ABSTRACT
Application that discussed in this paper is able to perform the process of finding web pages that have similar content to the url of the desired web page. Also developed an automated process for crawling web pages. This crawling process will continue since the process is activated. The search process begins by entering a url and web page url is obtained from the extract to get the key words that represent the web page. The keywords will be processed into a basic form using the Porter Stemmer algorithm. TF-IDF method used to obtain the importance of a keyword. Furthermore Jaccard Coefficient formula used to find similarity between web pages. Applications are limited to Web Page in English. Based on test results concluded that this application has worked well and can be utilized.

Keywords
Web Page Similarity, Crawler, TF-IDF, Porter Stemmer, Jaccard Coefficient, Keyword Extraction

1. INTRODUCTION
It has been widely available search engines to find data. No doubt, internet is the largest current source of data. Realizing these facts that the available data increases rapidly, then we can conclude tremendous potential to find data via the web. The web page has a variety of content, but sometimes there are web pages that discuss the same event.

The application will do a search on the web page that has similarities to other web pages. The similarity search based on the contents of the web pages.

2. SUPPORTING THEORY
The only way to collect URLs is to scan collected pages for hyperlinks to other pages that have not been collected yet. This is the basic principle of crawlers. They start from a given set of URLs, progressively fetch and scan them for new URLs (out-links), and then fetch these pages in turn, in an endless cycle. New URLs found thus represent potentially pending work for the crawler [3].

2.1 Automatic Keyphrase Extraction
We define automatic keyphrase extraction as the automatic selection of important, topical phrases from within the body of a document.

Many journals ask their authors to provide a list of keywords for their articles. We call these keyphrases, rather than keywords, because they are often phrases of two or more words, rather than single words. We define a keyphrase list as a short list of phrases (typically five to fifteen noun phrases) that capture the main topics discussed in a given document [7].

2.2 Stopword and Stemming
Stopword [3]: Most natural languages have so-called function words and connectives such as articles and prepositions that appear in a large number of documents and are typically of little use in pinpointing documents that satisfy a searcher's information need.

Stemming [3]: Device to help match a querry term with a morphological variant in the corpus. Common stemming methods use a combination of morphological analysis and dictionary lookup. Stemming ca increase the number of documents in the response, but may at times include irrelevant documents.

2.3 Porter Stemmer Algorithm
Here we present the Porter Stemmer algorithm (Suffix Stripping Algorithm) that we use in the application:

To present the suffix stripping algorithm in its entirety we will need a few definitions [5].

A 'consonant' in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. (The fact that the term 'consonant' is defined to some extent in terms of itself does not make it ambiguous.) So in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a 'vowel'.

A consonant will be denoted by c, a vowel by v. A list ccc... of length greater than 0 will be denoted by C, and a list vvv... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms:

CVCV ... C
CVCV ... V
VCVC ... C
VCVC ... V

These may all be represented by the single form

[C|VCVC ... [V]
where the square brackets denote arbitrary presence of their contents.

Using \((VC)^{m}\) to denote VC repeated \(m\) times, this may again be written as
\[
[C](VC)^{m}[V].
\]
\(m\) will be called the 'measure' of any word or word part when represented in this form. The case \(m = 0\) covers the null word.

The 'rules' for removing a suffix will be given in the form

(condition) \(S1 \rightarrow S2\)

This means that if a word ends with the suffix \(S1\), and the stem before \(S1\) satisfies the given condition, \(S1\) is replaced by \(S2\). The condition is usually given in terms of \(m\), e.g.

\((m > 1) \text{EMENT} \rightarrow\)

Here \(S1\) is 'EMENT' and \(S2\) is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which \(m = 2\). The 'condition' part may also contain the following:

*S - the stem ends with S (and similarly for the other letters).
*v - the stem contains a vowel.
*d - the stem ends with a double consonant (e.g. -TT, -SS).
*o - the stem ends cvc, where the second c is not \(W, X, Y\).

And the condition part may also contain expressions with 'and', 'or', and 'not', so that

\((m>1 \text{ and } (*S \text{ or } *T))\)

tests for a stem with \(m>1\) ending in \(S\) or \(T\), while

\((*d \text{ and not } (*L \text{ or } *S \text{ or } *Z))\)

tests for a stem ending with a double consonant other than \(L, S\) or \(Z\). Elaborate conditions like this are required only rarely.

In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching \(S1\) for the given word. For example, with

\[
\begin{align*}
\text{SSES} & \rightarrow \text{SS} \\
\text{IES} & \rightarrow \text{I} \\
\text{SS} & \rightarrow \text{SS} \\
\text{S} & \rightarrow \\
\end{align*}
\]

(here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for \(S1\). Equally CARESS maps to CARESS (\(S1='SS'\)) and CARES to CARE (\(S1='S'\)).

2.4 Term Frequency - Inverse Document Frequency

Tf-idf method is a way to give weight to the relationship of a word (term) of the document. This method combines the two concepts for calculating weights: frequency of occurrence of a word within a particular document and the inverse frequency of documents containing the word. Frequency of occurrence of the word in the document are given showing how important word in the document. Frequency of documents containing those words show how common the word [4].

The general formula for tf-idf:

\[
w_{ij} = tf \times idf
\]

\[
w_{ij} = tf_{ij} \times \log \frac{N}{n}
\]

Description:

\(w_{ij}\) = weight of the word / term \(t_j\) in the document

\(tf_{ij}\) = number of occurrences of the word / term \(t_j\) in the \(d_i\)

\(N\) = number of all documents in the database

\(n\) = number of documents containing the word / term \(t_j\)

Based on the above formula, regardless of the value of \(tf_{ij}\), if \(N = n\) then we will get the result \(0\) (zero) for the calculation of the IDF. It can be added to the value \(1\) in the idf, so the calculation of the weight to be as follows:

\[
w_{ij} = tf_{ij} \times (\log(N/n) + 1)
\]

In this paper the calculation of the tf will be replaced by the calculation method that we proposed in another paper [2].

2.5 Weight of Word / Term

Weight 1 (W1) is the frequency of words in a article. The number of same words in the article is calculated. The result will be divided by the total words in the article, by also considering the frequency of the words in the article [2].

Weight 2 (W2) is a value that is determined by the position of a first sentence that is used the word in a paragraph. In general, every paragraph in a good writing of an article usually only provides one main idea [2]. Because this application is used to process documents in English, then we use the formulation from Jonas and Araki to calculate W2, namely: \(W2 = \text{Early}(j) = 2\) if \(j < 10\) first sentence in a paragraph and 1 otherwise [6].

The calculation of the \(tf\) to be as follows:

\[
tf_{ij} = W1_{ij} \times W2_{ij}
\]

2.6 Jaccard Coefficient

The percentage of relevance covered by two sets is known as the Jaccard coefficient and is given by

\[
sim(q, d_j) = J(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]

\[
eq \frac{\sum_{k=1}^{n} W_{kq}W_{kj}}{\sum_{k=1}^{n} W_{kq}^2 + \sum_{k=1}^{n} W_{kj}^2 - \sum_{k=1}^{n} W_{kq}W_{kj}}
\]

This measure is fairly intuitive and often one of the more widely used measures when comparing IR systems. In a set theoretic
sense, the Jaccard measure signifies the degree of relevance covered by the union of two sets [1].

3. APPLICATION DESIGN
The design of the application can be seen in the flowchart in Figure 1 to Figure 5.
4. APPLICATION INTERFACES

The interfaces of the application are divided into two parts, namely:

a. Pages for the User:

On the main page (Figure 6) the user can enter a web page to search its similarity to another website. After pressing 'enter' or press the 'search' button then the application will find and display similar web pages. The Display results can be seen in Figure 7.

b. Pages for the Administrator:

On the administrator page (Figure 8), the admin can set the number of links in the search, the depth of crawling, what percentage of content in the English language for the web to be processed, the minimum similarity that is displayed, whether the position of words included in the calculations, and limits the location of a word included in the initial word in a paragraph. After doing all the settings, the administrator can enable the crawler and the similarity calculation process. Crawling and also the calculation results of similarity can be seen in Figure 9 and Figure 10.
5. TESTING

There are two kinds of experiments are performed, namely:

1. Testing of similarity ranking results from the application, which compared with the manually ranked by three sources and site Copyscape (http://www.copyscape.com) that offer similar services. List web pages that are tested and test results can be seen in Table 1 and Table 2.

| Table 1. List of Web Pages for The Testing |
|---|---|
| **Id** | **URL** |
| 1 | http://www.jbudge.com/story/fiction/bridge.htm |
| 2 | http://www.geocities.com/Athens/Acropolis/9343/bridge.htm |
| 3 | http://www.bebo.com/Chapters.jsp?ChapterId=3695291437&MemberId=3695253583 |
| 4 | http://www.strangeroad.com/Stories/Stories100.php |
| 5 | http://forum.gamenetworks.com/viewtopic.php?f=228&t=443&s t=0&sk=t&sd=a&start=490 |

| Table 2. Similarity Ranking Comparation |
|---|---|
| **Id** | **URL** | **Similarity Ranking** |
| **A** | **B** | **C** | **D** | **E** |
| 1 | - | - | - | - | - |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 2 | 3 | 2 | 2 | 2 |
| 4 | 3 | 4 | 3 | 4 | 3 |
| 5 | 4 | 6 | 7 | 6 | 7 |
| 6 | 5 | 5 | 5 | 7 | 5 |
| 7 | 6 | 2 | 4 | 3 | 4 |
| 8 | 7 | 7 | 6 | 5 | 6 |
| 9 | 8 | 8 | 8 | 8 | 8 |

Description:

*: URL id of the web that became the reference

A: The order of ranking the results of www.copyscape.com

B: The order of ranking results from Web Content Finder Application

C: The order of ranking respondent 1

D: The order of ranking respondent 2

E: The order of ranking respondent 3

From the test results can be seen that the ranking of the applications are not much different from the ranking produced by Copyscape site. But the result is quite different when compared to the manual ranking compiled by the respondents.

2. Testing the processing time of all calculations against the number of URLs that are processed. The test result can be seen in Figure 11.
From the test results of the processing time can be concluded that the more URLs that are processed then the longer the process.

6. CONCLUSION
From the comparison of the results of the ranking of URLs can be seen that the system can show good results. Because the results are not much different from the results of the ranking on a professional site that offers similar services. From the calculation speed of the process can be concluded that the application is ready to be implemented.

7. ACKNOWLEDGMENTS
Our thanks to Dr. Rolly Intan, who has helped us to get a better understanding of the research that he has done and also provide any important advice for the research we’ve done.

8. REFERENCES


