya disebut, "Satu kesulitan mustahil mengalahkan dua kemudahan." Kesulitan pertama yang disebut dalam ayat sama dengan kesulitan kedua, berarti kesulitan itu hanya satu. Sedangkan kemudahan itu berbilang." (Tafsir Al-Qur'an Al-'Azhim, 7: 598). Dari penjelasan tersebut dapat disimpulkan bahwa suatu kesulitan atau masalah pasti ada jalan keluarnya, dan jika jalan keluar yang satu tidak menyelesaikan masalah maka masih ada jalan keluar yang lain untuk menyelesaikan masalah. Tidak mungkin suatu kesulitan atau masalah tidak memiliki jalan keluar. Dan tidak mungkin pula jumlah kesulitan atau

masalah yang ada melebihi jumlah kemudahan atau solusi yang ada untuk menyelesaikan masalah itu sendiri.

Dari Abu Hurairah Radhiyallahu anhu, Nabi Shallallahu ‘alaihi wa sallam bersabda:

مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا، نَفَّسَ اللهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ،

“Barangsiapa yang melapangkan satu kesusahan dunia dari seorang Mukmin, maka Allâh melapangkan darinya satu kesusahan di hari Kiamat.”(Hadits shahih: Diriwayatkan oleh Muslim (no. 2699), Ahmad (II/252, 325), Abu Dawud (no. 3643), At-Tirmidzi (no. 2646), Ibnu Majah (no. 225), dan Ibnu Hibban (no. 78-Mawaarid), dari Shahabat Abu Hurairah radhiyallaahu ‘anhu)

Aplikasi ini diharapkan dapat memudahkan pengguna khususnya untuk pengelola/penulis jurnal ilmiah dimana pengelola/penulis jurnal dapat mengekstrak kata kunci dari artikel-artikel yang ada dalam waktu yang singkat. Dan juga dapat memudahkan pembaca artikel mendapatkan kata kunci dari artikel yang dibacanya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil implementasi sistem ekstraksi kata kunci yang telah dirancang kemudian dilakukan uji coba sistem, penulis mendapatkan kesimpulan sebagai berikut:

a. Tingkat efektifitas dari sistem dilihat dari seberapa besar akurasi dari kata kunci yang dihasilkan sistem. Nilai akurasi terdiri dari *Precision*, *Recall* dan *F-Measure*. Hasil pengujian sistem dengan data uji 50 artikel menunjukkan bahwa algoritma *Modified TextRank* yang penulis bangun memperoleh nilai akurasi dengan rata-rata nilai *F-Measure* sebesar 18,91%, rata-rata nilai *Recall* sebesar 36,27%, dan rata-rata nilai *Precision* sebesar 13%. Sementara untuk algoritma *Original TextRank* memperoleh rata-rata nilai *F-Measure* sebesar 18,38%, rata-rata nilai *Recall* sebesar 35,57%, dan rata-rata nilai *Precision* sebesar 12,6%. Dapat disimpulkan bahwa nilai rata-rata akurasi dari *Modified TextRank* lebih tinggi dibandingkan dengan algoritma *Original TextRank*. Oleh karena itu, tingkat efektifitas dari algoritma *Modified TextRank* lebih tinggi dibandingkan dengan algoritma *Original TextRank*.
b. Tingkat efisiensi dihitung dari waktu yang diperlukan untuk proses ekstraksi. Total waktu yang diperlukan untuk proses ekstraksi kata kunci dari 50 artikel menggunakan algoritma *Modified TextRank* yaitu 80784 ms dengan waktu rata-

rata ekstraksi 1615,68 ms/artikel. Sedangkan untuk algoritma *Original TextRank* waktu yang diperlukan yaitu 94027 ms dengan waktu rata-rata ekstraksi 1880,54 ms/artikel. Dapat dilihat bahwa algoritma *Modified TextRank* membutuhkan waktu lebih sedikit dibandingkan algortima *Original TextRank*. Oleh karena itu, tingkat efisiensi dari algoritma *Modified TextRank* lebih tinggi dibandingkan dengan algoritma *Original TextRank.*

5.2 Saran

Dari kesimpulan yang diperoleh dapat dilihat bahwa nilai akurasi masih tergolong rendah yakni dibawah 20%. Hal ini disebabkan karena algoritma yang dibangun tidak melibatkan analisis teks yang lebih mendalam. Maka dari itu, untuk pengembangan sistem ekstraksi kata kunci di penelitian yang akan datang diperlukan beberapa tambahan dan perbaikan untuk mendapatkan hasil kata kunci yang lebih baik, diantaranya:

a. Perbaikan pada proses pencarian kandidat kata kunci dengan melibatkan *semantic analysis* atau *lexical analysis* agar dapat mengetahui hubungan antar kata dan memperoleh kandidat kata kunci yang lebih baik.
b. Penambahan kamus kata dari berbagai bahasa sehingga proses ekstraksi kata kunci dapat dilakukan pada artikel berbahasa asing.

DAFTAR PUSTAKA

Figueroa, G., & Chen, Y. S. 2014. *Collaborative Ranking between Supervised and Unsupervised Approaches for Keyphrase Extraction*. ROCLING XXVI (2014), pp. 110-124.

Hu, X., & Wu, B. 2006. *Automatic Keyword Extraction Using Linguistic Features*. Sixth IEEE International Conference on Data Mining - Workshops. IEEE.

Ibnu Katsir. Tahqiq: Syaikh Abu Ishaq Al-Huwainiy. 2010. *Tafsir Al-Qur'an Al-'Azhim*. Cetakan pertama, tahun 1431 H.. Penerbit Dar Ibnul Jauzi.

Internet World Stats. 2017. *Top 20 Internet Countries by Users*. http://www.internetworldstats.com/top20.htm. Diakses tanggal 5 November 2018.

Li, G., & Wang, H. 2014. *Improved Automatic Keyword Extraction Based on TextRank Using Domain Knowledge*. Natural Language Processing and Chinese Computing, pp. 403-413.

Li, W., & Zhao W. 2016. *TextRank algorithm by exploiting Wikipedia for short text keywords extraction*. 2016 3rd International Conference on Information Science and Control Engineering, pp. 683-686

Mihalcea, R., & Tarau, P. 2004. *TextRank : Bringing order into texts*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain.

Ramadhiana, R. 2017. *Ekstraksi Kata Kunci Otomatis Teks Berbahasa Indonesia Menggunakan Metode Textrank*. Skripsi Sarjana (S1) Teknologi Informasi Fakultas Ilmu Komputer Dan Teknologi Informasi Universitas Sumatera Utara.

Sari, D.P., & Purwarianti, A. 2014. *Ekstraksi Kata Kunci Otomatis untuk Dokumen Bahasa Indonesia Studi Kasus: Artikel Jurnal Ilmiah Koleksi PDII LIPI*. BACA: Jurnal Dokumentasi dan Informasi, pp. 139-147.

Wen, Y., Zhang, P., & Yuan, H. 2016. *Research on Keyword Extraction Based on Word2Vec Weighted TextRank*. 2016 2nd IEEE International Conference on Computer and Communications, pp. 2109-2113.

Wicaksono, A.F., & Purwarianti, A. 2010. *HMM Based Part-of-Speech Tagger for Bahasa Indonesia*. On Proceedings of 4th International MALINDO (Malay and Indonesian Language) Workshop.

Zhang, X., An, J., & Liu, W. 2017. *Research And Implementation Of Keyword Extraction Algorithm Based On Professional Background Knowledge*. 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI 2017).

Zhao, D., Du, N., Chang, Z., & Li, Y. 2017. *Keyword extraction for social media short text*. 2017 14th Web Information Systems and Applications Conference, pp. 251-256