Dewdrop version 1.0 | Full Documentation

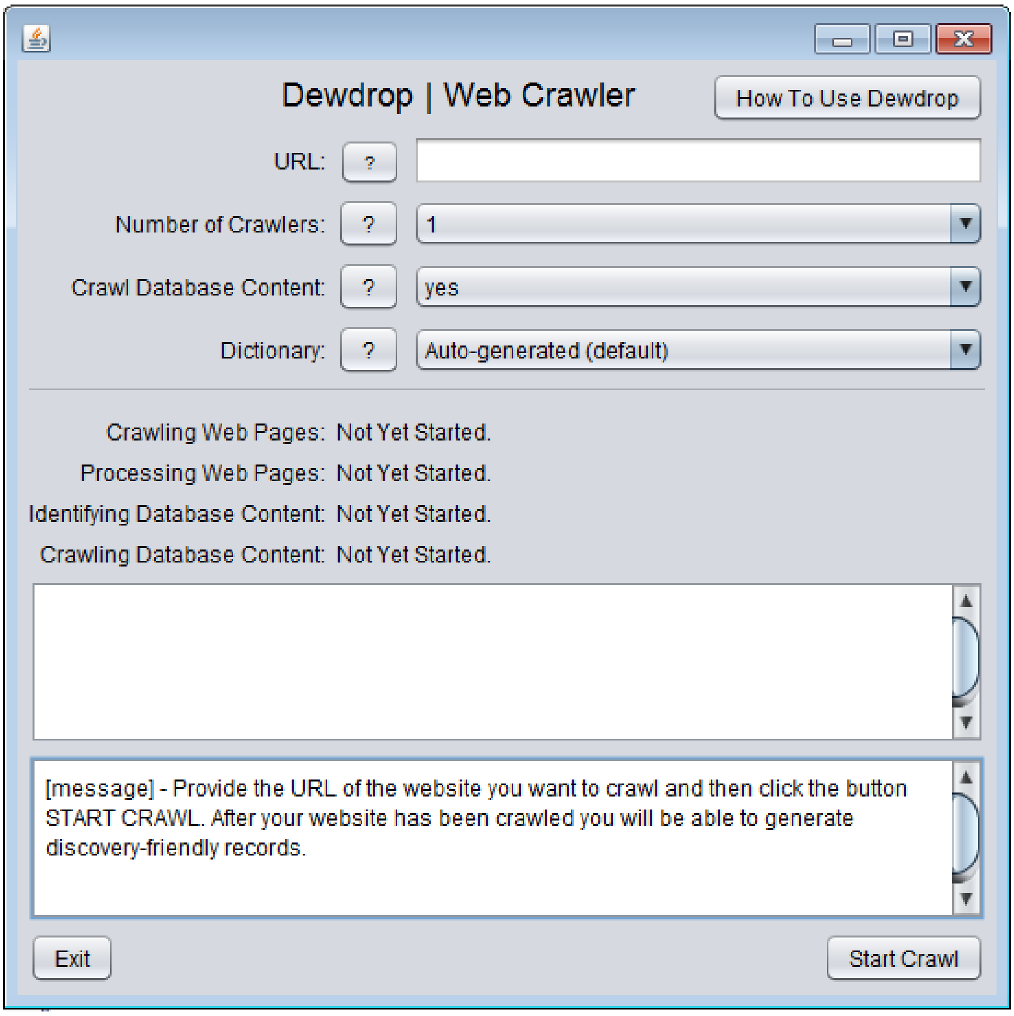
This documentation released on 3rd May 2016.

Credits: Michael Pidd, Ryan Bloor, Matthew Groves, Jamie McLaughlin.

Table of Contents

1. Introducing Dewdrop and Discovery-Friendly Records

1. Dewdrop is a software program designed to create discovery-friendly records of your website. It is a web crawler combined with a text analyser.
2. Dewdrop is designed to make your website content more easily accessible to search engines and third-party content aggregators by creating versions of your content (discovery-friendly records) that these services understand. You can also use Dewdrop to acquire content from third-party websites for your own data mining and other research purposes.
3. Dewdrop is particularly suitable for poorly designed websites that cannot be easily discovered or indexed by search engines and third-party content aggregators.
4. Dewdrop's discovery-friendly records do not make your existing website content invalid. The records act as *signposts*, directing search engines and content aggregators to your website.
5. Dewdrop requires minimal technical know-how.
6. Dewdrop will crawl your website (HTML pages, PDF documents, Word documents and database content), analyse the textual content of your website and then create new, discovery-friendly records.
7. Discovery-friendly records are a version of your website content that is presented in HTML RDFa, RDF or JSON-LD formats, depending on your preference. We recommend HTML RDFa for beginners.
8. When Dewdrop has finished you can ask your server administrator to save these records to the root directory of your website. They will sit *alongside* your existing website content.
9. Read the Quick Start Guide.
10. There are only two screens for the user interface. You can see what the screens look like here:



URL: ?

Number of Crawlers: ? 1

Crawl Database Content: ? yes

Dictionary: ? Auto-generated (default)

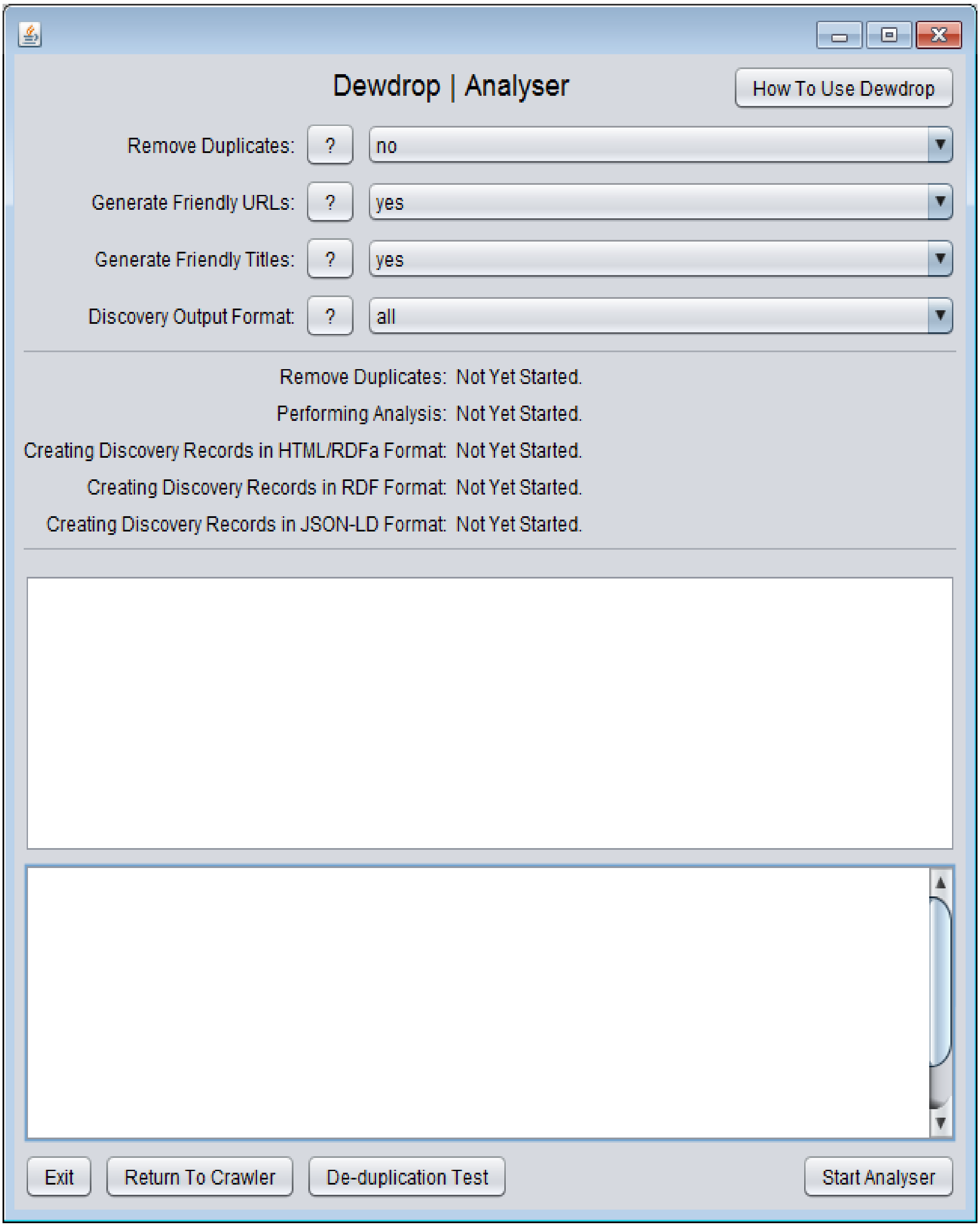
Crawling Web Pages: Not Yet Started.

Processing Web Pages: Not Yet Started.

Identifying Database Content: Not Yet Started.

Crawling Database Content: Not Yet Started.

[message] - Provide the URL of the website you want to crawl and then click the button START CRAWL. After your website has been crawled you will be able to generate discovery-friendly records.



Remove Duplicates: no

Generate Friendly URLs: yes

Generate Friendly Titles: yes

Discovery Output Format: all

Remove Duplicates: Not Yet Started.

Performing Analysis: Not Yet Started.

Creating Discovery Records in HTML/RDFa Format: Not Yet Started.

Creating Discovery Records in RDF Format: Not Yet Started.

Creating Discovery Records in JSON-LD Format: Not Yet Started.

2. Minimum System Requirements

1. 8GB RAM, i5 processor.
2. A hard drive of at least 397MB (Dewdrop's program size). However, your hard drive also needs enough space to accommodate the data being retrieved and processed by Dewdrop; which is dependent on the size of the website being crawled. At least 1GB is recommended.
3. Dewdrop requires Java to be installed on your computer. You can download the latest version from here: https://java.com/en/download.
4. Dewdrop will work on any operating system that has Java installed.
5. For information, the following system features will impact on the overall speed of Dewdrop:
    a. The read/write speed of your hard drive.
    b. The size and read/write speed of your RAM (memory).
    c. The speed of your internet connection.
    d. The speed of your computer's processor and the number of cores.

3. Quick Start Guide

1. First, you need to **crawl a website**:
    a. Launch Dewdrop.exe if you are using a Windows computer. Launch Dewdrop.jar if you are using a Linux or Apple computer.
    b. In the box "URL" provide a valid URL for the website you wish to crawl, such as "http://www.hrionline.ac.uk/dewdrop-test".
    c. The URL must identify a domain or a directory within the domain. It must not identify a specific web page. Web pages and any query string values must be removed from the URL by the user. For example, the following URL is correct: http://www.hrionline.ac.uk/dewdrop-test whereas the URL http://www.hrionline.ac.uk/dewdrop-test is incorrect.
    d. Keep the option "Number of Crawlers" as "1".
    e. Keep the option "Dictionary" as "default".
    f. Keep the option "Crawl Database Content" as "yes".
    g. Websites with large databases will have a significant impact on the duration of the crawl. As such, the user could consider using only the top 100 most frequent keywords

found on the website’s standard HTML pages. The option is available in the dropdown menu for “Crawl Database Content”.

h. Click the button “Start Crawl”.

i. Dewdrop will proceed to crawl the website. The first message panel provides information about the crawler’s progress. The second message panel provides instructions to the user.

j. Wait for the crawler to finish. On large sites crawling can take a very long time. When it has finished you will see an instruction in the second message panel which includes information about how many pages it has found.

2. Next, you need to **analyse the crawled content and create discovery-friendly records**:
    a. Click the button “Next”. This takes you to the Analyser screen.
    b. Keep the option “Remove Duplicates” as “no”.
    c. Keep the option “Generate Friendly URLs” as “yes”.
    d. Keep the option “Generate Friendly Titles” as “yes”.
    e. Change the option “Discovery Output Format” to “HTML with RDFa”.
    f. Click the button “Start Analyser”.
    g. Dewdrop will proceed to analyse the crawled content. When it is finished Dewdrop will create the discovery-friendly records in your chosen output format. Wait for the process to finish.
    h. When Dewdrop is finished the message panel will explain that you can now access your discovery-friendly records.
    i. There are two ways in which you can access your discovery-friendly records:
        i. First, by clicking the “View Discovery Records” button which provides a basic HTML interface for viewing the records.
        ii. Second, by navigating to Dewdrop’s home directory. You will see that Dewdrop has created a directory called “logs”. This contains a further directory called “discovery”. Your discovery-friendly URLs are located in the directory called “discovery”.

3. Finally, you might wish to **deploy the discovery-friendly records on your website**:
    a. We recommend that the following steps are undertaken by your server administrator.

b. First, check with your server administrator to ensure that the steps outlined below are permitted by your server environment. In some instances the server administrator might need to undertake some additional configuration of the server.
c. The “discovery” directory should contain a directory which is the name of the discovery output format. For example: “rdfa” if you chose the output format “HTML with RDFa”.
d. Copy the “rdfa” directory to the root directory of your website.
e. The “rdfa” directory will contain your discovery records, plus some additional files called “sitemap.xml”, “.htaccess” and “.htmapper”.
f. Move these additional files to the root directory of your website. Also, if present, move any file that includes the word “sitemap” in its filename, such as “sitemap_index.xml” and “sitemap1.xml”.
g. For example, if your website has the root directory “mywebsite” which contains the file “index.htm” and a directory of web pages called “pages”, the discovery-friendly records should be deployed to the root of the directory like so:

   /mywebsite
   - index.htm
   - /pages
   - sitemap.xml
   - /rdfa
   - .htaccess
   - htmapper.txt

4. Alternatively, you might wish to **provide a third party content aggregator with access to your discovery-friendly records**:
   a. The “discovery” directory should contain a directory which is the name of the discovery output format. For example: “rdfa” if you chose the output format “HTML with RDFa”.
   b. The “rdfa” directory will contain your discovery records, plus some additional files called “sitemap.xml”, “.htaccess” and “.htmapper”.
   c. Simply provide the third party with a copy of this directory.

4. Data Files and Directory Structure

1. For Microsoft Windows users, Dewdrop consists of a single executable file called "Dewdrop.exe".
2. For Linux and Apple users, Dewdrop consists of a single executable JAR file called "Dewdrop.jar".
3. Dewdrop includes a directory called "documentation".
4. In the process of running Dewdrop on a website the following files and directories will be created in a directory called "logs":
    a. The file "crawler4j.log" contains all messages that are outputted to first message window (in the Dewdrop interface).
    b. The file "console_log.txt" contains the last 50 lines of messages that are outputted to the first message window.
    c. The file "console_message.txt" contains all messages that are outputted to the second message window (in the Dewdrop interface).
    d. The file "crawler_stats.txt" contains the number of web pages crawled and the number of database pages crawled (deep web content). These stats are saved here so that they can be passed from one screen to another within the Dewdrop interface.
    e. The directory "initial" contains raw records from the crawled web pages.
    f. The directory "deepweb" contains raw records from the database crawl.
    g. The directory "filtered" contains the raw records created as a result of running a de-duplication process.
    h. The directory "discovery" contains the discovery–friendly records as well as a copy of the raw records.
5. If the user requests that database content be crawled, the contents of the directory "initial" will eventually be superseded by the contents of the "deepweb" directory. This is because a crawl of database content will inevitably result in the initial web pages being crawled again (i.e. the crawler will find these pages again).
6. If the user requests that the raw records be de-duplicated, the contents of the directory "deepweb" and "initial" will be superseded by the contents of the "filtered" directory. The directory "filtered" contains the results of the de-duplication process.
7. Each time the user asks Dewdrop to crawl a new website, the "logs" directory including its contents is deleted and replaced with a new directory. Therefore, the user should remove any files that s/he wishes to retain before running a new crawl (for example, copy the directory "discovery" to a new location so that the discovery-friendly records can be deployed).

5. Technical Specification

1. Dewdrop is a desktop application written in Java. The user interface is written in Swing (Java).
2. Dewdrop uses a number of open source third-party technologies to undertake specific tasks:
    a. Crawler4J is the underlying web crawler. We decided to use Crawler4J because it is extremely lightweight, heavily customisable and scalable. https://www.openhub.net/p/crawler4j
    b. Apache Tika is used for extracting textual content form binary data files such as PDF and Microsoft Word. https://tika.apache.org
    c. JSOUP is used for light parsing of HTML pages and to extract certain HTML elements such as forms and body text. http://jsoup.org
    d. OpenNLP is used for the initial parsing of a document, first by separating it into a collection of sentences and then tokenising each sentence into a collection of words. https://opennlp.apache.org
    e. StanfordNLP is used to extract entities from a document such as people, locations, dates and organisations. http://stanfordnlp.github.io/CoreNLP
    f. Site Map Gen 4J is used to create the final list of discovery records and produce the Sitemap. https://github.com/dfabulich/sitemapgen4j
    g. Maven Exec is used to manage the third party dependencies (all of the above). http://www.mojohaus.org/exec-maven-plugin
3. External classifiers are used for tokenisation, including parts of speech, sentences and named entities.
4. The Wordnet lexical database of English nouns, verbs, adjectives and adverbs is used in conjunction with part of speech tokenisation to generate word synonyms. https://wordnet.princeton.edu

6. Overview of the Dewdrop Process

1. When using Dewdrop to generate discovery-friendly records for a website, the software performs the following processes:
    a. The Crawler: The target website is crawled and pages are retrieved. The crawler will analyse web pages, attachments (e.g. PDF files) and database content.

   b. The Analyser: Entities (such as person names) and keywords are retrieved from the crawled content. The entities, keywords and original content are combined to create new, discovery-friendly records. Dewdrop exports these records along with a Sitemap ready for re-deployment.

2. Each process is described in detail below, along with a discussion of the technical challenges and considerations when performing each process.

7. Overview of the Crawling Process

1. When crawling a website Dewdrop will perform the following processes:
   a. A crawl of web pages and attachments by following hyperlinks.
   b. An analysis of the crawled web pages in order to identify keywords and search forms that can be used for crawling database content.
   c. A crawl of database content, if present, by submitting keywords via the search forms.

8. The Process of Crawling Web Pages

1. The crawler looks for a robots.txt file and adheres to its instructions. It informs the user if the robots.txt rules are too strict to enable an effective crawl of the website (for example, if the robots.txt prohibits crawling).
2. The crawler also has a politeness of 100 milliseconds. This means that the crawler will always wait 100 milliseconds between each request to the website’s server.
3. The process of crawling involves working through a stack of hyperlinks, beginning with the URL that has been supplied by the user. The stack of hyperlinks is created in the process of crawling, as described below:
   a. The crawler takes the first hyperlink from the stack and checks that by following the link it is adhering to the rules of the robots.txt file (at the outset of crawling the only hyperlink in the stack will be the index file implied by the URL).
   b. The crawler follows the hyperlink and parses the page it encounters to establish the content type. The crawler will permit three types of content: plain text, HTML/XML and binary data (PDF and Microsoft Word).

c. The crawler parses the page and extracts all hypertext links that use the HTML element <A> with the attribute HREF.

d. All links extracted from the page are added to the crawler’s stack.

e. The crawler extracts content from the page in line with Dewdrop’s required record elements. These required record elements are: the <TITLE>, any keywords that form part of the page’s <META> tags and the textual content of the page itself with stopwords and punctuation removed.

f. The crawler saves the extracted content to an XML file, which we refer to as the *raw record*.

g. The crawler examines the stack of hyperlinks to determine whether any hyperlinks remain. If the stack is not empty, the crawler takes the next hyperlink and repeats the steps outlined above.

9. Preparing to Crawl Database Content

1. After the crawler has retrieved all the web pages that can be accessed via hyperlinks it will proceed to analyse the content of the website’s database if: a) a database exists and b) the user has instructed Dewdrop to crawl database content.
2. Database content is content that can only be retrieved by submitting a search form. So although the majority of modern websites construct their web pages by retrieving content from databases (using languages such as PHP), we use the phrase *database content* in Dewdrop to refer to content that requires end-users to submit a search keyword. In other words, the content of the web page depends on the keyword that is supplied by the end-user, rather than a hyperlink that points to a specific page of content. This type of content is sometimes referred to as *deep web* content.
3. If the website has no database content (i.e. it is composed entirely of HTML files) then the user should select “Crawl Database Content” -> “no” from the dropdown menu.
4. If the website has no database content but the user does not select “no” from the dropdown menu, Dewdrop will waste time undertaking preparations to crawl a database only to discover that no database exists.
5. Dewdrop retrieves content from databases using the following process:

   a. It identifies whether database content exists by identifying the presence of HTML forms in the pages that it has already crawled.
   b. It submits keywords to the HTML forms, mimicking the actions of an end-user, in order to retrieve pages of database content. The keywords are generated from the pages that it has already crawled.
   c. It then parses each page and creates a stack of hyperlinks in the same way that ordinary HTML web pages are parsed (see: 8. The Process of Crawling Web Pages).

6. The preparations for crawling database content are as follows:
   a. Dewdrop analyses the raw records of all pages found during the initial crawl.
   b. Every unique word is added to a keyword list. For a keyword to qualify it must not be on the stopword list and it must be between three and 10 characters long.
   c. Every HTML form is examined and unique form actions are added to an actions list. A form action is the attribute ACTION that occurs in a valid HTML form. The value of the attribute ACTION is typically a hyperlink to a results page, for example <FORM METHOD="GET" ACTION="results.php">. A unique form action is therefore equivalent to a unique URL.
7. The keyword list is created by analysing the content of raw records, as described above. This is the default setting. However, the user can also select a specific dictionary via Dewdrop's interface. The current release of Dewdrop includes two dictionaries: English and French.
8. The dictionaries use Wordnet lexical databases. Different language dictionaries can be found here: http://globalwordnet.org/wordnets-in-the-world
9. The user should understand that language-specific dictionaries are not necessarily a requirement for websites that use foreign languages. In the vast majority of instances Dewdrop's default setting will suffice. The circumstance under which a specific dictionary is needed is as follows:
   a. The language of the database content is different to and not present in the website's interface or standard HTML pages. For example, let us imagine that there is a website with an English language interface that requires the user to search a database of French novels using French language keywords. Dewdrop's standard procedure will be to use words from the interface to query the database, so unless the interface includes some French vocabulary or the database includes some English vocabulary, Dewdrop will be unable to return any results. It is equivalent to an English user with no knowledge of French attempting to search a database of French vocabulary. In our experience this situation is rare.

10. The user should also understand that HTML forms that include foreign language vocabulary in dropdown menus will be able to retrieve foreign language content without the aid of a dictionary.

10. The Process of Crawling Database Content

1. Having acquired a list of keywords, the crawler adds an empty string and asterisk (*) to the beginning of the list. Their inclusion is intended to address the tendency for some websites to enable a blank search form to be submitted or a search form with the wild card character “*”. Typically these two types of searches retrieve the entire contents of a database, so they are privileged in the keyword list.
2. The crawler loads the list of keywords.
3. The crawler loads the list of form actions (essentially a list of URLs).
4. For each form action the crawler loops through the list of keywords, submitting individual keywords to the action URLs.
5. Each submission returns a results page. The crawler extracts all hyperlinks from the results page and adds them to a stack of hyperlinks.
6. When all keywords have been submitted to all action URLs, the crawler proceeds to work through the stack of hyperlinks as described in the section 8. The Process of Crawling Web Pages.

11. Crawler: Technical Issues and Considerations

1. The crawler will not follow hyperlinks that are written using Javascript. The crawler will only follow hyperlinks that are written using the HTML element <A> with a valid HREF attribute. For example, the crawler will recognise the link <A HREF=”index.htm”> but it will not recognise the link <A HREF=”javascript:void(0)” onClick=”getPage(‘index.htm’)”>. This limitation is true for all web crawlers (at the time of writing) and means that Dewdrop is unable to crawl websites that use JQuery or other forms of Javascript code for user navigation unless there is a HTML alternative (e.g. a sitemap).

2. The URL supplied by the user can include the HTTP prefix. If the user does not provide a prefix, the HTTP prefix will be added. Note that if the prefix should be HTTPS but the user does not supply this, the crawler is likely to fail because it will assume that the prefix is HTTP.
3. The URL supplied by the user must identify a domain or a directory within the domain. It must not identify a specific web page. Web pages and any query string values must be removed from the URL by the user. For example, the following URL is correct: http://www.hrionline.ac.uk/lane whereas the URL http://www.hrionline.ac.uk/lane/index.php is incorrect.
4. The crawler will not cross domains or follow links that give the appearance of crossing domains. So if the user provides the URL http://www.grassportal.org but the majority of the website's hyperlinks point to http://grassportal.shef.ac.uk the crawler will ignore these links even though, in reality, the domain names http://www.grassportal.org and http://grassportal.shef.ac.uk are pointing to content in the same directory.
5. The crawler will follow hyperlinks to a maximum depth of 10. That is, if the homepage is depth 0, every page linked to the homepage is depth 1 and every page that is linked to these is depth 2. A future release of Dewdrop might include this as a configurable option in order to increase the speed of the crawling process.
6. When crawling database content the crawler makes the assumption that a results page contains hyperlinks to content pages. The crawler only saves textual data from content pages. However, hypothetically there could be a case in which the results page is the only page of content. For example, a search form for a telephone directory will return a list of names and telephone numbers as its result page but there will be no hyperlinks from this page to individual telephone record entries because the results page presents all the information that an end-user requires. The current version of Dewdrop would not record this content. However, we have not come across a website to date in which this characteristic is present.
7. The option "Number of Crawlers" enables the user to undertake *multi-threaded* crawling. In practice this means that Dewdrop will send multiple crawlers to the website. The multiple crawlers will work through the same stack of hyperlinks, ensuring that they do not duplicate each other's work. In most instances the effect will be to shorten the time required to crawl the entire website.
8. Although it might seem logical to always use the maximum number of crawlers in order to crawl a website in the shortest time possible, the user should be cognisant of the following issues:
    a. In some instances, multiple crawlers could be interpreted as an attack against the server, such as a Denial of Service Attack.

   b. In some instances, multiple crawlers might slow down the process of crawling because of the time required to manage the allocation of hyperlinks from the stack. In other words, the time saved by having multiple crawlers could be lost due to the time taken to manage the distribution of work.

9. When setting the option “Number of Crawlers”, we recommend the following:
   a. For websites with less than 500 pages we recommend that 1 to 2 crawlers are used.
   b. For websites with 500+ pages we recommend that 3 to 4 crawlers should be used.
   c. The use of 5 or more crawlers should be used at the user’s discretion, bearing in mind that the host server could interpret this as a malicious action.
   d. For websites that are not owned by the user or websites that are hosted by a commercial hosting provider we recommend caution in the number of crawlers used.
10. Crawling will take different lengths of time, depending on the characteristics of the website. The following characteristics are known to impact on the duration of a crawl:
    a. Large websites. Generally, the larger a website is, the longer it will take to crawl. For large websites it is not unusual for the crawl duration to be between a 2 and 24 hours.
    b. Files that are not of interest to the crawler. When retrieving a web page the crawler also has to retrieve the HTML file but also other files that are associated with it (e.g. a CSS file, Javascript file and images) even though these will be subsequently discarded by the crawler.
    c. Internal redirects (301), if the redirect sends the crawler to a page which it has already seen.
    d. Politeness. Dewdrop has a politeness setting of 100 milliseconds between every request to the server.
    e. The host server. Some servers will be less responsive than others and some servers will delay or queue requests in order to prevent the server from becoming overloaded.
    f. The user’s computer. The process of crawling involves parsing content and saving it to the user’s hard disk as a raw record. Therefore the speed of the user’s processor and RAM read/write will impact on the overall speed of crawling.
    g. For database content, the size of the keyword list will have a significant impact on the duration of the crawl. As such, the user is able to dictate that Dewdrop crawls database content using only the top 100 most frequent keywords found on the website’s standard HTML pages. The option is available in the dropdown menu for “Crawl Database Content”. This is recommended for large websites.

11. In testing the crawler has consistently returned more results than are believed to be in the target website. In other words, Dewdrop has a tendency to over crawl rather than under crawl. This is largely due to duplication issues (see: 13. De-Duplication).

12. Overview of the Analyser Process

1. After crawling has finished the user is able to proceed with the analyser. The analyser will perform the following processes:
    a. Run a de-duplication test on a sample set of raw records, if requested by the user.
    b. De-duplicate the entire dataset of raw records, if requested by the user.
    c. Identify named entities (person names, location names, organisations and dates) from each raw record. These are added to each raw record as new metadata.
    d. Identify the top 10 most frequent keywords across the entire dataset of raw records. These are added to each raw record as new metadata.
    e. Identify the top 10 most frequent keywords for each raw record that are infrequent across the entire dataset of raw records. These are added to each raw record as new metadata.
    f. Identify synonyms for the frequent keywords. These are added to each raw record as new metadata.
    g. Modify the TITLE and URL for each record, as requested by the user using the dropdown menus “Generate Friendly URLs” and “Generate Friendly Titles”.
    h. Transform the raw records into discovery-friendly records using different file formats, as requested by the user using the dropdown menu “Discovery Output Format”.
    i. Generate a Sitemap so that the discovery-friendly records can be deployed.

13. De-Duplication: Considerations, Technical Challenges and the De-Duplication Test

1. The process of crawling a website will result in duplicate content being returned and treated as unique raw records.

2. Duplicate records are returned because the Dewdrop crawler considers unique URLs to constitute unique content. This prevents the crawler from analysing the same web page more than once. However, there are many instances when the same web page will have a slightly different URL, especially when dealing with database content. For example, let us imagine that the web page at www.mydomain.com/page.php?id=132 contains the words "the cat sat on the mat". If the only way of retrieving this page is by querying a database using the keyword "cat" or "mat" *and* the keyword is included in the resulting URL, we might end up with two unique URLs that point two the same page: page.php?id=132&kw=cat and page.php?id=132&kw=mat. The problem also exists where URL query strings include session IDs, navigation information etc.
3. The next release of Dewdrop should reduce this problem by running an algorithm that seeks to identify what is the significant (required) portion of the query string for retrieval purposes. For example, in the above example the attribute ID= is required but the attribute KW= is superfluous.
4. Dewdrop includes an ability to de-duplicate the dataset of raw records by comparing every record against every other record. For example, for a dataset of 100 records Dewdrop will compare file 1 against files 2, 3, 4 … 100. It will then compare file 2 against files 3, 4, 5 … 100.
5. A file is considered a duplicate if the similarity index (comparing character to character) is greater than 90%.
6. Although the number of files being compared declines by one with each loop, the process can be time consuming for a large dataset.
7. The user should consider running the De-Duplication Test before deciding whether or not to include de-duplication as part of the Analyser process. Running the De-Duplication Test should help inform the user whether duplication is a significant enough problem to require duplicates to be removed from the final dataset of discovery-friendly records.
8. The De-Duplication Test consists of comparing a sample set of up to 100 records against itself. The total number of comparisons for 100 records will be 4,950. If the total number of raw records is less than 100 the test will use the total number of raw records.
9. After the De-Duplication Test has finished the user can view the results. Assessing the success of de-duplication and the extent to which duplication in the dataset is a problem will vary on a case-by-case basis. As such, only a user who possesses a good knowledge of the website will be best placed to judge the extent of the problem.
10. When assessing duplication and the success of the De-Duplication Test the user should be mindful of the following:
    a. The larger the website, the greater the number of duplicates.

b. The larger the website, the longer it will take for Dewdrop to undertake de-duplication. For example, 1,000 records will require 499,500 comparisons.

c. The threshold of 90% might not be a useful indicator of duplication in all cases. For example, a catalogue of historical documents might contain individual records that are identical in every respect except for the catalogue reference number or a few words of description. For an end-user these are all unique records but their similarity might be far higher than 90%. Allowing the user to change the threshold might be a feature in the next release of Dewdrop.

d. It might not be necessary to have a dataset of entirely unique discovery-friendly records, depending on the use case. For example, if Dewdrop is being used in order to improve the discoverability of a website, duplicate records should not be a problem because the aim is not to replicate the website exactly. However, if Dewdrop is being used in order to retrieve content for data mining purposes (e.g. as part of a corpus linguistics research project), duplication might affect subsequent research results.

14. The Process of Analysing the Raw Records

1. The analyser parses all the raw records and generates a global list of keywords. The list represents every unique word in the dataset which is not a stopword and which is between three and 10 characters in length.
2. The analyser identifies the top 10 most frequent words in the global list of keywords. These are the most frequent keywords across the entire dataset of raw records.
3. The analyser parses each raw record and identifies *named entities* from the dataset of raw records. Named entities are: person names, location names, dates and organisations.
4. The analyser parses each raw record and identifies the top 10 most frequent words in the record that also have low frequency in the global list of keywords.
5. The analyser loops through both sets of frequent keywords and identifies synonyms for each word. This is *deduced metadata*. Synonyms are identified by using the synonym tables in Wordnet as a look-up.
6. The analyser adds the entities, popular keywords and synonyms to the raw records as additional metadata.
7. The analyser modifies the TITLE element of each record, if the user has requested this. Modification involves appending the existing title with: a) the record's top two most frequent entities from people and locations; b) and the record's top two most frequent keywords. The appended words are separated using a white space.
8. The analyser modifies the URL of each record, if the user has requested this. Modification involves appending the existing URL with: a) the record's top two most frequent entities from people and locations; b) and the record's top two most frequent keywords. The appended words are separated using a hyphen.

15. Analyser: Technical Issues and Considerations

1. The analyser identifies the top 10 most frequent keywords across the entire dataset of raw records. Our assumption is that these words will include search terms that identify the website as a whole (and therefore common search terms when end-users are trying to locate the website).
2. The analyser identifies the top 10 most frequent keywords in each record that are also infrequent across the entire dataset of raw records. Our assumption is that these words identify

unique content in the individual record that distinguish it from the website as a whole (and therefore might be used as search terms when end-users are trying to locate the record).

3. A potential pitfall of identifying words that are frequent in the record but infrequent across the rest of the dataset is that nonsense words (words that are misspelt or inaccurately rendered) might be privileged because they are unlikely to be widely reproduced.
4. The analyser uses StanfordNLP for entity extraction which we found to be more accurate than OpenNLP, although its performance is slightly slower.
5. The analyser identifies synonyms for each keyword. Our assumption is that synonyms will increase the discoverability of a record. For example, a record which contains the word “spade” will be retrieved if the user searches for the word “shovel”. The process of identifying synonyms includes using Stanford NLP to conduct part-of-speech tagging. Understanding the part of speech for each keyword enables more accurate identification of synonyms.
6. A potential pitfall when identifying synonyms is the quality of the thesaurus that Dewdrop uses. For example, the Wordnet database of synonyms includes the words “Bible” and “Koran” as synonyms of “book”. In many contexts these will be misleading. A future release of Dewdrop might include an improved thesaurus and/or the ability for the user to turn synonyms off.
7. The decision of whether to create discovery-friendly titles and URLs is left to the user. Ideally the user will be sufficiently familiar with the target website to ascertain whether or not the existing titles and URLs are already discoverable. For example, the URL “bessofhardwick.org?id=32” and the title “Bess of Hardwick” are not discovery friendly, whereas the URL “bessofhardwick.org/letter-to-john-salisbury” and the title “Bess of Hardwick: Letter to John Salisbury” can be considered more discovery friendly.
8. In some instances, notably where a website already has discovery-friendly URLs and titles, the use of this feature in Dewdrop can be unhelpful and potentially make URLs and titles less discovery-friendly.
9. The user should appreciate that *discovery friendly* is not necessarily considered to be the same as *human readable*.

16. Generating Discovery-Friendly Records

1. The final process that Dewdrop performs is to transform the raw records into discovery-friendly records using the file formats chosen by the user. The different file formats and recommendations for their use are described in the next section.

2. The process is as follows:
    a. Dewdrop transforms the raw records into the discovery file format(s) chosen by the user.
    b. Dewdrop generates a Sitemap in accordance with the Sitemap Protocol: http://www.sitemaps.org.
    c. Dewdrop creates a HTACCESS file and HTMAPPER file if the user requested that discovery-friendly URLs be created as part of the Analyser process.
    d. All files are saved to the directory “discovery” which is located in the directory “logs”.
3. The raw records are transformed into the discovery-friendly file format and given file names that are sequential numbers. For example: 1.html, 2.html, 3.html.
4. The Sitemap points to each discovery-friendly record.
5. The Sitemap Protocol permits a maximum number of 50,000 URLs. When the total number of records exceeds 50,000 Dewdrop will create a parent Sitemap that points to multiple Sitemaps. These subordinate Sitemaps will then point to the discovery-friendly records.
6. If the user has chosen to have discovery-friendly URLs, these URLs will be included in the metadata of each discovery-friendly record. If the user has *not* chosen to have discovery-friendly URLs, the true URLs of the web pages will be included in the metadata.
7. The precise notation that is used for identifying the true (canonical) URLs varies depending on the output file format. Please see the sections below for specific information.
8. If the user has chosen to have discovery-friendly URLs, a HTACCESS file and a HTMAPPER file will be created. The HTMAPPER file is used to map the discovery-friendly URLs to the true URLs. The HTACCESS file instructs the server to read the HTMAPPER file and perform the relevant redirect (i.e. redirecting the discovery-friendly URL to the true URL).
9. The HTACCESS file contains the following directives:

```
RewriteEngine On
RewriteMap urlremapper txt: htmapper.txt
RewriteRule (.*) ${urlremapper:$1} [R=301,L]
```

10. The HTMAPPER file is used as the URL look-up table because the inclusion of URL look-ups in the HTACCESS file is considered to cause server performance issues when the number of look-ups exceeds 100,000. The HTMAPPER file is used irrespective of the number of look-ups.
11. If a user has chosen to have more than one discovery file format, each type of file format is treated as a separate dataset, with its own discrete set of discovery-friendly records, Sitemaps, HTACCESS file and HTMAPPER file.

12. When the discovery-friendly records have been created, they can be accessed from the directory “discovery” which is located in the directory “logs”.

17. Discovery Output Format: HTML with RDFa

1. This is the recommended output format if the user wishes to improve the discoverability of their website for search engines. However, some search engines increasingly prefer JSON-LD as a format. The user should check which format is most appropriate. We do not recommend that discovery-friendly records with different output formats are deployed together.
2. The schema for HTML with RDFa can be summarised as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="canonical" href="{REAL URL}" >
<title>{TITLE}</title>
<meta name="keywords" content="{KEYWORDS: ACTUAL AND SYNONYMS}">
</head>
<body>
<div id="document" vocab="http://schema.org/" typeof="Document" about="{REAL OR FRIENDLY URL}">
<h1><a property="url" href="{REAL OR FRIENDLY URL}">{REAL OR FRIENDLY TITLE}</a></h1>
<p property="description">{CONTENT}</p>
<span property="Created">{DATE AND TIME}</span>
<div>
<h2>People</h2>
<ul>
<li vocab="http://schema.org/" typeof="Person">{PERSON}</li>
</ul>
</div>
<div>
<h2>Locations</h2>
<ul>
```

```
<li vocab="http://schema.org/" typeof="Location">{LOCATION}</li>
</ul>
</div>
<div>
<h2>Organisations</h2>
<ul>
<li vocab="http://schema.org/" typeof="Organisation">{ORGANISATION}</li>
</ul>
</div>
<div>
<h2>Dates</h2>
<ul>
<li vocab="http://schema.org/" typeof="Date">{DATE}</li>
</ul>
</div>
</div>
</body>
</html>
```

18. Discovery Output Format: RDF

1. This is the recommended output format if the user wishes to create a discovery-friendly version of their website that can be given or exposed to a third party content aggregator, or if the user wishes to acquire content from a website in order to perform further analysis of the content (e.g. text data mining or corpus linguistics).
2. The schema for RDF can be summarised as follows:

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns' xmlns:dc='{REAL URL}/dc#'>
<rdf:Description rdf:about='{REAL URL}'>
<dc:created>{dATE AND TIME}</dc:created>
<dc:url>{REAL URL}</dc:url>
<dc:friendly_url>{FIRENDLY URL}</dc:friendly_url>
<dc:title>{REAL OR FRIENDLY TITLE}</dc:title>
```

```
<dc:content>{CONTENT}</dc:content>
<dc:people>
<rdf:Bag>
<rdf:li>{PERSON}</rdf:li>
</rdf:Bag>
</dc:people>
<dc:dates>
<rdf:Bag>
<rdf:li>{DATE}</rdf:li>
</rdf:Bag>
</dc:dates>
<dc:organizations>
<rdf:Bag>
<rdf:li>{ORGANISATION}</rdf:li>
</rdf:Bag>
</dc:organizations>
</rdf:Description>
</rdf:RDF>
```

19. Discovery Output Format: JSON-LD

1. This is a recommended alternative to HTML with RDFa if the user wishes to improve the discoverability of their website for search engines. At the time of writing, JSON-LD is gaining some popularity with search engines such as Google. The user should check which format is most appropriate. We do not recommend that discovery-friendly records with different output formats are deployed together.
2. The schema for JSON-LD can be summarised as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>{REAL OR FRIENDLY TITLE}</title>
<link rel="canonical" href="{REAL URL}" >
```

```
<script type="application/ld+json">
{
"@context": "http://schema.org",
"@type": "Document",
"title": {REAL OR FRIENDLY TITLE}
"description": {CONTENT}
"url": {REAL OR FRIENDLY URL}
"createdDate": {DATE AND TIME}
"People": {PERSON}
"Locations": {LOCATION}
"Organisations": {ORGANISATION}
"Dates": {DATE}
"keywords": {KEYWORDS: ACTUAL AND SYNONYMS}
}
</script>
</head>
```

20. Deployment

1. If the user wishes to improve the discoverability of their website for search engines we recommend that they choose HTML with RDFa or JSON-LD as a discovery output format. Both these formats include the creation of Sitemaps, a HTACCESS file and a HTMAPPER file.
2. Deployment should consist of copying the entire dataset (discovery-friendly records, Sitemap, HTACCESS file and HTMAPPER file) to the root directory of the website.
3. When Dewdrop generates the discover-friendly records, all data files are saved to a single directory, such as “rdfa” if the user has chosen the output format HTML with RDFa. This directory will contain the records but also the Sitemap, HTACCESS file and HTMAPPER file. However, when deploying to a server the user should place the Sitemap, HTACCESS and HTMAPPER files *outside* the “rdfa” directory so that they reside in the root directory of the website.
4. For example, if the user has a website with the root directory “mywebsite” which contains the file “index.htm” and a directory of web pages called “pages”, the discovery-friendly records should be deployed to the root of the directory like so:

/mywebsite

- index.htm
- /pages
- sitemap.xml
- /rdfa
- .htaccess
- htmapper.txt

5. Some servers might be configured to ignore .htaccess files, preferring to use a central configuration file instead. In these instances the user should consult with the server administrator to modify the server's settings to read .htaccess files.

21. Troubleshooting