TOPIC-SPECIFIC WEB CRAWLER TO CREATE AUTOMATIC NARRATIVE EVOLUTION DATABASE

BY

HYE JIN YUN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2006

MASTER OF SCIENCE THESIS

OF

Hye Jin Yun

APPROVED:

Thesis Committee:

Major Professor ______________________________

____

____

Dean of the Graduate School ______________________________

UNIVERSITY OF RHODE ISLAND
2006

Abstract

Hypertext is a new generation of tools which allows a user to create interactive narratives on the World Wide Web. It is a young form of modern art and literature. Many artists, writers and programmers have performed various experiments to develop creative ways to build interactive stories on the web. Over the past few years, this genre of narrative has flourished, which brought into being online galleries that create permanent exhibits of the hypertext and interactive art.

Automatic Narrative Evolution (ANE) is a software environment which allows experiments using hybrids of human authorship, structural design, and machine writing. The purpose of this thesis is to extend ANE in order to allow text from the World Wide Web to be incorporated with narrative nodes created by human authors and thus providing a brand new way to evolve digital narratives. Furthermore, as the part of the implementation work for this thesis the ANE engine was redesigned and streamlined for the acquisition of web-based text.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Overview

The World Wide Web is the largest and most widely known hypertext data repository. Today, the web comprises billions of documents, images and multimedia data. It has also introduced many new genres in art and literature, and provided researchers with new challenges to find the best methodologies to index this dynamic search space. Hypertext, which is also called interactive textual art [8], is a new genre in modern literature and digital art.

Automatic Narrative Evolution (ANE) is "a tool to enable complex, unfixed temporal structures in digital narrative and facilitate the creation of works that are a hybrid of human authorship, structural design, and machine writing"[1]. The system displays a page that contains non-linear and interactive narratives. In addition, this page constantly rewrites itself in response to user interaction. It is unique in its form and visual/textual behavior [1]. ANE is unique from a technical perspective, since it applies evolutionary computation to hypertext literature. Evolutionary computation has been successfully applied to a number of areas – art and music, architectural design, engineering and drug discovery [1]. However, there has been very little work done in literature.

The system consists of two major pieces, a user interface that allows interaction with digital narratives and displays the rewritten page, and an engine on the back-end that constantly manipulates digital narratives based on constraints. The original architecture was designed to evolve digital narratives only from material written for the engine by a human author. However, these narratives can be generated from many different sources – scanning hard-copy documents or garnering off the web. “Foreign text” narrative nodes are text extracted from various data sources other than the text specially constructed by human authors for the ANE engine [1]. The foreign text acquisition was one of intriguing research directions due to the complexity of computing viable semantics for text so that the evolution engine can use this text in the actual construction of the evolving text page.

For this attempt to adopt foreign text, we need to choose a widely used open data source in order to develop a proper application for analyzing the text. The World Wide Web is a huge data repository that has increasing volume of data and information in numerous pages for public readers. However, even though it is such an enormous information resource, web content mining is not an easy task due to noisy and semi-structured data [4].

In this thesis, we present a search engine, Automatic Narrative Web Crawler (ANWC), which crawls the World Wide Web to collect foreign text narrative nodes for ANE, and a new architecture to incorporate this new component into the ANE engine. One interesting insight from this research is that an effective first-pass content analysis can be performed using efficient search expansion and simple heuristic rules to find quality content in noisy html documents.

1.2 Thesis Statement

The general purpose of this research is to construct a web crawler to collect data from the World Wide Web, analyze the quality of the collected data, and build a database for ANE. This database is used by ANE to further to evolve the digital narrative based on the foreign text narrative nodes.

The crawler performs these functions:

- Determine seed URLs to initialize the search using the Google API.
- Analyze the content of the downloaded HTML pages. If a page contains useful information, it will be saved for the next process. If it is not relevant, the content will be ignored.
- Parse the HTML content to get the next search links.
- Convert the HTML content to the ANE database format and deliver it to the database.

The architecture of the ANE engine is also redesigned to merge the new foreign text by the crawler with the existing digital narrative nodes written by human authors.

1.3 Thesis Outline

This thesis is organized as follows:

- The thesis begins with background information on web search engines, information retrieval, and evolutionary computation theory.
- Chapter 3 gives an overview of web search.
- Chapter 4 details the search expansion algorithm.
- Chapter 5 describes the design of the Automated Narrative Evolution environment, and a new architecture to adopt web crawler search result.
- Next, Chapter 6 discusses related research, including other systems that have used an evolutionary approach in inductive logic programming.
- The thesis concludes with achievements, future directions for research and closing remarks.

Chapter 2: Background

2.1 Web Content Mining

Web content mining aims to extract information and knowledge from Web page content [4]. The majority of documents in the World Wide Web are encoded in HTML. They are not well-structured and are hard to query [5]. The Web creates new challenges for information retrieval in many different ways [2]. Traditional information retrieval works with finite document collections. Documents are self-contained units, and are generally descriptive and truthful about their contents. In contrast, we are not able to collect a snapshot of the web because the Web is an indefinitely growing and shifting universe [12].

The amount of data on the web is huge and HTML contents are noisy. The styles and types of data are diverse - structured tables, texts, and multimedia data. In many cases, the HTML codes are nested. Due to the varying structure of web data, it is difficult to automate discovery of unexpected information in the web content as with traditional information retrieval. A single web page also contains a mixture of many kinds of information, e.g., the main content, advertisements, a navigation panel, copyright notices, etc. so that only a particular part of the content is useful, and the rest can be considered noise [4]. In many cases, multiple web pages are linked, and present the same or similar information. Those pages simply use different formats or syntaxes with the same information so that it is a challenging task to create a tool to automatically discover information [4].

The web is dynamic [4]. Information on the web constantly changes, and millions of new pages are created everyday. Keeping up with changes and updates from the same links is important because the same links can contain new information.

Unstructured text extraction has become a popular research topic with the expansion of the World Wide Web. The majority of current techniques are based on machine learning or natural language processing to learn extraction rules [5]. In contrary, the ANWC is a web crawler that looks for digital narrative nodes that can be used in ANE based on a set of simple heuristics. Chapter 3.2.3 introduces these simple heuristics to evaluate the quality of text for the ANE engine, and explains why the simplicity can be powerful in some complex problems.

Chapter 3: Crawling the World Wide Web

The World Wide Web is a search space that contains a massive amount of data. The amount of information on the web site is growing rapidly, as well as the number of internet hosting servers and text pages. Due to the rapid advance in technology and web proliferation, a web search engine can be designed in various ways.

For the Automatic Narrative Engine, we designed a small-scale content-analysis web search engine, **Automatic Narrative Web Crawler** (ANWC). The engine is not designed to index web sites or parse a large volume of HTML pages at a high speed like other large-scale engines such as Googlebot or Yahoo crawler. Instead of indexing and querying a large number of sites and links, the ANWC focuses on the content analysis and intelligent search expansion. The engine keeps a minimal search history and indexes of sites and URL paths to avoid visiting the same pages over and over. But it does not track times or frequencies of visits because the system does not require revisiting links or checking for updates to the HTML content from the same link. The ANWC focuses on the content analysis and converting the extracted text into the digital narrative.

3.1 Robot exclusion and user-agent

Search engines are popular application. Programmers design search crawlers with personal preferences to get the best search results from the web. However, if it is not designed correctly, the application can cause many problems for server applications. In particular, since a lot of sites analyze the number of unique visits from human users and their activities on the web, web crawlers can create unrealistic statistics or unnecessary traffic.

There are several guidelines for creating web crawlers, but they are not strict rules. Servers can deny any requests from web crawlers if they do not follow the standard protocol or have invalid HTTP header properties that could be problematic for the server applications. The ANWC follows two major rules: the robot exclusion standard and the HTTP header property, user-agents.

The standard of robot exclusion, which is commonly known as “robots.txt”, was proposed in 1994, as a mechanism for keeping robots out of some areas of the protected web resources such as, [6]:

- Resource-intensive URL spaces, e.g. dynamically generated pages.
- Documents which could represent a site unfavorably, e. g. bug archives.
- Documents which are not useful for world-wide indexing, e.g. local information or an intranet.

The robot exclusion protocol is a simple file robots.txt that is located in the root directory of the domain. The crawler can easily find this file by concatenating the domain name and “/robots.txt”. For example, the robot exclusion of the University of Rhode Island should be http://www.uri.edu/robots.txt. The file contains several keywords to indicate the names of allowed web crawlers and locations allowed and diallowed by tagging: User-agent, Allow and Disallow.

```
User-agent: *

Allow: /searchhistory/
Disallow: /search
Disallow: /groups
Disallow: /images
Disallow: /catalogs
Disallow: /catalogues
Disallow: /news
Disallow: /nwshp
Disallow: /?
Disallow: /addurl/image?
```

[Figure 1] The partial contents of robots.txt on http://www.google.com

Figure 1 is an actual example of robots.txt. The line "User-agent: *" means that this site allows all web crawlers. The site disallows multiple links to protect its resources. If the robots.txt file does not exist in the root folder of the domain, the crawler assumes that the site has no restrictions for crawlers to scan its web pages.

In addition to the robots exclusion protocol, there exists the robot META tag. The robot META tag is a method for the author of an html page to communicate with a web crawler. HTML authors specify restrictions in the html meta tag whether a document may be indexed, or used to harvest more links. No server administrator action is required [6]. For example,

```
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
```

This Meta tag specifies that the crawler must not index the page or follow links on the page. Whereas a server can deny any requests for disallowed paths in the robot exclusion list, there is no obvious way to enforce this META tag rule with a web crawler.

Another important rule is to provide a proper HTTP header to the server application. The Hypertext Transfer Protocol (HTTP) is an Internet Official Protocol Standard, and the World Wide Web Consortium (W3.org) specifies the syntax of the latest version, HTTP 1.1.

```
GET / HTTP/1.1
User-Agent: Mozilla/4.7 [en] (WinNT; I)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
image/png, */*
Host: www.yahoo.com
Accept-Encoding: gzip, deflate
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

[Figure 2] The sample HTTP client request to the server

The user-agent request-header field contains information about the user agent originating the request [7]. It contains various information about the user's browser. This header field is used for statistical purposes, tracking of protocol violations, and automated recognition of user agents for tailoring some responses to avoid particular user agent limitations [7]. Many sites use this header property field to measure web traffic generated by actual internet users. Since there are many automated applications that create web traffic, it is important for a site to check the user-agent and create the right statistics for their web traffic analysis. The user-agent of the ANWC is below:

User-Agent: Automatic Narrative Web Crawler 1.0

There are several organizations that manage lists of active web crawlers on the World Wide Web, and provide it to companies to help them create correct traffic analysis. The current version of ANWC is not registered yet since it is in beta.

3.2 Automatic Narrative Web Crawler System Design

The ANWC looks up topic specific resources in the World Wide Web, which means that the root search terms are initialized before it starts crawling. These initial terms create ten root links using the Google Web API. The crawler starts to scan documents from these initial seed URLs, and efficiently expands the search.

3.2.1 Search Terms to initialize search

Digital Narratives for Automatic Narrative Engine were initially designed by a human composer. He/she created a story and converted it to a proper format that the Automatic Narrative Engine could understand. The purpose of this research is to create a foreign text database that is able to evolve with existing or newly written digital narratives by a human author. It is important to choose initial search terms that can be blended well with the existing context.

The ANE creates a memory-resident hash-table database by importing a file written in the markup language when the system initializes. The language consists of three major tags: <section>, <node> and <sen>. Each node has a property, called meta, which is the collection of semantic words related to the context. The ANWC starts the initial search from this set of semantic keywords in the file.

Figure 3 shows a sample narrative node in the ANE markup language. This node consists of four sentences related to the semantic words, “incorrigible, wake, refrain, halt, June, begin”. For this node to be evolved with foreign text, we need to look up narrative nodes related to this keyword list on the web. The crawler starts to search the resources with the semantic words by reading this initial markup file and creating a list of seed URLs.

```
{\node \meta="incorrigible, wake, refrain, halt, June, begin"
  {\sen Will she disappear?}
  {\sen I said to you, "be careful. Today is a strange day, and
        that was the end of it."}
  {\sen I had written impassioned letters that expressed the
        urgency of my situation.}
  {\sen To hand to you the consecrated sum of your gifts, the
        secret you imparted persistently and without knowledge,
        these expressions of your will that lured, and, in a
        cumulative fashion, became a message.}
}
```

[Figure 3] Sample ANE markup language

Google is a leading search company that provides various web search services. Google Web API is a beta product that provides the search results by direct access without a web browser. The ANWC initiate the starting URLs for the given semantic words via the Google API. It initializes a set of URLs for the meta keywords from each node that have common relationship between the semantic words to crawl the World Wide Web.

$S = t_1 \cap t_2 \cap t_3 \cap \ldots\ldots\ t_{n-1} \cap t_n$ (S = Set of URLs, t_i = semantic word)

The database for the ANWC has a table to contain semantic keywords from the ANE markup file import. When the search engine starts, it checks this table to see if there are new semantic words and initializes the search root URLs for the new keywords. From a set of these root URLs, the crawler scans the HTML pages, analyzes the contents and collects all available links, which lead to the next level of search. All links are saved in the database, and the crawler selects the next link to search using the algorithm called “Hill-Climbing”.

3.2.2 Query HTML

HTML is a semi-structured data format. It is not easy to extract the data inside without proper lexical analysis. Figure 3-1 is an example of noisy HTML in a typical format. The ANWC extracts two types of data from HTML – text and URL links. Other types of data, images and videos, are not relevant for ANE data so that they are ignored in this step.

Even though ANE digital narratives require only text data, we found that extracting the whole HTML page as text only by eliminating all HTML tags was suboptimal due to the fact that the tags properly group text information and are helpful in predicting types of data not interesting for the evolution of narratives - menus, advertising copy or copyright notices at the bottom of the page. Therefore, we need to parse the HTML page without damaging the original HTML tag structure in order to keep details of the page.

```
<table border="0" cellpadding="0" cellspacing="0" width="612">
    <tr>
     <td width="612" valign="top" colspan="4">
     <table border="0" cellpadding="0" cellspacing="0" width="612">
          <tr>
          <td width="231" valign="top">
          <img border="0" src="home/newimages2/uritext.gif"
            alt="University of Rhode Island" width="231"
height="62"></td>
          <td width="384" valign="top"><map name="FPMap1">
<area href="home/help/" shape="rect" coords="287,2,332,25"
alt="Help">
<area href="home/services/" shape="rect" coords="1,3,51,28"
```

```
alt="Campuses">
<area href="home/campus/" shape="rect" coords="62,3,128,26"
      alt="Directories">
<area href="home/dir/" shape="rect" coords="141,0,210,23" alt="Fast
Links">
<area href="https://webmail.uri.edu/" shape="rect"
coords="221,1,276,27"
      alt="WebMail">
<area href="home/search/" shape="circle" coords="353, 25, 23"
      alt="Search"></map>
<img border="0" src="home/newimages2/top_bar_new.gif"
usemap="#FPMap1"
      width="379" height="50"></td>
          </tr>
        </table>
      </td>
    </tr>
</table>
```

[Figure 3-1] Semi-structured raw HTML

Because HTML is a very popular format for content, JDK 1.5 provides a stable HTML parser, HTMLEditorKit, as a part of the Java Swing Kit. To extract only certain tags by our rule, we restructure the raw HTML in a different data format that makes it easy to analyze. HTMLEditorKit converts an HTML page into two-dimensional array structure that represents a hierarchy of the nested tags. Figure 3-2 shows the converted form of HTML.

```
  .table[0].@cellpadding : 0
  .table[0].@border : 0
  .table[0].@cellspacing : 0
  .table[0].@width : 100%
  .table[0].tr[0].td[0].@width : 345
  .table[0].tr[0].td[0].a[0].@href : http://www.url.edu
  .table[0].tr[0].td[0].a[0].img[0].@height : 96
  .table[0].tr[0].td[0].a[0].img[0].@border : 0
  .table[0].tr[0].td[0].a[0].img[0].@width : 345
  .table[0].tr[0].td[0].a[0].img[0].@alt : Logotip de la Universitat Ramón Llull
  .table[0].tr[0].td[0].a[0].img[0].@src : http://www.url.edu/images/pagina1_01.jpg
  .table[0].tr[0].td[1].@style : background:url(images/pagina1_02.jpg)
  .table[0].tr[0].td[1].@valign : bottom
  .table[0].tr[0].td[3].@width : 84
  .table[0].tr[0].td[3].@style : background:url(images/pagina2_4.jpg)
  .table[0].tr[0].td[3].table[0].@cellpadding : 0
  .table[0].tr[0].td[3].table[0].@border : 0
  .table[0].tr[0].td[3].table[0].@cellspacing : 0
  .table[0].tr[0].td[3].table[0].tr[0].td[0].@width : 6
```

```
.table[0].tr[0].td[3].table[0].tr[0].td[1].@valign : top
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].@cellpadding : 0
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].@border : 0
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].@cellspacing : 0
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[0].td[0].@height : 25
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[0].td[0].@class : celda_mapa
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[0].td[0].@valign : top
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[0].td[0].a[0].@href :
http://www.url.edu/es/index.php
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[0].td[0].a[0].text[0] : Español
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[1].td[0].@height : 25
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[1].td[0].@class : celda_mapa
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[1].td[0].@valign : top
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[1].td[0].a[0].@href :
http://www.url.edu/en/index.php
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[1].td[0].a[0].text[0] : English
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].@height : 25
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].@class : celda_mapa
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].@valign : top
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].a[0].@target : _blank
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].a[0].@href :
http://www.url.edu/mapa.php
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].a[0].@id : lnkMapa
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[2].td[0].a[0].text[0] : Mapa Web
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[3].td[0].@valign : bottom
.table[0].tr[0].td[3].table[0].tr[0].td[1].table[0].tr[3].td[0].@class : adriadna
```

[Figure 3-2] Structured HTML converted by the HTML EditorKit in Java

This is a visual representation of the data. This transformed data format does not change any of the original HTML text. This HTMLEditorKit data set provides two advantages: firstly, it eliminates all white space from the HTML text so that we do not need to do additional lexical analysis to concatenate broken sentences inside of the table or paragraph tags. Secondly, in this way, it is also easy to follow the hierarchy of the nested tags so that the re-creation of the pure text can be properly performed. Each parsed <p> or <table> is considered one narrative, and evaluated by the heuristic rule.

The ANWC parses the HTML in two ways – link extraction and text extraction. The link extraction simply collects all URL links which are represented as “src=” or “href=” in source, anchor and frame tags. Since we do not need any multimedia resources, the ANWC does not keep the source links from image or any object-type tags. The ANWC extracts text data from <p> and <table> tags. One paragraph or table is considered as one single narrative node. These tags often used to graphically format the text on the browser, so they often contain other table or paragraph tags. The current version does not extract nested tables of paragraph separately. If the <table> tag contains another table tag inside, the parser considers that it is invalid format data and rejects them.

3.2.3 Stop words and Porter Stemming Algorithm

In information retrieval, analyzing any type of document is based on two sets of data: the kinds of semantic words the document contains and how many times they are present in the whole text. However, counting the frequency of all words in the text is not a good way to approach content analysis since there are many common words in English that do not carry inherent meaning themselves but modify other words, such as adverbs, conjunctions, prepositions, and forms of "be". Stop words and word variation, so-called stemming, are the most used techniques to retreive the proper semantic words from documents in information retrieval [9]. The list of stop words is attached in Appendix A.

Stop words are common words that are ignored by search engines at the time of searching a key phrase on both search query and found documents. In other search engines, this was done in order to save space for HTML documents and to accelerate the search process. The common words are often used interchangeably as "Filter words" because they are not usually key phrases in the content. Most search engines eliminate these words from the search query to avoid dilution of meta words. The ANWC uses the stop words in a different way than traditional search engines to evaluate the context. To create general English phrases, it is not possible to avoid using adverbs, conjunctions, prepositions, or forms of "be". Web content is very noisy since pages include advertising copy, menus, and copyright notices. The difference between these non-sentence phrases and paragraphs is that the phrases do not contain enough common words. The phrases are simple and sometimes contain only nouns, adjectives, symbols and numbers. In human linguistics, we cannot judge what percentage of common words is exactly used in each sentence or conversation. However, it is clear that if the sentence does not have a certain number of stop words, it is highly likely the text is invalid. The current engine checks for that 20% of the content are stop words in the context. If the found node has less stop words than that, the engine decides that the text in the node is not properly formed as a general English sentence.

Stemming is defined as a form of automatic truncation of each word in the index to its root. It is performed in order to accommodate the variety and ambiguity of the English language. For example, the words “searcher”, “searches”, “searched”, “searching” and so on can be stemmed as “search”. There are mainly two types of stemming: Plural and Porter stemming [9]. Plural stemming tries to determine the singular form of a word, whereas Porter stemming attempts to find the root, or stem, of a word. The ANWC uses the Porter stemming algorithm to count the accurate frequency of the stem words.

The Porter stemming algorithm was introduced by Martin F. Porter in 1980 [10]. Removing suffixes by automatic means is very useful in the field of information retrieval. In a typical information retrieval environment, a document can be represented by a vector of words, or terms. Terms with a common stem have usually similar meaning, for example [10]:

CONNECT

CONNECTED

CONNECTING

CONNECTION

CONNECTIONS

All words are related to a single term, CONNECT. The suffix stripping process reduces the total number of terms in the information retrieval system, and reduces the size and complexity of the data, which improves the performance.

The algorithm is simple, but requires several steps to find a complete stem word. Every word, or part of a word, has one of four forms:

CVCV ….C C = Consonant, V = Vowel

CVCV ….V A list of ccc… of length greater than 0 is still denoted

VCVC ….C by C, and V uses the same denotation.

VCVC ….C

These four forms can be represented by one single form,

[C]VCVC…. [V] → **[C]VC{*m*}[V]** *m*: the number of VC repeats

Some examples [10]:

m=0	TR, EE, TREE, Y, BY
m=1	TROUBLE, OATS, TREES, IVY
m=2	TROUBLES, PRIVATE, OATEN, ORRERY

The rule to remove a suffix is given in this form,

(condition) S1 → S2

This means that a word ends with S1 is replaced by S2 in the given condition. The condition contain the following including the value *m* which is usually used:

***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, or Y (e.g, –WIL, –HOP).

The condition also allows ***and***, ***or***, and ***not*** operations.

With the above rules, the algorithm follows total nine conversion steps:

Step 1a

	SSES	→	SS	caresses → caress
	IES	→	I	ponies → poni
	SS	→	SS	ties → ti
	S	→	(null)	cats → cat

Step 1b

(*m*>0)	EED	→	EE	feed → feed
(*v*)	ED	→	(null)	agreed → agree, plastered → plaster
(*v*)	ING	→	(null)	bled → bled, motoring → motor

The full description of this conversion step is attached as Appendix B. The Porter stemming algorithm is simple, but it reduces the complexity of calculating the vector of words list in the document. This algorithm is used twice in the overall Automatic Narrative Engine Architecture: the web content analysis in the ANWC and the evolutionary seed to mutate context in the Automatic Narrative front-end implementation, which will be described in Chapter 5, Collaboration with the Automatic Narrative Engine.

3.2.3 Simple Heuristics to evaluate content

A large volume of data analysis does not require finding all details in the data set. In data mining, too many constraints in the decision making process can create an underfitting rule. It is necessary to create frugal heuristics for large and noisy datasets [14]. As it was mentioned earlier, the web data are noisy and polluted with many types of information. Simple heuristic can eliminate useless information so that the web crawler can find the proper content in the HTML pages, but it also increases the performance of analysis to handle large volume of semi-structured data.

The ANWC looks for contents that are able to collaborate with existing digital nodes, which can blend with existing stories. The content must be a paragraph in proper English Grammar with a fixed length. Based on this fact, we could create several simple rules to evaluate the text from HTML.

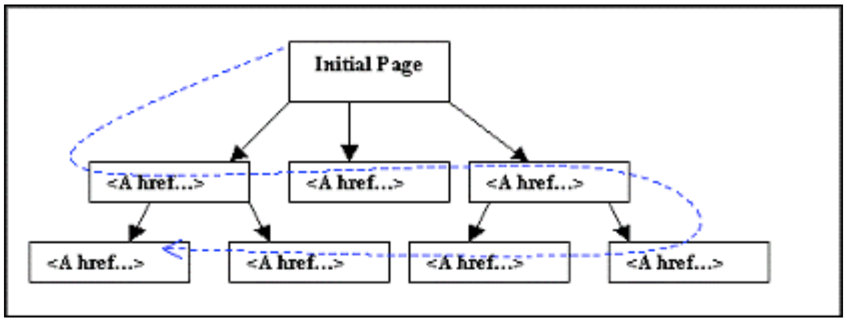
1. The number of keywords in the narrative node must be more than ten keywords.
2. The length of the node should be smaller than 8000 characters due to the size of display screen.
3. To be proper English, the paragraph must not contain any special characters such as like sharps, brackets, and etc. This rule eliminates the text like mathematical formulas and malformed HTML text that is displayed as broken text on a browser. It also filters out price lists from online stores.

4. The sentence must start with an uppercase character and end with a termination character like a period or quotation mark. This rule is effective for eliminating copyright notices, page menus and advertising copy.
5. Calculate the percent of stop words in the content. To become a readable English paragraph, the content must have a certain number of stop words. Requiring a higher percentage of stop words can eliminate too much useful information. Requiring too little can bring unreadable narrative nodes into the database. This value is one of the property settings on the crawler so that the performer of Automatic Narrative Engine can control the output from the crawler. The default value is twenty percent, which we find that it creates the best number of narrative nodes. In addition, this rule also evaluates text in foreign languages like French, German, or any other Asian languages.

Evaluating web content can be complicated. However, applying simple rules to perform evaluation can be powerful and robust in web content mining. Our five rules are simplistic, but they eliminate a lot of noise from the web content. We can easily find resources without expensive computation. The decision making process is complicated, but the fast and frugal heuristics can be a very powerful way to resolve issues.

Chapter 4: Search Technique by Optimization Algorithm

One of the main tasks in creating a web crawler is deciding how to expand the search intelligently. General HTML pages link to other web pages using the <a>, <frame>, or HTML source tags. A simple way to branch search links is to follow all links as they are found in order. Breadth-first search is a tree search algorithm that starts a search at the root node and traverses all the neighboring nodes. It then explores their nearest unexplored nodes, and repeat this process until it finds a goal. By using this mechanism, the crawler can visit all links found while scanning HTML documents and follow them in order. However, this is an expensive method because the World Wide Web changes every day and millions of web pages link to one another. It is also likely to visit pages that do not provide good content. This traditional tree structure is not valid to describe the web.



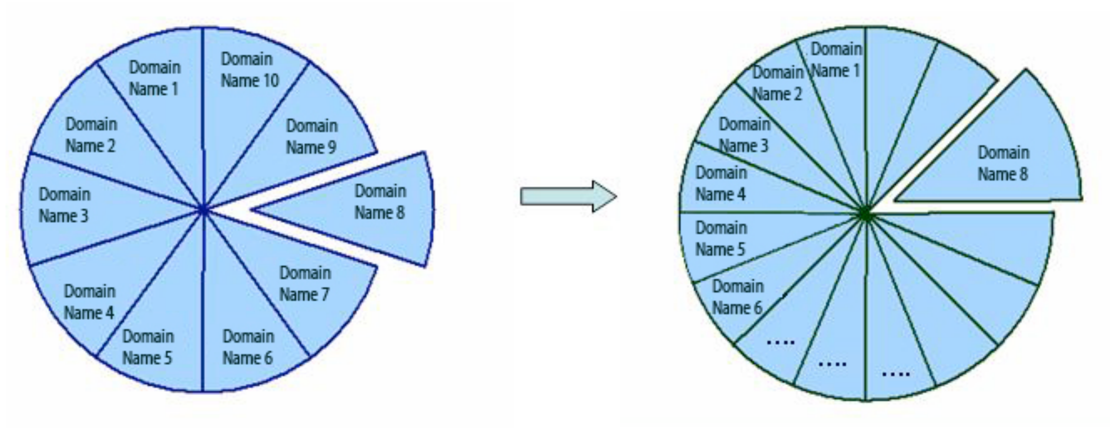
[Figure 4] Breadth-First Search

Hill Climbing algorithm is a depth-first search with a heuristic cost analysis. It sorts a new path by the estimated distances between its terminal nodes and the goal [20], and the next destination is selected by the local maximum [19]. This approach fits our need because the crawler has to manage a massive amount of URL links and pick a link that can provide the best result set. But links that have a small chance of providing good content still need to be picked in a random fashion because they might lead to interesting content in the long run.

4.1 Algorithms to implement intelligent search expansion

The ANWC is a web crawler to find topic-specific web resources. All saved web resources are scored when they are saved into the database. Hill climbing is a search algorithm with a heuristic measurement that orders choices when nodes are expanded [20]. It finds a local maximum value among the heuristic values of the linked node. Roulette-wheel selection, which is also known as fitness-proportionate selection, is a genetic operator used in genetic algorithms for selecting potential solutions for recombination [15]. The search expansion was implemented using the hill climbing algorithm to find the local maximum scored site, but it borrowed the model of the roulette-wheel selection evolutionary computation.

By the set of semantic keywords, the ANWC initializes ten URLs as the search seed using the Google Web API. The number of the seed URLs is set by the ANWC configuration file so that it can be overwritten by the operator's preference. The URLs are saved into two pieces, Domain Name and the Link Path. When the search starts, these ten initial domain names have the default score 1, so that they have the same size of pie on the wheel. All domains have an equal opportunity to be chosen on the first roll. When one of the domain names is chosen, the crawler get a URL path belong to the selected domain, downloads the HTML page from the link and analyzes the content. All links listed on the page by <a>, <frame> or other source tags are saved into two separate tables, Domain Name and the Link Path, like the search seeds. The text data are analyzed by the heuristic rule specified in Chapter 3.2.3, Simple Heuristics to evaluate content. After this analysis, the site gets scored by the rule whether the content was valuable or not. On the second roll, there are more than ten initial domains on the wheel because of the links collected from the search. The new domain names will have the default score 1. If the previous link contained valuable content and got scored, this domain has a slightly bigger slice of the pie than other sites on the wheel with the score 2. This will provide a higher priority to the domain on the next random selection. Other domain names still have an opportunity to be selected on the next roll with the default score 1. If the link did not get scored, all domain names have the same score again and share the same slice of pie on the wheel so that all of domains have an equal opportunity to be chosen. After running the algorithm many times, the crawler collects many domain names and the wheel will be sliced into many small pieces. But the algorithm will always ensure the correct possibility for each site to be selected on the next search because the algorithm does not select a site by the global maximum value. Figure 5 provides a graphical image of the mechanism.



[Figure 5] All domain names have an equal size of pie on the wheel. If the crawler finds a good quality content from the selected domain 8, its piece will take a bigger slice of the pie on the next roll. This means that the domain name 8 will have a better chance than others.

We achieved two goals. First, the algorithm provides a better chance to be chosen for a site that provides good quality content. The more the domain site provides good content, the bigger slice of the pie will be assigned on the wheel. This increases the chance for the crawler to collect good web resource.

Secondly, the low score site still has a chance to be searched in a random fashion. The ANWC gives a higher priority to the high scored domain, but the low scored domain still has a possibility to be selected. The algorithm properly assigns chances for high and low score sites.

```
Procedure search-expansion (semantic_keywords)

   Initialize the search seeds(semantic_keywords);
   Initialize domain names from the search seeds;
   Score the default value to the seed domain names;

   Do while (true)
    Int total_score = get_total_score(all domains);
    Int random_number = get_random_number(total_score);
    Int domain_id = get_domain_id(random_number);
    URL link = get_next_link_path_of_the_domain(domain_id);

    Search_and_analyze(link);

    Int number_of_found_narrative_node=Search_search(link);
    If (number_of_found_narrative_node>0)
          Add_score_to_the_domain(domain_id);
   End while

End Procedure
```

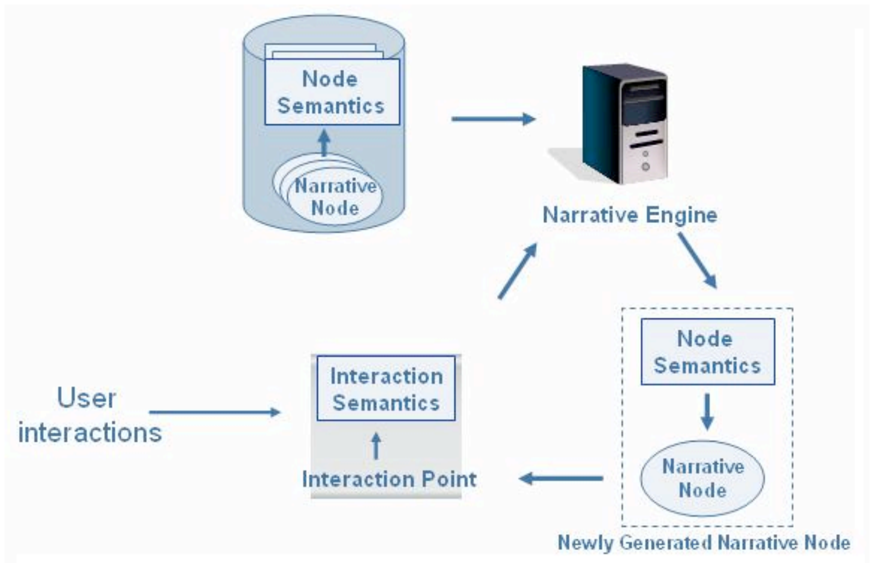
[Figure 6] Pseudo-code to expand the search

The implementation of this algorithm uses a random number generation to simulate the wheel. First, it gets the total score of all domains names in the database. It generates a random number from this total value, and finds the site where to random number belonged. Since all domain names have unique IDs, it is easy find where the random number belongs. Figure 6 shows the pseudo-code of the algorithm. The initial ten domain names on the first roll or any new domains added later are scored the default value, 1. When the search is expanded, they will have different values.

Chapter 5: Incorporation with Automatic Narrative Engine

Automatic Narrative Engine is a tool to experiment with a hybrid of human authorship, structural design, and machine writing. Automatic Narrative Engine (ANE) collects unfixed temporal structures in digital narrative and re-creates a work of art [1]. In ANE, digital narrative is created by assembling a set of narrative nodes. Each narrative node is a fragment of text that is denoted semantically by a set of keywords, and also contains a set of interactive points to drive the digital narrative through user interactions. When a user clicks on an interaction point in the digital narrative, the system interprets it as a constraint and adapts the digital narrative. During this adaptation, an existing narrative node is replaced with a new narrative node that is better qualified under the current constraints [1].

When the ANE system is first launched, the user is presented with a page of text. The initial text layout is designed by the composer of the evolving text. Within the page, any words can act as interaction points for the user. Evolution of the narrative is driven by either highlighting or double-clicking one of the words. The system interprets this as a constraint and adapts a new narrative to this constraint. Adaptation takes the form of swapping out a segment of text and replacing it with new text deemed more appropriate given the current constraint [1]. Figure7 shows the steps of the evolution by user interaction. Given a set of narrative nodes with their semantics and an interaction point with its semantics or constraints, the narrative evolution engine attempts to find the most appropriate narrative node based on the match of semantics against constraints. The best node is selected and its syntax is incorporated into the evolving text page.



[Figure 7] Automatic Narrative Engine Evolutionary Steps by user interactions

The first implementation of ANE was designed to import semantic nodes from a file in structured markup language. The system read a data file, parsed it, and kept narrative nodes in memory. With the adoption of foreign narrative nodes from the crawler, the original architecture of ANE needed to be modified to combine digital narrative nodes from two sources. The system also required a persistent database rather than memory resident hash tables because the amount of data collected by the web crawler grows more quickly.

5.1 The Markup Language and the Database Schema

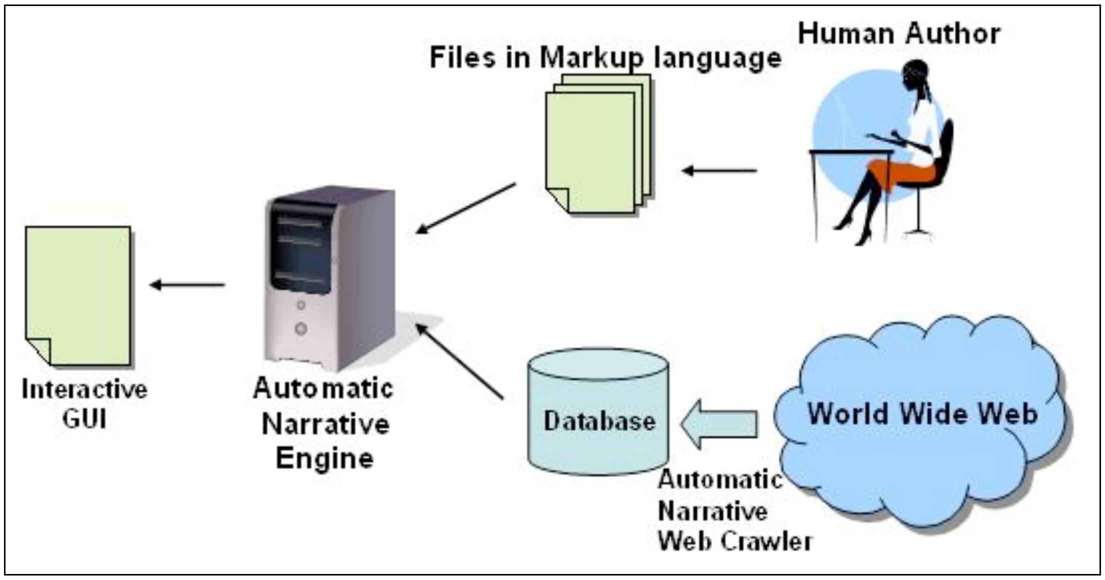
The markup language consists of three major tags: “section”, “node” and “sen”. One section represents one or more narrative nodes. Narrative nodes can be paragraphs, simple collections of sentences, a single sentence or just utterances of several words. Figure 8 shows an example of a section in the markup language. This set of data is created by a human author, and the Automatic Narrative Engine imports it when the engine starts.

```
{\section \display \id="begin"

    {\node  \meta="none, recited, start, begin"
    {\sen I am revisiting a reading error.}
    {\sen The machine demands language.}
    {\sen You have to get the ball rolling.}
    {\sen I know: it has always been easier to be on the receiving
    end.}
    }

    {\node  \meta="vision, blindness"
    {\sen But what could this new loss of vision possibly reveal to
    you about yourself.}
    }
}
```

[Figure 8] ANE Markup language – full section example

However, the ANWC collects a large number of narrative nodes from the World Wide Web so that the solution to keep digital nodes in a file is not relevant. We need a persistent database to handle and index a large amount of data.

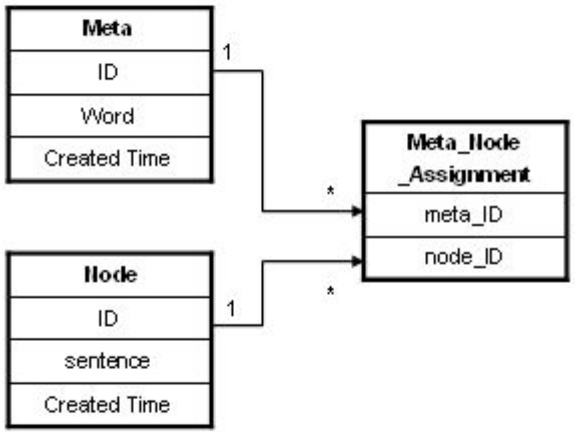
MySQL is a popular open source database. It is lightweight and provides reliable service. Since the ANE reads digital narrative nodes from two sources, it was necessary to use an efficient database to store collected narrative nodes from the World Wide Web. The initial architecture of ANE read narrative nodes from a file when the engine started. The engine also needed to be modified to read additional nodes from a database.



[Figure 9] Automatic Narrative Engine architecture with web crawler

The data evaluated by the heuristic rules consist of one or multiple sentences. By the definition of the markup language, one narrative node contains one or more than one sentence [1]. However, the actual interactive page evolves in nodes. It does not evolve or mutate sentences in each node. The ANWC considers that one node contains one sentence to simplify the conversion process and avoid unnecessary parsing of the narrative node. The meta words for the narrative node are the search keywords where the page was found.

Figure 10 shows the data relationship between meta words and narrative nodes. The meta words and narrative nodes have a many-to-many relationship. This drives the evolution that one constraint, one meta keyword, is associated with many nodes so that an interacted node can be replaced by another node that has the constraint. In the same manner, one node can have multiple constraints that can increase the chance to be chosen as in the example in the Figure 8.



[Figure 10] Database Schema for narrative node and meta words relationship

This database schema can be expanded to adopt narratives in a file. In that way, the engine does not have to import the data file every time when the system initializes, which increases the performance of the system at initialization time. However, since direct file import to the database requires an additional tool and requires another set of research expertise, it will be added in the future.

5.2 Content Evolution with foreign text

The digital narratives evolve by user interactions like clicking or highlighting in the ANE. The system interprets user interaction as a constraint and adopts a new narrative node linked to the constraint. To find a proper narrative node, the constraint, the chosen keyword, is prefixed by the Porter stemming algorithm. The system looks up a narrative node related to this prefixed keyword, and adopts a new node. Adaptation replaces the existing node, given the current constraint, with the new one [1]. Since the system has started to adopt foreign text from the crawler, it is necessary to redesign this architecture in part.

The original design adopted only digital narrative node designed by a human author. The system created an unfixed open-ended story, but the narrative nodes from the file were designed to be related to author's intent. Adopting random foreign text from the Internet is a challenge because the text may contain several random factors to prevent text evolution. It is hard to guarantee robustness. The new Automatic Narrative Engine has one more property configuration to resolve this issue. The initial configuration file contains a value, proportion_of_crawler_node, which controls the mix of narrative nodes from the human author and the web crawler. In this way, we can perform various evaluation experiments on the foreign text.

Narrative nodes from the World Wide Web are diverse. They have texts for politics, sports, entertainments, and sometimes advertising copy. For example, the human author can write limited narrative nodes with a meta word, "cookie". In contrast, the web crawler can collect broad knowledge of cookies in computers, recipes, history, and even product descriptions. This kind of randomness creates an interesting phase of text evolution in the interaction page of the Automatic Narrative Evolution.

Chapter 6: Related Research

6.1 Hypertext Literature

Hypertext literature, which is also called as interactive textual art, is new but it is treated as a single genre. It is variously called hypertext, hypermedia, cybertext, interactive fiction, and terms which are still being invented [8]. The World Wide Web has naturalized the clicking on a link as a basic form of interaction to switch documents. This type of interaction created another form of entertainment, which provides an immediate change in sight. Hypertext is non-sequential writing, which is a presentation of text by a linked network of nodes which readers are free to navigate them in a non-linear fashion. Writers provide stories with anchor tags, and readers re-create a story by interacting with them.

To define this genre for literature purposes, hypertext is non-linear, unbound, and unfixed. It is a non-linear text that allows multiple paths for users to sequence the reading. Stories are not bounded like books that are held together between covers. They are open-ended. The node can loop back onto itself, link out into different information in the global network, or repeat another hypertext world that it has no final reading of the text. The story does not require any definite reading, which means that there is no "fixed" text.

The Electronic Poet Center at the University at Buffalo is a center of this movement. They list hundreds of writers and digital artists who work in this new area. There are many compelling works from these artists. Jörg Piringer is an artist and programmer, and he created a software package, *rimmixa*, which reads digital poetry audibly in response to user interaction [16]. John Pierre Balpe and the groupe@graphe, created the hypertext novel *ebbflux* [1]. John Cayley creates poetic works, *Lense*, in which the text undergoes animated transformations [17]. Gavin Steward is a digital media artist who actively works on many digital poetry and cybertext projects. Digital images of all his works are available on http://www.gavinsteward.net.

6.2 Ontology in Computer Science and Semantic Web

An ontology is a conception of reality. In computer science, the term ontology refers to a data model that represents a specific part of the realworld. It explains abstract representations of objects and their relationships [17]. Ontologies are commonly used in artificial intelligence, knowledge representation, inductive reasoning, classification, and a variety of problem solving techniques, as well as to facilitate communication and sharing of information between different systems [17].

An ontology is a major piece of the Semantic Web framework. The Semantic Web intends to create a universal space, the World Wide Web, for information exchange by given semantics in a manner understandable by machines [18]. The current World Wide Web is based primarily on documents written in HTML, which has limited ability to classify or organize different types of documents. The Semantic Web consists of standards and tools of XML, RDF and OWL. These technologies are also combined in order to provide a better way of providing the content of Web documents. Therefore, the content can be formed in Web-accessible databases, or as its own markup within documents. The machine-readable descriptions enable automatic search and easier information gathering by computers. The World Wide

Web Consortium organizes the project of the Semantic Web, and maintains standards to share machine-understandable information.

Chapter 7: Conclusion

The Automatic Narrative Web Crawler is a topic-specific web search engine. It focuses on intelligent search expansion and content analysis to create the best results in search spaces given appropriate problem constraints. We modified the original hill climbing algorithm to expand the search using a model of roulette-wheel algorithm introducing some randomness in the search strategy. This could be interpreted as a exploration versus exploitation tradeoff. The simple heuristics were frugal and powerful solutions to evaluate a massive amount of noisy data. The new algorithm and the heuristics together created an optimized search analysis to find the best digital narrative nodes in the World Wide Web.

The Automatic Narrative Engine is a software environment to experiment with self-evolved text. It was initially designed to evolve only digital narrative by a human author, but this research enhanced the system to adopt foreign nodes collected by the web crawler. As a result, the text in the Automatic Narrative Engine can evolve from the richer database as was proposed. In addition, the foreign narratives are collected from various sites and sources, so they add more random factors to mutate texts.

The current system evolves one single hypertext by many different users. Even though user interaction is a main key of the evolution, the system does not record the evolutionary trace. Perhaps, this is the next step for the Automatic Narrative Engine since the evolutionary trace by the interacted semantics can be an important factor to differentiate our work from other hypertext systems.

List of References

[1] Lutz Hamel, Judd Morrissey and Lori Talley: *Automated Narrative Evolution,* A White Paper, http://www.errorengine.org (2004)

[2] Sergey Brin, Lawrence Page, Stanford University, *The anatomy of a large-scale hypertextual web search engine,* http://www-db.stanford.edu/~backrub/google.html (1998)

[3] Charu C. Aggarwal, Fatima Al-Garawi, Philip S. Yu, IBM T. J. atson Research Center: *Intelligent Crawling on the World Wide Web with Arbitrary Predicates* (2001)

[4] Bing Liu, Kevin Chen-Chuan Chang, Department of Computer Science, University of Illinois at Urbanan-Campaign: *Special issue on web content mining,* SIGKDD Explorations (2005)

[5] Ruth Yuee Zhang, Laks V.S LAkshmanan, Ruben H. Zamar, Department of Statistics, University of British Columbia, Vancouver: *Extracting Relational Data from HTML Repository,* SIGKDD Explorations (2005)

[6] Martijn Koster, http://www.robotstxt.org/*, Evaluation of the Standard for Robots Exclusion* (1996)

[7] World Wide Web Consortium, http://www.w3.org/Protocols/rfc2616/ rfc2616.html, *Hypertext Transfer Protocol -- HTTP/1.1* (2005)

[8] Dean Taciuch, http://mason.gmu.edu/%7Edtaciuch/htext_lecture*, Hypertext and Literary Form* (2005)

[9] Search Engine Optimization Ethics*, Stop Words***,**
http://www.searchengineethics.com/ stopwords.htm (2002)

[10] M. F. Porter*, An algorithm for suffix stripping,* http://www.tartarus/org/martin/ porterstemmer/ (1980)

[11] *Chapter 1. Evolutionary Computation, Creative Evolutionary System,* Morgan Kaufmann (2005)

[12] *Chapter 7. Social Network Analysis, Mining the web, discovering knowledge from Hypertext Data,* Morgan Kaufmann (2005)

[13] *Evolutionary Computation,* http://en.wikipedia.org/wiki/Evolutionary_ Computation, Wikipedia, (2005)

[14] Gerd Gigerenzer, Peter M. Todd, and the ABC Research Group: Chapter 1, *Fast and frugal heuristics: The adaptive toolbox, Simple Heuristics that Make Us Smart,* Oxford University Press (1999)

[15] Wikipedia (http://en.wikipedia.org/wiki/), *Roulette-Wheel Algorithm,* http://en.wikipedia.org/wiki/Roulette_wheel_ selection (1999)

[16] Jörg Piringer, *digital sound visual interactive poetry etc.,* http://joerg.piringer.net/ (2006)

[17] John Cayle, *Lense*, http://homepage.mac.com/shadoof/lens/lens.html, (2005)

[18] Wikipedia (http://en.wikipedia.org/wiki/), *Ontology in Computer Science*, http://en.wikipedia.org/wiki/Ontology_%28computer_science%29, (1999)

[19] Wikipedia (http://en.wikipedia.org/wiki/), *Semantic Web,* http://en.wikipedia.org/wiki/Semantic_Web(1999)

[20] Patrick Henry Winston, Chapter 4. Nets and Basic Search, *Artificial Intelligence*, Third Edition, (1992)

APPENDIX A.

Stop Word List

PRONOUNS FORMS				
I	me	my	myself	
we	us	our	ours	Ourselves
you	you	your	yours	yourself
he	him	His	himself	yourselves
she	her	hers	herself	
they	them	their	theirs	themselves
what	which	who	whom	
this	that	these	those	
VERB FORMS (using F.R. Palmer's nomenclature)				
am	is	was		
are	were			
be	been	being		
have	has	had	having	
do	does	did	doing	
ARTICLES				
a	an	the		

prepositions, conjunctions, adverbs etc				
and	but	if	or	Because
As	until	while	of	At
by	For	with	about	Against
between	into	through	during	Before
after	above	below	to	From
up	down	in	out	On
off	over	under	again	Further
then	once	here	there	When
where	why	what	how	All
any	both	Each	few	More
most	other	some	such	No
Nor	not	only	own	Same
So	than	too	very	
Common English Words to eliminate				
One	every	least	less	Many
Now	ever	never	say	Says
said	also	go	goes	Went
just	make	made	put	See
seen	whether	like	well	Back
Even	still	way	take	Since
Another	However	two	three	Five
first	second	New	old	High
long				

APPENDIX B

Porter Stemmer Algorithm Steps

All definitions, denotations and examples in this section are originated from the white paper in the official site of porter stemming algorithm, http://www.tartarus.org/martin/PorterStemmer.

Definitions and Conditions,

C = Consonant, V = Vowel

[C]VCVC…. [V] → [C]VC{m}[V] m: the number of VC repeats

***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, or Y (e.g, –WIL, –HOP).

Rules

(condition) S1 → S2

Steps

- **Step1a**

SSES	→	SS	caresses → caress
IES	→	I	ponies → poni
SS	→	SS	ties → ti
S	→	(null)	cats → cat

- **Step1b - a**

(*m*>0)	EED	→	EE	feed → feed
(*v*)	ED	→	(null)	agreed → agree, plastered → plaster
(*v*)	ING	→	(null)	bled → bled, motoring → motor

- **Step1b - b**

AT	→	ATE	conflate(ed) → conflate
BL	→	BLE	trouble(ed) → trouble
IZ	→	IZE	siz(ed) → size

(*d and not (*L or *S or *Z)) → single letter	
	hopp(ing) → hop
	tann(ed) → hop
	fall(ing) → fall
	hiss(ing) → hiss
	fizz(ed) → fizz
(m=1 and *o) → E	fail(ing) → fail
	fil(ing) → file

- **Step1c**

(*v*) Y → I	happy → happi
	sky → sky

- **Step2**

(*m*>0)	ATIONAL→	ATE	relational → relate
(*m* >0)	TIONAL→	TION	conditional → condition, rational → ration

(*m* >0) ENCI →	ENCE	valenci → valence
(*m* >0) ANCI →	ANCE	hesitanci → hesitance
(*m* >0) IZER →	IZE	digitizer → digitize
(*m* >0) ABLI →	ABLE	comformabli → comformable
(*m* >0) ALLI →	AL	radicalli → radical
(*m* >0) ENTLI→	ENT	differentli → different
(*m* >0) ELI →	E	vileli → vile
(*m* >0) OUSLI→	OUS	analogously → analogous
(*m* >0) IZATION→	IZE	vietnamization → vietnamize
(*m* >0) ATION→	ATE	predication → predicate
(*m* >0) ATOR →	ATE	operator → operate
(*m* >0) ALISM→	AL	feudalism → feudal
(*m* >0) IVENESS→	IVE	decisiveness → decisive
(*m* >0) FULNESS→	FUL	hopefulness → hopeful
(*m* >0) OUSNESS→	OUS	callousness → callous
(*m* >0) ALITI →	AL	formaliti → formality
(*m* >0) IVITI →	IVE	sensitiviti → sensitive
(*m* >0) BILITI→	BLE	sensibiliti → sensible

- **Step3**

(*m* >0) ICATE→	IC	triplicate → triplic
(*m* >0) ATIVE→	(null)	formative → form
(*m* >0) ALIZE→	AL	formalize → formal
(*m* >0) ICITI →	IC	electriciti → electric

(m >0) ICAL	→	IC	electrical → electric
(m >0) FUL	→	(null)	hopeful → hope
(m >0) NESS	→	(null)	goodness → good

- **Step4**

(m >1) AL	→	(null)	revival→ reviv
(m >1) ANCE	→	(null)	allowance → allow
(m >1) ENCE	→	(null)	inference → infer
(m >1) ER	→	(null)	airliner → airline
(m >1) IC	→	(null)	gyroscopic → gyroscop
(m >1) ABLE	→	(null)	adjustable → adjust
(m >1) IBLE	→	(null)	defensible → defins
(m >1) ANT	→	(null)	irritant → irrit
(m>1) EMENT	→	(null)	replacement → replac
(m >1) MENT	→	(null)	adjustment → adjust
(m >1 and (*S or *T)) ION	→	(null)	adoption → adopt
(m >1) OU	→	(null)	homologou → homolog
(m >1) ISM	→	(null)	communism → commun
(m >1) ATE	→	(null)	activate → active
(m >1) ITI	→	(null)	angulariti → angular
(m >1) OUS	→	(null)	homologous → homolog
(m >1) IVE	→	(null)	effective → effect
(m >1) IZE	→	(null)	bowdlerize → bowdler

- **Step5a**

(m >1) E	→	(null)	probate → probat, rate → rate
(m -1 and not *o)E→		(null)	cease → ceas

- **Step5b**

  (m >1 and *d and *L) → single letter

controll	→	control
roll	→	roll

Bibliography

Aggarwal, Charu C., Fatima Al-Garawi, and Philip S. Yu, IBM T. J. Watson Research Center: *Intelligent Crawling on the World Wide Web with Arbitrary Predicates,* 2001.

Bentley, Peter J., and David W. Corner, *Creative Evolutionary Systems,* Morgan Kaufmann, 2002.

Brin Sergey, and Lawrence Page, Stanford University*, The anatomy of a large-scale hypertextual web search engine*, 1998.

Cayle, John, *Lense,* http://homepage.mac.com/shadoof/lens/lens.html, 2005.

Chakrabarti , Soumen*, Mining the web, discovering knowledge from Hypertext Data,* Morgan Kaufmann, 2005.

Carriere, Jeromy, Nortel Ottawa, ON, and Rick Kazman, Software Engineering Institute Pittsburgh, PA, 15213*, WebQuery:Searching and Visualizing the Web through Connectivity,* http://www.cgl.uwaterloo.ca/Projects/Vanish/webquery-1.html.

Dong, Xing, Jayant Madhavan, and Alon Halevy, University of Washington*, Mining Structures for Semantics,* SIGKDD Explorations, 2005.

Electronic Poetry Center (http://epc.buffalo.edu), *E-Poetry,* http://epc.buffalo.edu/e-poetry/.

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung, Google Lab*, The Google File System,* http://labs.google.com/papers/gfs.html, 2003.

Gigerenzer, Gerd, Peter M. Todd, and the ABC Research Group: *Simple Heuristics that Make Us Smart, Oxford University Press,* 1999.

Gruhl, D., R.Guha, David Liben-Nowell, and A. Tomkins, IBM Research, Almaden, *Information Diffusion through Blogspace,* SIGKDD Explorations,2005

Hamel, Lutz, Judd Morrissey and Lori Talley: *Automated Narrative Evolution,* A White Paper, 2004.

Koster, Martijn, http://www.robotstxt.org/, *Evaluation of the Standard for Robots Exclusion,* 1996.

Kraft, Reiner, and Jason Zien, IBM Almaden Research Center, San Jose, CA, *Mining Anchor Text for Query Refinement,* 2005.

Liu, Bing, and Kevin Chen-Chuan Chang, Department of Computer Science, University of Illinois at Urbanan-Campaign*, Editorial: Special issue on web content mining,* SIGKDD Explorations, 2005.

Lonton, Tony, *Web Content Mining with Java,* Willy, 2002.

MIT, World Wide Web Consortium, http://www.w3.org/Protocols/rfc2616/rfc2616.html, Hypertext Transfer Protocol -- HTTP/1.1, 2006 March 20th**.**

Martynov, Maxim, and Boris Novikov, University of St.-Petersburg, Russia*, An indexing algorithm for text retrieval,* 1996.

Piringer , Jörg, *digital sound visual interactive poetry etc.,* http://joerg.piringer.net/

Porter, M. F., *An algorithm for suffix stripping,* http://www.tartarus/org/martin/ porterstemmer/, 1980.

Raghavan, Sriram, and Hector Garcia-Molina**,** *Crawling the Hidden Web,* Computer Science Department, Stanford, CA94305 USA, 2001.

Search Engine Optimization Ethics, *Stop Words,* http://www.searchengineethics.com/ stopwords.htm, 2002.

Smith, Lesley, *Hypertext and Hypermedia Bibliography,* http://osf1.gmu.edu/%7Elsmithg/htextbiblio.htm, 2004.

Song, R., Haifeng Lui[1], Ji-Rong Wen, and Wei-Ying Ma, Microsoft Research Asia, 49 Zhichun Road, Beijing, 100080, P. R. China, [1]Department of Computer Science, University of Toronto, On, Canada, *Learning Important Models for Web Page Blcoks based on Layout and Content Analysis,* SIGKDD Exploration, 2005.

Taciuch Dean, George Mason University, *Hypertext and Hypermedia Resources*, http://mason.gmu.edu/%7Edtaciuch/medialinks.html, March 12.2006.

Wikipedia (http://en.wikipedia.org/wiki/), *Fitness proportionate selection*, http://en.wikipedia.org/wiki/Fitness_proportionate_selection, February 19.2006.

Wikipedia (http://en.wikipedia.org/wiki/), *Roulette-Wheel Algorithm,* http://en.wikipedia.org/wiki/Roulette_wheel_ selection**.** February 19.2006

Wikipedia (http://en.wikipedia.org/wiki/), *Genetic algorithm,* http://en.wikipedia.org/wiki/Genetic_algorithm. February 19, 2006.

Wikipedia (http://en.wikipedia.org/wiki/), *Hill-climbing algorithm,* http://en.wikipedia.org/wiki/Hill-climbing. March 12, 2006.

Wikipedia (http://en.wikipedia.org/wiki/), *Ontology in Computer Science*, http://en.wikipedia.org/wiki/Ontology_%28computer_science%29. March 12, 2006.

Wikipedia (http://en.wikipedia.org/wiki/), *Semantic Web*, http://en.wikipedia.org/wiki/Semantic_Web. March 12, 2006.

Winston, Patrick Henry, *Artificial Intelligence, Third Edition*, Addison-Wesley Publishing Company**,** 1992.

Zhang, R. Y., Laks V.S LAkshmanan, and Ruben H. Zamar, Department of Statistics, University of British Columbia, Vancouver: *Extracting Relational Data from HTML Repository*, SIGKDD Explorations, 2005.

Zhang, Y., School of Information Science & Technology, Pennsylvania State University, Xiang Ji, NEC Laboratories America, Cupertino, CA, Chao-Hsien Chu, School of Information Science & Technology, Pennsylvania State University, and Hongyuan Zha, Department of Comptuer Science & Engineering, The Pennsylvania State University, *Correlating Summarization of Multi-source News with K-Way Graph Bi-clustering*, SIGKDD Exploration, 2005.