ECE 467 Natural Language Processing
Professor Carl Sable
Spring 2010

## Final Project:

# Content Association in Wikipedia



Christopher Mitchell MEE 2010
Akshay Anand MEE 2011
May 5, 2010

## 1. Problem Statement

Wikipedia, the free online encyclopedia, contains a wealth of intellectually and monetarily free content (in common terminology, "free as in speech and free as in beer"). The sheer number of users editing the corpus means that the majority of the articles are well-written and largely factual. However, the relationship between related articles, usually inferred by the *See Also* links at the bottom of each article, are generally incomplete compared to the relationships implied by words linked amidst the text of each article, such as the sample from "The Cooper Union" shown in Figure 1 below.

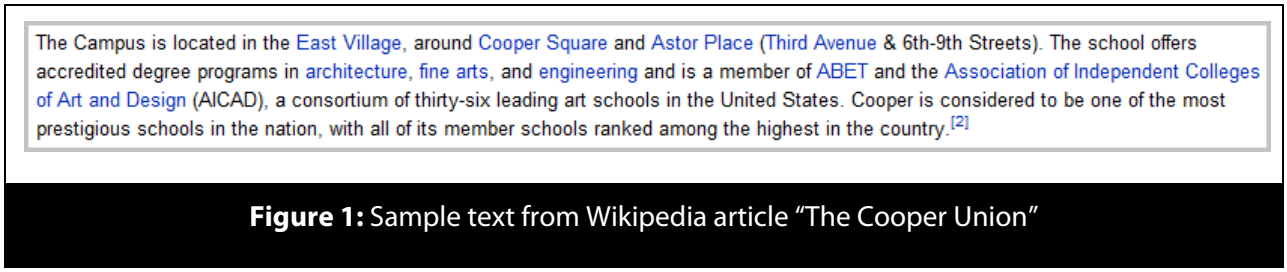
The Campus is located in the East Village, around Cooper Square and Astor Place (Third Avenue & 6th-9th Streets). The school offers accredited degree programs in architecture, fine arts, and engineering and is a member of ABET and the Association of Independent Colleges of Art and Design (AICAD), a consortium of thirty-six leading art schools in the United States. Cooper is considered to be one of the most prestigious schools in the nation, with all of its member schools ranked among the highest in the country.[2]

**Figure 1:** Sample text from Wikipedia article "The Cooper Union"

NLP could provide a unique method of discovering both articles that are related through such correct linking, and that are mutually relevant but insufficiently linked or completely unlinked.

## 2. Proposed Implementation and Hypothesis

We propose a PHP framework to spider Wikipedia, collecting both full-text word lists and lists containing only the words from the text of internal links. We propose comparing the relative performance of a system that attempts to find similarity metrics between articles based on the full text of each article and one based on only on the linked words in each article. PHP is well-suited to the NLP problem; although slower than other interpreted languages, its powerful regex, string manipulation, and web interface capabilities make it a good choice for a spider and tokenizer. In addition, PHP's powerful use of the GD2 graphics library makes visualizations of relationships straightforward; an example from a previous project is shown in Figure 2; a similar visualization tool to be called an "outgraph" was proposed for this project. Because the set of all articles cannot be broken into a trainingset and testing set, we performed full training over a bounded set of articles before testing to determine similarity. Contrary to the original proposal, no implementation of simultaneous training and testing against the growing training set was attempted.

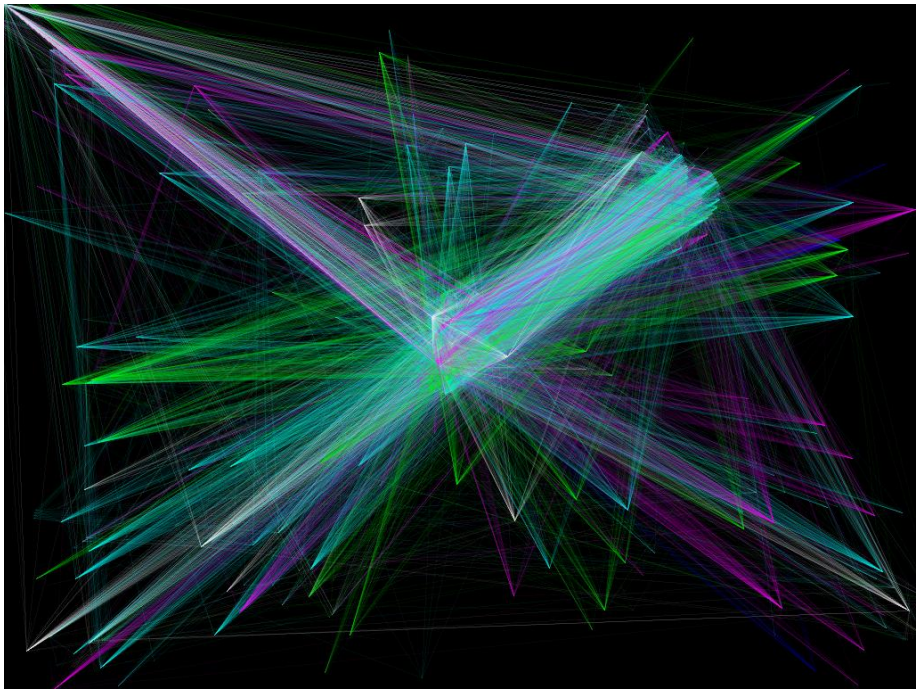

**Figure 2:** Sample partial screenshot from one of Christopher's earlier projects from 2006-7 called "WordNet" (no relation to the Princeton project of the same name)

We hypothesized that the full-text method would be the more effective of the two, as the internal links generally refer to explanations and background information rather than directly related topics. We further hypothesized that both methods would generate coherent similarity rankings and valid matches for each article assuming a general topic, along with some very specific articles that would have no good matches among the extremely small total percentage of Wikipedia articles under examination.

## 3. Experimentation

Experimentation began with a simple implementation of a program to load articles from Wikipedia. Using the socket libraries in PHP, for a given topic string {T}, the page http://en.wikipedia.org/wiki/Special:Export/{T} is fetched. Each such page contains an XML-formatted article including the original markup for headings, images, references, external and internal links, categories, and more. Two full sets of regular expressions were determined, one to remove all extraneous markup other than internal links, specified in Listing 1 below, and a second set to further simplify and remove internal links to leave only plaintext sentences, as shown in Listing 2. Both sets of

code could be further optimized for speed by combining related expressions, such as the removal of images and file links, but have been left separated in this implementation for the sake of clarity.

```
$lines = preg_replace('/\[\[Image\:[^\]]*\]\]/','',$lines); //images
$lines = preg_replace('/\[\[File\:[^\]]*\]\]/','',$lines);  //files
$lines = preg_replace('/\[\[(Category:[^\|\]]+)\]\]/','',$lines);
$lines = preg_replace('/\[\[[^:]+:[^\]]+\]\]/','',$lines);
                                  //remove multilingual synonyms
$lines = preg_replace('/\<\!\-\-[^\-]\-\-\>/','',$lines);  //comments
$lines = preg_replace('/\'\'+/','',$lines);           //remove emphasis
$lines = preg_replace('/\<ref\>[^\<]*\<\/ref\>/','',$lines);   //refs
$lines = str_replace('<references ?/>','',$lines); //<references />
$lines = preg_replace('/\s+/',' ',$lines);   //merge excess whitespace
```

**Listing 1:** PHP regular expression code to remove most markup except internal links

Once the regex was debugged, a set of global arrays was constructed to store the internal links extracted from each article. As the algorithm was written, it examines up to M=30 articles linked from each "initial" article before choosing a new initial article from the set of articles already examined. The spidering process is terminated once at least M articles have been parsed and tokenized. For development experiments, values of M=30, M=100, M=500, and finally M=1000 were used. Experiments for analysis all used M=1000.

```
$lines = preg_replace('/(==+) *[^=]+ *\\1/','',$lines);
                                              //section headers
$lines = preg_replace('/\{\{[^\}]*\}\}/','',$lines);  //taxonomy boxes
$lines = preg_replace('/\[\[([^\|\]]+)\]\]/',"$1",$lines);
                                          //simplify internal links
$lines = preg_replace('/\[\[[^\|]*\|([^\]]*)\]\]/',"$1",$lines);
                                          //simplify internal links #2
$lines = preg_replace('/\[\[([^\]]*)\]\]/',"$1",$lines);
                                          //simplify internal links #3
$lines = preg_replace('/\{\{([^\}]*)\}\}/',"$1",$lines);
                                          //simplify internal links #4
$lines = preg_replace('/\[([^\]]*)\]/','',$lines);
                                          //remove external links
$lines = preg_replace('/\s+/',' ',$lines); //merge excess whitespace
```

**Listing 2:** PHP regular expression code to remove remaining markup

Next tokenization was added, considering each article as one document among the M-document corpus that was the total examined portion of Wikipedia on each run. Tokenization was performed via a simple algorithm that retained only uppercase and lowerspace letters with whitespace, merged all sequential whitespace into a single space, and converted all text to lowercase. Based on experiences from the first project, more complex techniques such as stemming and stop words were not attempted.

After the program was successfully able to extract a word frequency list for each article, a short section of code was written to perform TD-IDF. A similarity calculator could then be added, based on the functionality of the KNN algorithm and using the cosine similarity metric. The only difference between the implementation and KNN is that each article is considered as its own class, so the K most similar articles instead of the K most similar document classes are found for each test article. Debugging was minor for this section. After the first few tests, a file storage section was appended to write the results to a human-readable file, containing the K most similar articles along with similarity scores for each article in the corpus for that run.

The outgraph renderer was the final component written. It is a simplified version of a previous rendering project by Christopher (see Figure 2). The initial article is represented by a point at the center of the render; articles most similar to that article are then placed in a circle around that point, and the points are connected with lines of higher intensity for higher similarity. Because the similarity values are clustered around 0.01 through 0.09 on a 0.0 to 1.0 scale, a logarithmic equation was written to represent intensity, where *s* is the similarity score:

$$\alpha = log(1 + (e - 1)s)$$

The α or transparency value is then scaled from its original [0 1] range to [0 127]•N. The distance between the initial article and each of its similar articles is also scaled in inverse proportion to α, such that more similar articles are placed closer and less similar articles are placed further away. The render process is repeated taking each article as the initial article until every article has been placed in the render and every similarity relationship has been drawn.

## 4. Final Implementation

The code written for this project can be broken into four logical parts: the Wikipedia spider, the tokenizer, the similarity calculator, and the outgraph drawing module. The Wikipedia spider segment of the code takes a starting article and repeatedly loads new articles, using the internal links on each page to find new articles to spider. The tokenizer uses a large amount of regex to strip away images, formatting, tables, category tags, and other extraneous information, leaving only the raw text with internal links. If the current run is performing full-text analysis, then it tokenizes the full text of the article, including internal links. If not, it tokenizes only the text of the internal links. The similarity calculator runs after all spidering and tokenization is complete, comparing each of the articles seen to each of the other articles in that run to find the K most similar. Finally, the outgraph drawing module

renders an interpretation of the results taking represents each article as a point in space, and draws lines of varying intensity connecting the articles proportional to their similarity.

## 4.1 Wikipedia Spider

The Wikipedia spider uses PHP's socket functions to open a connection to Wikipedia and load each specified article. As discussed in Section 3 above, it is responsible for using a series of regular expressions to extract the actual text of the article from the XML-formatted file that is returned from the Wikipedia server, remove markup, and obtain a list of internal links. On any given run, it examines a total of M articles, starting from an initial article and examining N articles linked from that article before choosing a new initial article.

## 4.2 Tokenizer

The tokenizer operates in parallel with the Wikipedia spider, either tokenizing the full text of each article collected and cleaned or tokenizing only the text of the links. For internal links, as the markup allows the text of a link to differ from the title of the target article, the link text was used for tokenization, but the link target for each link was stored in the list of articles for future examination and tokenization. The tokenizer strips all non-alphabetic, non-whitespace characters, merges all sequential lowercase into a single space, and converts all text to lowercase. Stop words, stemming, POS tagging, and similar approaches were not implemented. After all spidering and tokenization is complete, the Term Frequency – Inverse Document Frequency (TF-IDF) algorithm is applied.

## 4.3 Similarity Calculator

The similarity calculator takes each of the M articles in turn, and uses the cosine similarity formula to determine the K most similar articles from among that run's corpus. A sample of results of the first K=8 (out of a total K=16 calculated) for M=1000, N=30, and initial topic "Laptop" is shown below in Table 1. The left-hand column contains results from similarity obtained comparing the full text of each article, while the right-hand column is derived from comparing only the text of internal links in respective articles. Additional details are given in Section 3 above.

| | |
|---|---|
| 0 [0.04966]: Kaypro | 0 [0.06209]: Commodore_PET |
| 1 [0.02261]: APL | 1 [0.05127]: Dell_Inspiron_E1405 |
| 2 [0.01446]: ExpressCard | 2 [0.04791]: Influence_of_the_IBM-PC_on_the_personal_computer_market |
| 3 [0.01434]: Commodore_PET | 3 [0.04314]: Personal_computer |
| 4 [0.01326]: Computer_case | 4 [0.04137]: Palm_IIIc |
| 5 [0.01163]: Wang_2200 | 5 [0.04108]: Wang_2200 |
| 6 [0.01110]: Personal_computer | 6 [0.03810]: Incremental_search |
| 7 [0.01022]: IMac | 7 [0.03711]: Lifebook |

**Table 1:** For a run starting at the "Laptop" article, the eight most similar articles to the "Desktop Computer" article for full-text (left) and internal links only (right). The numbers in square brackets are similarity metrics, where 0=perfect mismatch, 1=perfect match

## 4.4 Outgraph Rendering

The outgraph renderer generates a visual representation of the relationship and similarity between articles. Those in the K most similar matches for many articles will have a large number of connections, while those matching fewer other articles will have a smaller number of connections. The higher the similarity score, the bolder the line connecting two articles, as detailed in Section 3 above. In addition, articles with higher similarity scores are placed closer to an initial article. Two examples, demonstrating outgraphs for full-text and internal-link analysis for the initial article "Laptop", are shown in Figure 3 below. Note that the full text algorithm has identified two very significant articles, one above the center and one below, while the internal links algorithm has generated a more uniform outgraph.

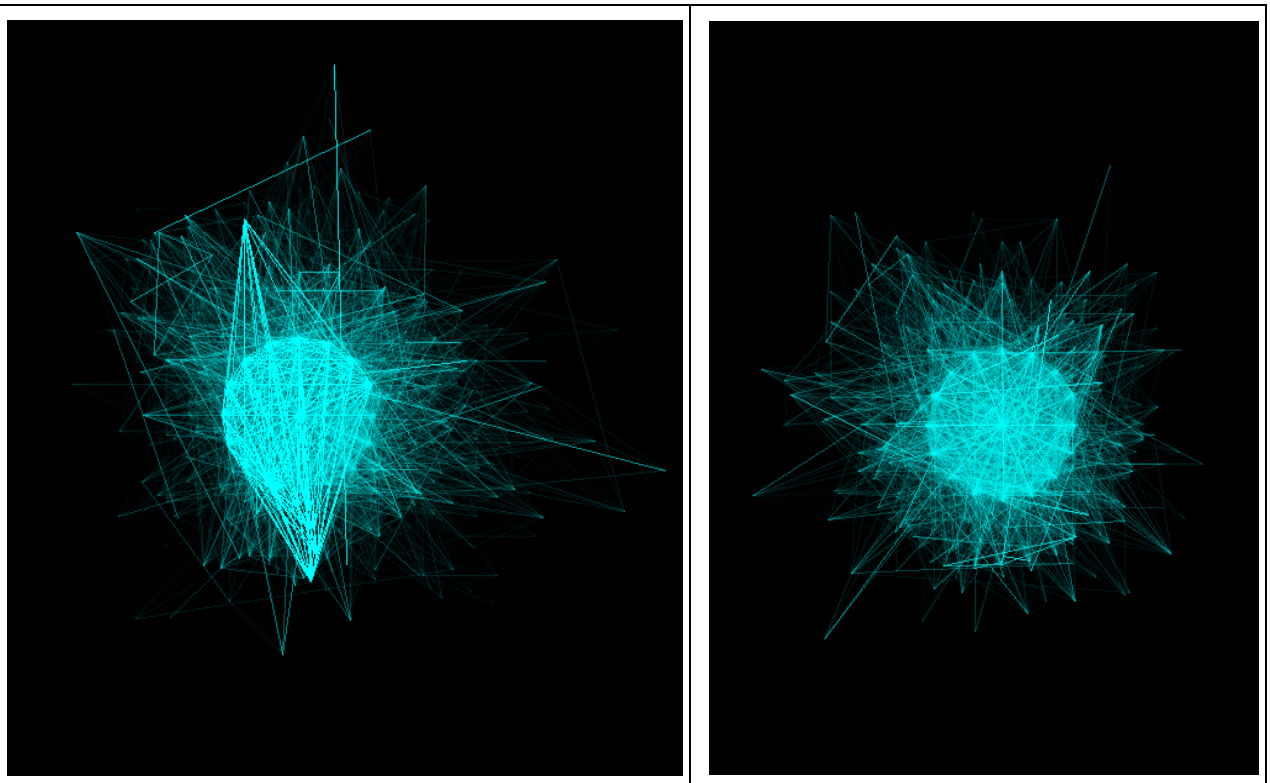

**Figure 3:** Outgraphs for the starting point "laptop" for full-text tokenization (left) and tokenization performed only on the text of internal links (right). Note the higher complexity of the full-text image.

## 5. Running the Program

To run the PHP program for this project, three files are required: `wiki_nlp.php` (the main executable), `func_misc.php` (containing non-NLP support functions), and func_wiki.php (containing NLP and Wikipedia-related functions). The parameters for a given run are set via a series of global variables at the top of `wiki_nlp.php`, as illustrated in Table 2 below. Once parameters have been set, the program is executed by running `php -f wiki_nlp.php`. The program will run, displaying its progress as exemplified in Listing 3. Once it completes, it will echo the most similar article to each article, write the K articles with the largest similarity to each article in a KM-line file `[$inittopic]_N[$M]_K[$K]_[i|f].sim`, where `$inittopic` is the initial topic, `$M` is the number of total articles to tokenize, $K, is the number of most similar articles to find for each test article, and the final character before the extension is "`i`" for an internal links run or "`f`" for a full text run. The outgraph is written to `outgraph_[$inittopic]_[$M]_[i|f].png`. Note that PHP5, php5-gd2, and php5-cli are required to run the program.

| Variable | Default | Explanation |
|---|---|---|
| **$inittopic** | "Ducks" | The starting topic for the analysis. This article must at contain at least one internal link, or the program will terminate. |
| **$maxarticles** | 1000 | The maximum number of articles to fetch and tokenize before calculating similarity. |
| **$maxperarticle** | 30 | The maximum number of articles linked from any initial article to be examined before choosing a new initial article. |
| **$K** | 16 | The number of most similar articles to store for each article. |
| **$textmode** | "f" | "f" for full text, "l" for internal links only |
| **$imsize** | array(1024,768) | A 2-element array of the (x,y) size of the outgraph in pixels. |
| **$distscale** | 64 | The nominal scaling basis for the distance between articles in the outgraph. |

**Table 2:** Settable parameters in wiki_nlp.php, along with their respective functions and default values.

## 6. Results and Analysis

In order to evaluate the two aforementioned methods of finding similar articles, namely calculating similarity based on the full text of the article and similarity based on internal links in the article, 10 volunteers were recruited. The volunteers were given a survey style test (see Appendix A) comprised of five different cases. Each case contained one main article and the top three similar articles as calculated by the two methods respectively. The volunteers were not told the actual nature of the two methods to prevent any form of bias. For each case, volunteers were asked to rate whether algorithm A or algorithm B seems to be better at finding articles similar to a given initial article. The results of the survey are tabulated and summarized below in Table 2. The questions were taken from two sets of runs, each set consisting of a full text session and an internal links session. The first set of runs used "Laptop" as the initial article, and parameters K=16, M=1000, and N=30; the second set used "Ducks" as the initial article with the same parameters.

| Name | case 1 | case 2 | case 3 | case 4 | case 5 |
|---|---|---|---|---|---|
| **volunteer 1** | A | A | A | A | A |
| **volunteer 2** | A | A | A | A | A |
| **volunteer 3** | A | A | A | A | A |
| **volunteer 4** | A | B | B | A | B |
| **volunteer 5** | A | A | A | A | A |
| **volunteer 6** | A | A | A | A | A |
| **volunteer 7** | A | A | A | A | A |
| **volunteer 8** | A | A | A | A | A |
| **volunteer 9** | A | A | A | A | A |

**Table 3:** Algorithm A (responses in light green) uses full text-based similarity, B (responses in light red) uses only internal links

As can be seen in Table 3, the ten volunteers preferred full text-based similarity over internal link-based similarity 94.0% of the time, or in 47 out of 50 trials. The overwhelming preference for the results of the full text algorithm confirms our initial hypothesis that this method would yield superior results.

## 7. Conclusion

A full PHP suite was completed that reads and tokenizes a large corpus of documents from Wikipedia, determines similarity rankings between articles based on an examination of either their respective full texts or the text only of internal links, then saves these results and renders a visual representation of the network of similarities. As hypothesized, the full text algorithm performed better, determining a more meaningful set of similar articles for each given article, as confirmed with blind human analysis. A sample set of ten humans chose the full text algorithm as superior in 47 out of 50 tests, and the internal links algorithm as superior in the remaining three tests belonging to a single volunteer. Notably, the internal links algorithm still extracts many meaningful similarity relationships, as can be seen in Table 1. Indeed, Wikipedia contains many orders of magnitude more articles than were examined in any run of this project, and the internal links method can parse and tokenize at least ten times as fast (1,700 seconds versus 12,000 seconds) as the full text method while using much less memory per article. Therefore, although the full text method produces superior accuracy, a practical application of the system demonstrated in this project might choose speed and efficiency at the expense of a moderate accuracy reduction by implementing the internal links algorithm.

## 8. Appendix A: Human Evaluation Survey

Hey Guys,

We need your help evaluating two methods that we have developed for a class. I'll refer to them as method A and B. Given a Wikipedia article (say... Article X), both these methods find other articles similar to Article X. Following is a list of main articles and the top 3 similar articles found by method A and B (ranking matters). We just need you to skim through the articles and tell us which method (A or B) in your opinion is doing a better job of finding similar documents.

**Note**: In most cases looking at the topic will help you arrive at your decision. If you are unfamiliar with the topic please click on the provided link to quickly skim through the article. =)

You can reply in the following format: Test 1 - A , Test 2 - B... etc

We know you guys are really busy... but we will really really appreciate it if you can send the results back to us by the end of this week =) Thanks!

Akshay

Christopher

**Test 1**

Main Article: Spec_sharp http://en.wikipedia.org/wiki/Spec_sharp
Method A results:

1. Sing_Sharp http://en.wikipedia.org/wiki/Sing_Sharp
2. C_Sharp_(programming_language) http://en.wikipedia.org/wiki/C_Sharp_(programming_language)
3. S60_(software_platform) http://en.wikipedia.org/wiki/S60_(software_platform)

Method B results:

1. C_Sharp_(programming_language) http://en.wikipedia.org/wiki/C_Sharp_(programming_language)
2. Sing_Sharp http://en.wikipedia.org/wiki/Sing_Sharp
3. Cω http://en.wikipedia.org/wiki/Cω

**Test 2**

Main Article: Partition_of_Vietnam http://en.wikipedia.org/wiki/Partition_of_Vietnam
Method A results:

1. Vietnam http://en.wikipedia.org/wiki/Vietnam
2. Governor-General_of_India http://en.wikipedia.org/wiki/Governor-General_of_India
3. Loyalist http://en.wikipedia.org/wiki/Loyalist

Method B results:

1. Birth_of_the_Italian_Republic http://en.wikipedia.org/wiki/Birth_of_the_Italian_Republic
2. High-Level_Conference_on_World_Food_Security http://en.wikipedia.org/wiki/High-Level_Conference_on_World_Food_Security
3. World_Food_Conference http://en.wikipedia.org/wiki/World_Food_Conference

**Test 3**

Main
Article: Birth_of_the_Italian_Republic http://en.wikipedia.org/wiki/Birth_of_the_Italian_Republic
Method A results:

1. Italian_Colonial_Empire http://en.wikipedia.org/wiki/Italian_Colonial_Empire
2. Italy http://en.wikipedia.org/wiki/Italy
3. National_Assembly http://en.wikipedia.org/wiki/National_Assembly

Method B results:

1. Partition_of_Vietnam http://en.wikipedia.org/wiki/Partition_of_Vietnam
2. High-Level_Conference_on_World_Food_Security http://en.wikipedia.org/wiki/High-Level_Conference_on_World_Food_Security
3. World_Food_Conference http://en.wikipedia.org/wiki/World_Food_Conference

**Test 4**

Main Article: Dell_Inspiron http://en.wikipedia.org/wiki/Dell_Inspiron
Method A results:

1. Dell_Inspiron_E1405 http://en.wikipedia.org/wiki/Dell_Inspiron_E1405
2. Dell_Studio http://en.wikipedia.org/wiki/Dell_Studio
3. Lundell_Settlement http://en.wikipedia.org/wiki/Lundell_Settlement

Method B results:

1. Lundell_Settlement http://en.wikipedia.org/wiki/Lundell_Settlement
2. Dell_Studio http://en.wikipedia.org/wiki/Dell_Studio
3. Dell_Inspiron_E1405 http://en.wikipedia.org/wiki/Dell_Inspiron_E1405

**Test 5**

Main Article: Wader http://en.wikipedia.org/wiki/Wader

Method A results:

1. Tern http://en.wikipedia.org/wiki/Tern
2. Charadriiformes http://en.wikipedia.org/wiki/Charadriiformes
3. Auk http://en.wikipedia.org/wiki/Auk

Method B results:

1. Auk http://en.wikipedia.org/wiki/Auk
2. Charadriiformes http://en.wikipedia.org/wiki/Charadriiformes
3. Society_(journal) http://en.wikipedia.org/wiki/Society_(journal)