Aplikasi Pencarian Jurnal Ilmiah dengan Term Frequency-Inverse Document Frequency

Amanda Tanari, Andreas Handojo, Justinus Andjarwirawan
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: amanda_tanari@yahoo.com, handojo@petra.ac.id, justin@petra.ac.id

ABSTRAK

Kebutuhan seorang peneliti untuk mencari artikel ilmiah untuk dipelajari maupun sebagai pendukung penelitiannya tidak dapat dihindari. Artikel ilmiah yang kemunculannya pesat membuat jurnal ilmiah yang dicetak lebih lambat penerbitannya dibandingkan dengan *website*. Oleh karena itu, para peneliti dapat mencari referensi di *website* agar lebih mudah dan dapat melihat artikel yang terbaru. Namun, hasil pencarian dari artikel ilmiah juga bervariasi di setiap *website* dan urutan hasilnya terkadang tidak sesuai dengan yang ingin dicari.

Penelitian ini mencoba menghasilkan hasil pencarian yang sesuai dengan keinginan pengguna berdasarkan kata kunci yang dimasukkan. Penelitian akan dicoba menggunakan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). Pengurutan hasil pencarian mulai dari bobot terbesar yang didapat dari perhitungan TF-IDF.

Aplikasi dapat menampilkan hasil pencarian yang cenderung relevan dengan satu *term* kata kunci, tetapi bila kata kunci merupakan *phrase*, maka hasil pencarian cenderung tidak relevan. Aplikasi pencarian jurnal ilmiah dengan metode *term frequency-inverse document frequency* ini dapat menjawab kebutuhan mahasiswa maupun dosen dalam mencari jurnal.

Kata Kunci: Term Frequency, Inverse Document Frequency, aplikasi pencarian, REST, PHP

ABSTRACT

The need for a researcher to search scientific articles to be studied as well as supporting his research is unavoidable. Scientific articles whose emergence is rapid makes printed scientific journals published more slowly than websites. Therefore, researchers can find references on the website to make it easier and could see the latest articles. However, the search results of scientific articles also vary on each website and the order of results sometimes does not match what you want to find.

This study tries to produce search results that are in accordance with the wishes of users based on the keywords entered. Research will be tried using the Term Frequency-Inverse Document Frequency (TF-IDF) method. Sorting search results starting from the largest weight obtained from the TF-IDF calculation.

The application showed more relevance results when using one term in the keyword, but if the keyword is a phrase then the results tend to be not relevant. The scientific journal search application using the term frequency-inverse document frequency method can answer the needs of students and lecturers in finding journals.

Keywords: *Term Frequency, Inverse Document Frequency, Search Engine, REST, PHP*

1. PENDAHULUAN

Kebutuhan seorang peneliti untuk mencari artikel ilmiah untuk dipelajari maupun sebagai pendukung penelitiannya tidak dapat dihindari. Artikel ilmiah yang kemunculannya pesat membuat jurnal ilmiah yang dicetak lebih lambat penerbitannya dibandingkan dengan *website.* Oleh karena itu, para peneliti dapat mencari referensi di *website* agar lebih mudah dan dapat melihat artikel yang terbaru. Namun, hasil pencarian dari artikel ilmiah juga bervariasi di setiap *website* dan urutan hasilnya terkadang tidak sesuai dengan yang ingin dicari.

Penelitian ini mencoba menghasilkan hasil pencarian yang sesuai dengan keinginan pengguna berdasarkan kata kunci yang dimasukkan. Penelitian akan dicoba menggunakan metode *Term Frequency - Inverse Document Frequency* (TF-IDF). Dari sebuah survei literatur terhadap penelitian-penelitian sistem rekomendasi artikel, yang dilakukan oleh Joeran Beel, TF-IDF adalah metode perhitungan yang paling populer [1] Sehingga pada penelitian ini akan digunakan metode TF-IDF. Metode ini melakukan perhitungan yang bergantung pada *bag-of-words* (BoW) *model*. *Bag-of-words* (BoW) *model* berarti urutan dari setiap kata dalam sebuah dokumen diabaikan tetapi jumlah kemunculan dari setiap kata diperhitungkan [7].

Di penelitian ini, pengguna akan memasukkan kata kunci yang ingin dicari dan mendapatkan hasil pencarian dari kumpulan data yang ada. Hasil pencarian akan ditampilkan sesuai dengan kata kunci yang dicari. Pengurutan hasil pencarian mulai dari relevansi terbesar yang didapat dari perhitungan TF-IDF. Dengan menggunakan TF-IDF, tiap kata kunci yang dimasukkan pengguna akan dihitung relevansinya terhadap abstrak dari satu artikel dan terhadap abstrak seluruh artikel yang ada.

Penelitian ini berfokus melakukan perhitungan relevansi terhadap kumpulan dokumen yang ada dengan metode TF-IDF. Hasil perhitungan relevansi akan diurutkan mulai paling besar dan ditampilkan ke pengguna agar pengguna dapat mendapatkan hasil yang sesuai.

Data untuk perhitungan relevansi akan diambil dari kumpulan jurnal Pusat Penelitian Universitas Kristen Petra. Kumpulan jurnal yang diambil adalah jurnal-jurnal yang sudah terakreditasi sehingga terjamin kualitasnya.

2. DASAR TEORI

2.1 Term Frequency – Inverse Document Frequency

Menurut Shuvayan Das [2], TF adalah frekuensi dari sebuah kata di dalam sebuah dokumen. Sedangkan IDF adalah invers dari frekuensi dokumen yang mengandung kata tersebut dari kumpulan dokumen. Contoh bila kita mencari "*the rise of analytics*" di Google. Akan sudah pasti bahwa "*the*" akan muncul lebih banyak daripada "*analytics*" tetapi tingkat kepentingan "*analytics*" lebih tinggi. Dalam beberapa kasus, TF-IDF meniadakan efek dari kata yang memiliki frekuensi tinggi dalam menentukan tingkat kepentingan sebuah dokumen.

Rumus IDF, yaitu: $idf_t = \log10(N/df_t)$ (2.1)

Keterangan :

N : jumlah seluruh dokumen di kumpulan dokumen.

df_t : jumlah dokumen yang mengandung kata yang menjadi target. Semakin sedikit jumlah dokumen yang mengandung kata target, maka bobot IDF akan semakin besar.

Rumus TF-IDF adalah mengkalikan bobot TF dengan IDF dari tiap kata. Rumus TF-IDF, yaitu: $Tf\text{-}idf_t = tf_t * Idf_t$

2.2 Tokenisasi

Tokenisasi adalah proses memecah teks yang dapat berupa kalimat atau paragraf menjadi bagian-bagian kecil yang disebut dengan token. Sebuah token adalah bagian dari karakter-karakter yang berurutan dari sebuah dokumen yang bila digabung menjadi satu kesatuan akan membentuk unit semantik yang berguna untuk diproses [7]. Pemecahan bagian dalam teks biasanya menggunakan tanda baca spasi, bahkan juga termasuk membuang tanda baca. Contoh tokenisasi dari kalimat "Dadang makan sate, nasi, dan bakso." akan menghasilkan enam token, yaitu "Dadang", "makan", "sate", "nasi", "dan" dan "bakso".

2.3 Stop Words Removal

Stop words adalah kata yang nilainya sangat sedikit terhadap sebuah dokumen. Biasanya mempunyai frekuensi yang tinggi dalam dokumen tetapi sebenarnya tidak berarti, misalnya hanya untuk menggabungkan kalimat. [7]

Kata-kata yang termasuk dalam *stop words* ini tidak berarti, sehingga lebih baik untuk menghilangkan kata-kata yang termasuk dalam *stop words*. Penghilangan *stop words* ini berfungsi untuk mengurangi term yang disimpan di basis data dan mengurangi waktu proses. Contoh stop words, yaitu "*a*", "*an*", "*and*", "*in*", "*of*", "*or*", dan "*that*".

2.4 Porter *Stemmer*

Algoritma paling umum untuk stemming Bahasa Inggris dan telah terbukti efektif adalah Porter *Stemmer*. *Website* resmi untuk Porter *Stemmer* adalah https://tartarus.org/martin/PorterStemmer/. [7]

Algoritma Porter Stemmer merupakan algoritma pembuangan imbuhan untuk mendapatkan kata dasar atau *root word*. Algoritma ini memiliki lima langkah utama, yaitu penanganan *plurals* dan *past participles*, penghapusan suffiks, mengubah suffiks umum, mengubah suffiks spesial, mengubah kata dengan akhiran –e, dan menghapus huruf pada akhiran ganda.

2.5 New Porter *Stemmer*

Algoritma ini merupakan pengembangan dari Porter *Stemmer* yang dilakukan oleh Wahiba Ben Abdessalem Karaa [5]. Algoritma ini memiliki enam penanganan yang dilakukan untuk mengatasi kelemahan Porter *Stemmer* adalah untuk bentuk *irregular* berpola dan tidak berpola, bentuk *past* dan *present participle* yang memiliki akhiran –s, kata berakhiran –y yang tidak mengandung huruf vokal, kata dengan akhiran konsonan ganda, suffiks dalam bentuk *past* dan *present participle*, dan *compound suffixes*.

2.6 REST

REST – (*Representational state transfer*) berkaitan dengan hubungan *client* dan *server* serta bagaimana *state* disimpan. *Request* dan *response* dibuat berdasarkan proses transfer antar seluruh *resource* yang ada. REST tidak membutuhkan format seperti *header-message.* REST pertukaran datanya menggunakan JSON sehingga lebih mudah untuk di-*parse* dan digunakan oleh komputer [4].

Website yang menggunakan arsitektur REST akan disebut sebagai RESTful *web service*. RESTful *web service* menggunakan GET, PUT, POST dan DELETE dengan metode HTTP untuk melakukan CRUD (*Create*, *Read*, *Update*, *Delete*) terhadap *resource* [9].

2.7 Composer

Menurut Arya Febian [3], Composer adalah *dependency manager* khusus PHP yang memiliki fungsionalitas seperti Gem (Ruby) atau Maven (Java). Suatu *library* dapat diinstall melalui Composer dan Composer akan secara otomatis menginstall maupun meng-*update library* yang dibutuhkan.

Menurut Nafies Luthfi [6], yang dimaksud *dependency manager* dalam Composer adalah bahwa Composer akan mengelola "ketergantungan" antara *library-library* PHP (yang disebut *package*), yang dibuat oleh seseorang untuk digunakan orang lain.

2.8 Slim Framework

Slim Framework adalah *micro framework* di PHP yang membantu dalam pembuatan *web applications* dan API. Slim berfungsi sebagai alat komunikasi yang menerima HTTP *request*, mengirimkan *request* tersebut ke *code* yang sesuai, dan mengembalikan HTTP *response*.

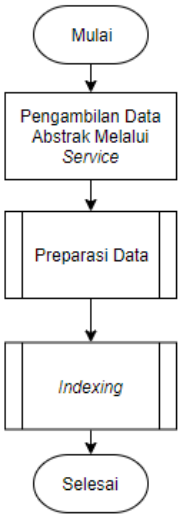
Menurut Friandy Dwi Noviandha [8], *micro framework* biasanya digunakan untuk proyek skala kecil yang memiliki tujuan khusus dan tingkat kompleksitas rendah. Fitur-fitur utama dari Slim Framework, yaitu HTTP router, middleware, dependency injection, PSR-7 support. Beberapa contoh *micro framework* lain, yaitu Silex, Lumen, dan Phalcon.

3. DESAIN SISTEM

3.1 Flowchart Gambaran Besar Sistem

Aplikasi yang dibuat memiliki empat langkah utama. Gambaran besar aplikasi untuk tiga langkah pertama dapat dilihat pada Gambar 1 dan langkah keempat dapat dilihat pada Gambar 4. Langkah pertama adalah melakukan pengambilan data abstrak dari basis data untuk melakukan *indexing*. Langkah kedua adalah melakukan preparasi data atau *data preprocessing* terhadap abstrak yang didapat, seperti menghapus *stopwords*, melakukan

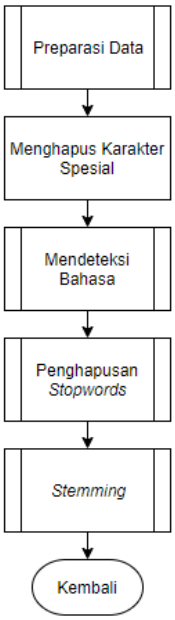
stemming, dan penghapusan karakter spesial. Langkah ketiga adalah proses *indexing* untuk mengetahui setiap kata ada di artikel mana, jumlah kemunculan tiap kata di artikel, dan *document frequency*. Langkah keempat adalah proses pencarian atau *searching* dokumen sesuai kata kunci yang dimasukkan pengguna, termasuk perhitungan TFIDF. Langkah keempat dilakukan berdasarkan hasil *indexing* yang telah didapat dari tiga langkah utama yang sebelumnya telah dilakukan.



Gambar 1. *Flowchart* Gambaran Besar Sistem

3.2 Flowchart Preparasi Data

Setelah melakukan pengambilan data abstrak, akan dilakukan preparasi data. Langkah-langkah dalam preparasi data dapat dilihat pada Gambar 2.



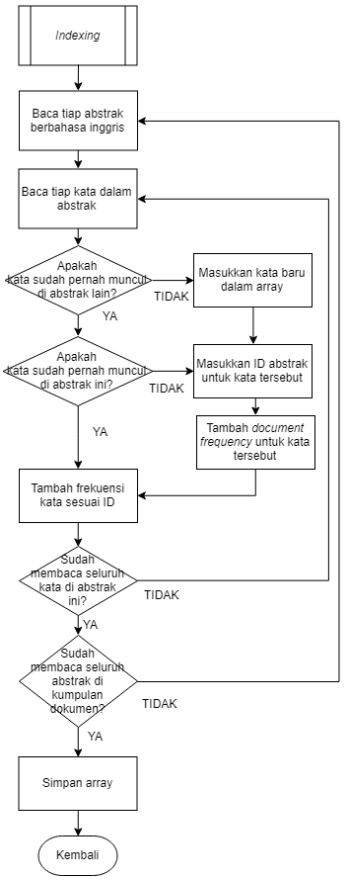
Gambar 2. *Flowchart* Preparasi Data

Penjelasan untuk tiap langkah, yaitu:

- Menghapus karakter spesial, langkah ini akan menghapus tag HTML yang ada di abstrak, men-decode karakter UTF-8, menghapus spesial karakter selain – (tanda penghubung), / (garis miring), . (tanda titik), dan ? (tanda tanya). Karakter tanda penghubung, garis miring, tanda titik, dan tanda tanya akan diganti menjadi spasi. Kemudian akan dilakukan penghapusan terhadap seluruh karakter yang tidak termasuk huruf A/a hingga Z/z dan angka 0 hingga 9. Kemudian seluruh karakter akan dibuat menjadi huruf kecil.
- Mendeteksi bahasa, langkah ini akan mencari tahu apakah abstrak yang didapat dalam Bahasa Inggris, Indonesia, atau keduanya. Bila terdapat dua bahasa, akan ada kata "Abstract in Bahasa Indonesia" sebagai pembatas antara kedua bahasa. Sehingga abstrak akan di *split* bila terdapat pembatas tersebut. Tetapi bila tidak ada pembatas, maka akan dicek apakah abstrak mengandung Bahasa Indonesia. Bila ya, berarti abstrak merupakan Bahasa Indonesia. Bila tidak, berarti abstrak merupakan Bahasa Inggris.
- Penghapusan *stopwords*, langkah ini akan menghapus kata dari abstrak berbahasa Inggris yang termasuk dalam daftar *stopwords*.
- *Stemming*, langkah ini akan mencari kata dasar atau root word dari setiap kata yang ada di abstrak berbahasa Inggris. Alur stemming dengan metode Porter *Stemmer*

3.3 Flowchart *Indexing*

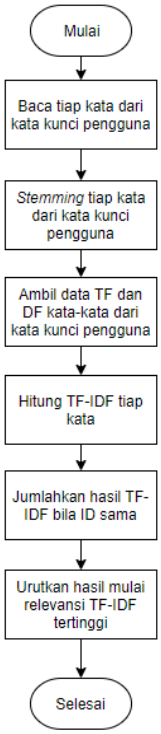
Langkah untuk melakukan *indexing* adalah dengan menghitung *term frequency* dan *document frequency* dari tiap kata. Untuk menghitung *term frequency* dan *document frequency* dari tiap kata, maka perlu dilakukan proses pembacaan dan penghitungan dari setiap kata yang ada di setiap abstrak dalam kumpulan abstrak. Alur dari *indexing* dapat dilihat pada Gambar 3.



Gambar 3. *Flowchart Indexing*

3.4 Flowchart *Searching*

Langkah *searching* ini akan dijalankan bila pengguna memasukkan kata kunci. Pada proses ini, akan dilakukan perhitungan relevansi antara kata kunci terhadap kata yang sudah dilakukan proses *indexing* sebelumnya. Alur proses *searching* dapat dilihat pada Gambar 4.



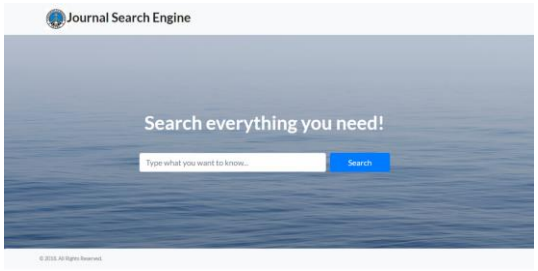
Gambar 4. *Flowchart searching*

4. PENGUJIAN SISTEM

Pengujian dilakukan dengan menguji aplikasi, membandingkan Porter *Stemmer* dan New Porter *Stemmer*, pengujian preparasi data, mengecek apakah perhitungan TF-IDF yang dilakukan telah benar.

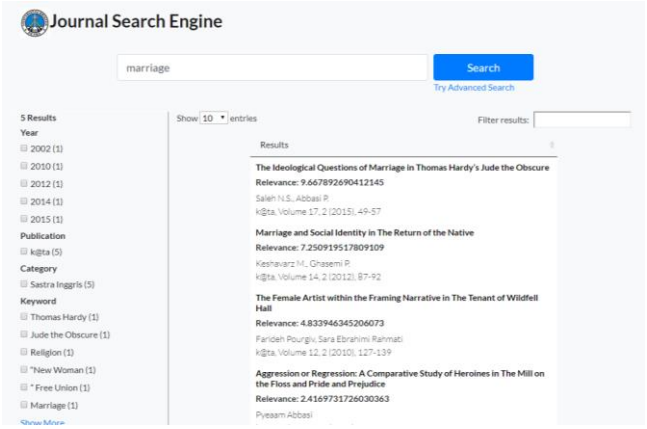
4.1 *Interface* Aplikasi

Halaman awal pada aplikasi ini dapat dilihat pada Gambar 5 Pengguna dapat memasukkan kata kunci yang diinginkan, kemudian menekan tombol *search* sehingga akan muncul halaman yang menampilkan hasil pencarian.



Gambar 5. *Interface* awal

Halaman hasil pencarian pada aplikasi ini dapat dilihat pada Gambar 6. Pada halaman ini akan ditampilkan informasi judul artikel, bobot relevansi, penulis, nama jurnal ilmiah yang memuat, volume, isu, tahun, dan halaman untuk setiap baris hasil pencarian.



Gambar 6. *Interface* hasil pencarian

Pengguna dapat menyaring hasil pencarian dengan mencentang *checkbox* yang ada pada bagian kiri, maupun mengetikkan pada *textbox* di kanan atas. Pada bagian kiri disediakan penyaringan hasil pencarian untuk menyaring tahun, kategori, dan publikasi.

Bila pengguna mengklik judul artikel pada hasil pencarian, akan muncul halaman tentang artikel yang diklik. Pengguna dapat melakukan *advanced search* dengan mengklik tombol bertuliskan '*Try Advanced Search*'.

4.2 Pengujian Perbedaan Porter *Stemmer* dan New Porter *Stemmer*

Pengujian ini dilakukan untuk mencari perbedaan hasil kata dari algoritma Porter *Stemmer* dan *New* Porter *Stemmer*. Pada pengujian ini, akan dicari kata yang tidak berhasil disatukan/di*conflate* oleh Porter *Stemmer* tetapi berhasil disatukan/di*conflate* oleh *New* Porter *Stemmer*. Kumpulan kata yang digunakan dalam pengujian ini berasal dari kata-kata yang ada di kumpulan abstrak dari jurnal-jurnal Pusat Penelitian Universitas Kristen Petra.

Berikut beberapa kata yang tidak berhasil disatukan/di*conflate* oleh Porter *Stemmer*, dapat dilihat pada Tabel 1.

Tabel 1. Perbandingan Porter *Stemmer* dan *New* Porter *Stemmer*

Kata Awal	**Hasil dengan Porter**	**Hasil dengan New Porter**
Add/added/adding/adds	Add/add/ad/add	Ad/ad/ad/ad
Assurance/assure/ assured/assuredness/ assuring	Assur/assur/assur/ assured/assur	Assur/assur/ assur/assur/ assur
Based/bases/basis	Base/base/basi	Ba/ba/ba
Bias/biased/biases	Bia/bias/bias	Bia/bia/bia
Bought/buy/buying	Bought/bui/buye	Buy/buy/buy

Pada pengujian ini dapat disimpulkan bahwa *New* Porter *Stemmer* dapat menyatukan/meng*conflate* lebih banyak kata. Tetapi pengujian ini juga terbatas karena langkah dari *New* Porter

Stemmer tidak sepenuhnya dapat dilakukan karena keterbatasan data.

4.3 Pengujian Preparasi Data

Pengujian ini bertujuan untuk melihat apakah preparasi data yang dilakukan telah benar. Dalam pengujian ini akan digunakan tiga dokumen. Dokumen tersebut berisi:

- Dokumen 1: "*Shipment of gold damaged in a fire*"
- Dokumen 2: "*Delivery of silver arrived in a silver truck*"
- Dokumen 3: "*Shipment of gold arrived in a truck*"

4.3.1 Pengujian Proses Stopwords Removal

Pada langkah *stopwords removal* ini, seluruh kata yang termasuk dalam *stopwords* akan dihilangkan. Sehingga ketiga dokumen yang digunakan, akan berubah menjadi :

- Dokumen 1: "*Shipment gold damaged fire*"
- Dokumen 2: "*Delivery silver arrived silver truck*"
- Dokumen 3: "*Shipment gold arrived truck*"

4.3.2 Pengujian Proses Porter Stemming

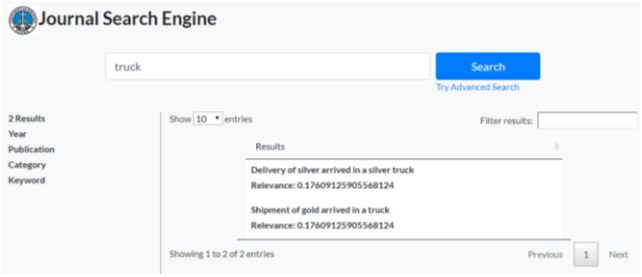
Pada langkah *stemming* ini, seluruh kata akan diubah ke bentuk *root word*. Sehingga ketiga dokumen yang digunakan, akan berubah menjadi :

- Dokumen 1: "*Shipment gold damag fire*"
- Dokumen 2: "*Deliveri silver arriv silver truck*"
- Dokumen 3: "*Shipment gold arriv truck*"

4.4 Pengujian Perhitungan TF-IDF

Pada pengujian ini, dilakukan perbandingan pengecekan TF-IDF antara sistem dan hitungan manual. Dokumen yang digunakan adalah ketiga dokumen pada pengujian preparasi data.

Perhitungan TF-IDF oleh sistem untuk kata 'truck' menampilkan dokumen 2 dan dokumen 3 dengan masing-masing memiliki bobot 0.17609125905568124. Seperti dapat dilihat pada Gambar 7.



Gambar 7. Hasil perhitungan TF-IDF oleh sistem untuk kata *'truck'*

Sedangkan perhitungan manual untuk kata truck yaitu :

- $IDF_{truck} = \log(3/2) = 0.1760912591$
- $TF\text{-}IDF_{truck,d1} = TF_{d1} * IDF_{truck} = 0*0.1760912591 = 0$
- $TF\text{-}IDF_{truck,d2} = TF_{d2} * IDF_{truck} = 1*0.1760912591 = 0.1760912591$
- $TF\text{-}IDF_{truck,d3} = TF_{d1} * IDF_{truck} = 1*0.1760912591 = 0.1760912591$

Dari pengujian ini kita ketahui bahwa perhitungan TF-IDF oleh sistem telah benar sesuai perhitungan manual.

5. KESIMPULAN

Berdasarkan hasil pengujian yang dilakukan pada sistem, maka dapat disimpulkan bahwa :

- *New* Porter *Stemmer* dapat menyatukan/meng*conflate* lebih banyak kata daripada Porter *Stemmer*.
- Aplikasi dapat menampilkan hasil pencarian yang cenderung relevan dengan satu *term* kata kunci, tetapi bila kata kunci merupakan *phrase*, maka hasil pencarian cenderung tidak relevan.
- Waktu kerja sistem ketika menyimpan hasil *indexing* ke basis data menggunakan REST memakan waktu yang lama dibandingkan dengan menyimpan hasil *indexing* ke *file*.
- Aplikasi mudah digunakan, sangat mudah dipahami, memiliki tampilan yang baik, informasi yang ditampilkan dan kesesuaian hasil pencarian telah memenuhi kebutuhan dengan baik.
- Aplikasi pencarian jurnal ilmiah dengan metode *term frequency-inverse document frequency* ini dapat menjawab kebutuhan mahasiswa maupun dosen dalam mencari jurnal.

6. DAFTAR PUSTAKA

[1] Beel, J., Gipp, B., Langer, S., & Breitinger, C. 2016, November. Research-Paper Recommender Systems: A Literature Survey. International Journal on Digital Libraries, 17(4), 305-338.

[2] Das, S. 2015, August 11. Beginners Guide to learn about Content Based Recommender Engines. Retrieved June 30, 2018, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/

[3] Febiyan, A. 2014, May 1. Apa Itu Composer. Dipetik September 30, 2018, dari DUMETschool: https://www.dumetschool.com/blog/Apa-Itu-Composer

[4] Halili, F., & Ramadani, E. 2018. Web Services: A Comparison of Soap and Rest Services. Modern Applied Science, 12(3), 175-183.

[5] Karaa, W. B. 2013, July. A New Stemmer to Improve Information Retrieval. International Journal of Network Security & Its Aplications, 5(4), 143-154.

[6] Luthfi, N. 2017, February 25. Bekenalan dengan Composer. Dipetik September 30, 2018, dari https://blog.nafies.id/composer/berkenalan-dengan-composer/

[7] Manning, C. D., Schütze, H., & Raghavan, P. 2008. Introduction to Information Retrieval. Retrieved June 6, 2018, from The Stanford Natural Language Processing Group: https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[8] Noviandha, F. D. 2017, February 15. Mengenal Slim Framework, Microframework Berbasis PHP. Dipetik September 30, 2018, dari CODEPOLITAN: https://www.codepolitan.com/mengenal-slim-framework-589f4003286cb

[9] Sinha, R., Khatkar, M., & Gupta, S. C. 2014, September. Design & Development of a REST based Web service platform for mobile applications integration on Cloud. IJISET - International Journal of Innovative Science, Engineering, & Technology, 1(7), 385-389.