Grouping Web Search Results by Semantic Likeness

Amittai Aviram (amittai.aviram@yale.edu)

25 May 2007

The proliferation of pages on the World Wide Web has made the use of Web search services indispensable, but the inherent ambiguity of natural human language makes it inevitable that Web searches will often return a mixture of results representing both the intended and unintended meanings of the search string. This problem has been known since the beginning of the World Wide Web, and methods of grouping or clustering pages by content have been developed (for instance, [1] and [2]). In this project, I attempted to help the user to find the desired pages by grouping Web search results based upon standard methods of measuring semantic likeness. Pages are represented by vectors of their word frequencies, weighted by the inverse document frequency (idf) for each entry, and the cosine of the angle between each pair of vectors measures the semantic proximity of the two respective pages. With several optimizations, computation of word frequency vectors and cosines was made reasonably efficient (about 10 to 15 seconds for 100 pages). An efficient iterative algorithm clusters the pages by nearest neighbor. In such nearest-neighbor clustering, it is often difficult to find an appropriate point to stop iterations, somewhere between the initial condition where each page is its own singleton cluster and a final condition in which all pages belong to a single cluster. Here, I offer a new solution, representing an intuitive compromise between two representations: a flat partitioning and a hierarchical tree of clusters.

1 Introduction.

Words and phrases in human language are inherently ambiguous, and Web search results reflect this ambiguity. A user who wishes to know about the Hilton hotel in Paris, France, might submit the search string *paris hilton*, and receive a handful of references to pages about the hotel, scattered among a swarm of references to pages about the celebrity Paris Hilton. Likewise, a user looking for the author the one-act play *The Sandbox* (Edward Albee) might submit *sanbox play*, and then have to sift through the many advertisements of children's sandboxes and metaphoric uses of *sandbox* in blogs to find the occasional page about the dramatic work. Indeed, it often happens that the same morphemes may represent both proper and common nouns. Although it would be difficult or impossible for a machine to infer which of the ambiguous meanings the user had intended, it would benefit the user to organize the search results into semantic groups. Since such a grouping does not require an understanding of the content of the pages but merely some measure of their relative likeness, a grouping method would not have to entail high-level artificial intelligence.

The relative frequency of words in the respective pages can provide an adequate basis for measuring their similarity. For instance, pages on the Hilton hotel in Paris will have a higher frequency of such words as *reservation*, *booking*, and *Champs Élysée* than pages about the celebrity Paris Hilton, which would, in turn, have a higher frequency of terms such as *video*, *revelation*, or *heiress*. By "words," we should consider word *stems* only, since we are only interested in the semantically- significant parts of words: *accommodate* and *accommodation* should be placed in the same equivalence class. Likewise, words that make little or no contribution to semantic content and do not help to distinguish pages by content should be kept from confounding the computations — so-called *stopwords* such as articles, pronouns, and conjunctions. Thus all n unique words (or word-classes) that occur in all pages in the set of pages returned from a Web search, after stemming and removing stopwords, define an n-dimensional space. The number of instances of each such word in a given document form an n-dimensional vector that locates that document in the space. The origin of the space corresponds to a hypothetical document that has no instances of any words other than stopwords. The size of the angle between any two documents' word frequency vectors in relation to the origin expresses the degree of similarity or difference in the vocabulary between the two documents and therefore in their semantic content. This angle may be measured by its *cosine*. The cosines, in turn, then provide the equivalent of weighted edges between the nodes representing documents on a complete, undirected graph. Partitioning this graph into subgraphs, consisting of the nodes closest to each other, will give us the grouping we seek.

A natural method for partitioning the graph is to start with each node (representing a page) in a cluster by itself, and then iteratively to join the nearest neighbors to form larger, disjoint clusters. The problem, then, is at what point we stop clustering. (See [3] for a treatment of this problem in a different context involving n-gram-based document similarities.) In this project, I have arrived at an approach that is neither a simple stopping criterion nor a full representation of the hierarchical tree of clutserings, from one cluster at the top down to n clusters for n pages, one page per cluster, at the bottom. Instead, I present a hybrid of these two reprsentations that manages to give the viewer some sense of coherency in the semantic groups about which he or she is likely to care the most. Put succinctly, this hybrid representation is a system of *nested* clusters. Therefore, taken as a whole, the clusters do *not* partition the set of documents; at each nesting depth, however, the clusters are disjoint.

2 Algorithm.

2.1 Procedural Sequence.

Starting with an interestingly ambiguous search string, such as *Paris Hilton*, my program executes the following steps:

- Data Gathering and Preprocessing.
  - Submit search string repeatedly to get a set of 100 results.
  - Fetch documents to which the search results point.
  - Reduce to lower case, stem, and remove nonalphanumeric symbols.
- Union of Word Sets.
  - Find the set W of all words occurring in all documents, excluding stopwords.
  - Find the *document frequency* for each word, which is the number of documents in which word w occurs at least once, for each word w in the union set W. These values will be used to compute the *inverse document frequency*, explained below.
- Word Frequency Vectors and Cosines.
  - Using the sorted list of words in W as keys, find the number of occurrences of each entry w for each document to form that document's word frequency vector.
  - Adjust the word frequencies by the corresponding inverse document frequencies (explained below).
  - Compute the cosine between each pair of vectors in $|W|$-dimensional space.
- Clustering.
  - Sort the cosines from highest to lowest value.
  - Given each pair of documents d_1, d_2 in the list of descending cosines, join the clusters to which d_1 and d_2 belong, if they are not already joined.
  - Repeat these steps until an appropriate stopping point, which, as explained below, will be when all n pages have been joined into one or another cluster.

- Output.
  - Convert the internal representation of nested clusters into a serial representation of groups.
  - Write the groups of corresponding URLs to an HTML document.

2.2 Technical Considerations and Optimizations.

The maximum number of search results that the Google search service will return is $1,0000$. Tests showed no significant difference in grouping results between 100 and $1,000$ results, with a substantial increase in time and space costs for the larger search.

The formation of the union word set and the tabulation of document frequencies are easily combined by means of a data structure such as a Python dictionary. A set of unique word entries is formed out of the content of each document. Then, if any word in this set is not already a key in the dictionary, it is added to the keys and its value is set to 1; if it is already in the dictionary, its value is incremented. Thus the keys of the dictionary constitute the union of all such sets, and the value assigned to each key is the number of documents in which the key term occurs at least once.

Word frequency vectors are sparse. Storing all the 0 entries can consume a lot of space, lead to disk swapping and thrashing, and degrade performance substantially, even in a small corpus of only 100 documents. By replacing simple vectors with lists of index-value pairs, I reduced the time cost of computing vectors about 50-fold. Overall, this optimization and the use of the dictionary reduced the total time for processing results from the search string *Paris Hilton* from about 550 seconds to about 100 seconds.

At the end of the tabulation of each word frequency vector, the values are multiplied by the respective inverse document frequency weight, explained below.

I used the simple Euclidean cosine:

$$\cos\theta = \frac{\mathbf{v}\cdot\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}$$

One such cosine must be computed for each pair of pages, so there are $\mathrm{C}(n,2) = (n^2 - n)/2$ such cosines for n pages, or $4,950$ for $n = 100$.

3 Adjustments to the Data.

Initially, I attempted to find clusters based solely on the cosines between vectors of raw word frequencies. The results were unsatisfactory: it was extremely difficult to find a metric that would accept documents into clusters when they should belong together and not when they should remain apart. One reason is that the mean cosine was very high. quite close to 1.0. Though the minimum was around 0.5, the preponderance of cosines was close to the mean, with a low variance. Indeed, the only semantic distinctions that seemed to show up clearly on my clustering documents by, say, counting merely the 300 highest cosines were of a sort where pages in one cluster were in a different language (Spanish) from pages in the other, or pages in one cluster were a warning about not having a Flash player, while other pages were about Paris Hilton — in either sense of the phrase. Three adjustments to the data made far more reliable results possible: removal of stopwords, discarding of cosines very close to 1, and weighting by the idf.

3.1 Removal of Stopwords.

I used a stopword list made available on the Web from Glasgow University's Department of Computer Science [1] . Words occurring in the stopword list were never included in either the union set (the keys of the document frequency dictionary) nor, accordingly, in any individual document's word frequency vector. Removal of stopwards lowered the mean cosine more than an 10-fold, to around 0.04.

[1]`http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words$`

3.2 Removal of High Cosines.

I discovered that the documents initially appearing closest to each other were merely duplicates or near-duplicates. (Near-duplicates would be documents with the same URL and substantially the same text, but perhaps some tiny change in some advertisement copy that rotates on different hits). I eliminated these by setting an upper limit of 0.9999. Interestingly, the next highest cosine to appear in descending order was substantially lower — about 0.75. We could then consider this lower value to represent genuine high proximity rather than identity. Given how much lower the mean now was, we can also see that the highest cosines did indeed represent significant relative likeness.

3.3 IDF.

The inverse document frequency allows us to give greater weight to those words that occur in only a few documents than words that tend to occur in many or all documents, even if they are not stopwords.[4] For instance, *Champs Élysée* occurs *only* in those documents about the Hilton hotel in Paris and *never* in documents about the celebrity Paris Hilton, so its presence should mean more than such words as *room*, *television*, or even *video*, which could occur in either semantic group. For this project, the inverse document frequency (idf) for a given word w_i, which occurs as the ith key when the union dictionary's keys are sorted, was the total number of documents in the complete set D ($|D| = 100$) divided by the number of documents d in which w_i appeared at least once:

$$\text{idf} = \frac{|D|}{|\{d : w_i \in d\}|}$$

This causes a word of particular rarity, such as *champs élysées*, to have a very high weight (close to 100), whereas a word occurring in all documents would have a weight of 1.

4 Clustering.

Ideally, for the present problem, we want some number k of clusters that group the documents into reasonable semantic classes, so that we can find the right documents and have them all listed together. It does not make much sense to use a k-means clustering algorithm, because we cannot know at the outset the value of k, which, presumably, would vary from search to search. Rather, the number of clusters should depend somehow on the characteristics of the documents themselves — how similar they are to each other and whether they naturally fall into distinct groups. It is finding these natural groups that is both our goal and our challenge.

A hierarchical clustering approach seems intuitively to bring us closer to this goal. Hierarchical clustering can be performed from the top down, starting with a single cluster of all n items and then dividing this cluster successively into subclusters according to relative distance. Alternatively, it can be performed from the bottom up: each item is its own cluster, and new clusters are then formed by drawing together, iteratively, those clusters that were closest on the previous iteration. It is this second approach that I took here.

Bottom-up, nearest-neighbor clustering introduces the question of *stopping criteria*: on what iteration do we stop clustering — presumably, before all the documents are drawn together into a single, total cluster? After a number of unsatisfactory attempts to find a good stopping point, based either on the absolute value of the lowest cosine allowed to join two documents into a cluster, the relative value of the lowest cosine as a ratio of the highest cosine or the mean, or the overall number of clustering iterations, I decided to abandon this approach altogether. Indeed, any stopping point that would render a "best" clustering for one search string, one set of documents, and one *user* is unlikely to generalize to other combinations of these three factors. The stopping point represents a degree of granularity, whose optimimum cannot be known with certainty for all cases in advance.

Instead, the clustering algorithm iterates until every document has been included into at least one cluster (other than itself). The result should be a complete tree of clusterings. The root of the tree is the single cluster that includes all documents. Branching out below it are the clusters and subclusters that represent semantically- related subsets of this union set, and having various sizes, with various numbers of clusters at each depth. At the bottom of the tree, the leaves are the n single-document clusters for the n original documents. Now, the path from many of the leaves passes through at least one cluster on its way to the root; but some leaves are connected directly to the root, because they are never absorbed into any subordinate cluster.

The tree takes on an internal representation as a series of nested lists. The outermost list is the root node of the tree, to which all documents belong. Within this list, there are some single documents — the ones with edges connecting them directly to the root — while most documents are in one or another of one or more second-level lists, nested within the first list. Inside each of these lsits, again, are some individual documents and some smaller subordinate lists. The innermost nested lists correspond to the most highly specialized groups of documents.

As each cluster grows, it absorbs neighbors that consist either of individual documents or of other clusters of documents. My algorithm is based on an intuitive distinction between these two kinds of growth. We start with a dictionary of n entries, whose keys are the indices of the documents, and whose values are singleton lists containing those same respective indices:

$$\{001 : [001], 002 : [002], \ldots, n : [n]\}$$

In the rest of the discussion below, I shall use the notation $d[x]$ for the value (list) in the dictionary under the key x. Thus, at the beginning,

$$d[x] = [x]$$

In addition, we have the complete lists of cosines in descending order, with the extremely high cosines cut off from the top, as mentioned earlier. Each cosine is represented by a pair of values, the first element fo which is a pair consisting of the indices of the two respective pages whose vectors form the angle measured by the cosine:

$$\text{cosines}[i] = \left((v, w), \cos\theta\right), 0 <= i <= n, \text{cosines}[i][1] >= \text{cosines}[i + 1][1]$$

The first (highest) cosine on this list ties the two closest documents together, say, a and b, so we join $d[a] = [a]$ and $d[b] = [b]$, with the result that $d[a] = d[b] = [a, b]$.

More generally, then, for any entry $\left((x, y), \cos\theta\right)$ in the *cosines* list, we do the following:

- If $|d[x]| = 1$ or $|d[y]| = 1$, then set $d[x] \leftarrow d[x] \cup c[y]$ and set $d[y] \leftarrow d[x]$. That is, if either cluster to be joined contains only one document, consider this an *incremental* growth of the cluster to which the single document will be added.
- If $|d[x]| > 1$ or $|d[y]| > 1$, create a new list to enclose, together, both $d[x]$ and $d[y]$. Set $d[x] \leftarrow [d[x], d[y]]$ and then set $d[y] \leftarrow d[x]$. Intuitively, when we join two *clusters* together (each of which contains more than one document), we make them into subclusters of a new, enclosing cluster. This saves the history of hierarchical clustering for the final result.

We continue iterating through the list of descending cosines until all n pages have been seen at least once. This means that all n pages have been joined into at least one cluster, even if only the catch-all outermost "cluster" that encloses all other clusters as well as otherwise unclustered individual document indices.

We now have a complex, nested list that, when sorted recursively, looks something like this:

$$[[[[069, 082, 099, 100]\,[003, 051, 075]\,012, 038]\,057, 063, 065]\,[[001, 019, 057, 083]\,[010, 011, 055, 096]]\,089, 090, 091]$$

(I standardized document indices to three digits. The above obviously includes only a small portion of all $n = 100$ documents.)

5 Output

Finally, to make this nested list useful to the user, we *serialize* it into a sequence of HTML lists. This sequence is *not* hierarchical but *flattened.* The hierarchy is thus reinterpreted into a simpler representation.

The intuition underlying this decision is as follows. Suppose document x occurs in the outermost "cluster" but in no other, nested clusters — for instance, document 089 in the illustration above. This document and the others in this outermost list have only the most general qualities in common with each other, the default likeness that had driven the search engine to return them in the first place. So they form a "group" of the most loosely-associated documents that have little in common other than the search string itself. Next, inside any of the clusters nested within the outermost cluster at the first (shallowest) nesting depth, again, any documents that do not occur in any more deeply-nested clusters form a group of documents that have in common whatever criteria most generally bind this cluster together. The rules for translating the nested list into a flat series of lists are as follows:

- At nesting depth d, look only for document indices that do not occur in any list at depth $d' > d$. If there are any such entries, use the URL of the lowest-numbered (highest-ranked) document as the header for an HTML list and then, if there are any remaining such documents, list them below the header.
- If there is more than one list at nesting depth d, start with the leftmost list in ascending sorted order (i.e., whose first document has the highest rank) and proceed rightwards through all lists at the same nesting depth d.
- Proceed to nesting depth $d + 1$.
- Stop when all documents have been represented in an HTML list.

Examples of the resultign output for the test search strings *paris hilton* and *sandbox play* are attached. Note that both output sets show a small, coherent subset, in a single list, of the documents pertaining to the "minority" meaning of interest: the Hilton hotel in Paris, France, and the one-act play by Edward Albee called The Sandbox, respectively.

6 Conclusion.

In this project, I attempted to address a real-world problem: the semantic classification of Web search results, given the inherent semantic ambiguity of many (or all) search strings and the likelihood that search results reflecting the desired meaning of the search string would come back mixed in with documents pertaining to irrrelevant meanings of the same search string, so that the user would have to hunt manually through the returned list. My proposed solution combines known data mining techniques with a novel method of assembling the results and then translating them into a usable representation. The complex nested list allows us to avoid the difficult or impossible problem of finding appropriate stopping critieria clustering in the general case and to preserve the history of hierarchical clustering in a compact form. Then, the serialization of this multiply-nested list into a sequence of flat HTML lists presents to the user groupings of documents that seem to reflect fairly well their intuitive relationship to each other and to the original multiple senses of the search string.

This project suggests several directions for further research:

- **Parallelization.** With the optimizations discussed above, and excluding the mere fetching and pre-processing of text documents, the program took only about 13 seconds to provide useful results on the search string *sandbox play*. The computation of a document's word frequency vector does not depend on that of any other word frequency vector, and cosines are likewise independent, which makes these tasks good candidates for parallel computing. My efforts to parallelize the algorithm at an earlier stage of development did not provide the expected speedup, but a second attempt should, in principle, bear better results.
- **Better labelling.** In the attached two samples, I used the URL of the highest-ranked document in the respective group as the header for that group. More helpful to the end user would be a semantically richer form of labelling. In an earlier experiment, I tried providing a list of the words most commonly occurring in a given cluster that do not also occur in other clusters. (This output page is attached as the third item.) Such individual label-words do not serve as an entirely reliable reference device, and they are often hard for a reader to interpret. Phrases would probably be more helpful.

- **Ordering.** It is likely that the user would wish to have at the top of the list the *most diverse* groups of documents, and then have groups closely related to those respective groups somehow accessible, seconarily, through them. For instance, the page of grouped results on *Paris Hilton* should have one group about the celebrity and a second group about the hotel. The first group would then hava a "further ..." link to all the other groups that are ultimately related to this meaning of the search string. As it is, my "hotel" cluster occurs as the eighth group on the output page.

Acknowledgments

Many thanks to Jiang Du, Edo Liberty, Professor Martin Schultz, and Professor Mark Gerstein of the Department of Computer Science, Yale University, for their help and encouragement in this project.

References

1. Cutting, D. R., Karger, D. R., Pedersen, J. O. 1992. Scattter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. Proc. 15th Int. SIGIR, 318-329.

2. Hearst, M., Pedersen, J. O. 1996. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. Proc. 19th Int. SIGIR, 76-84.

3. Savova, G., Therneau, T., Chute, C. Cluster Stopping Rules for Word Sense Discrimination. `http://acl.ldc.upenn.edu/W/W06/W06-2502.pdf`

4. Salton, G., Buckley, C. 1987. Term Weighting Approaches in Automatic Text Retrieval. ACM Technical Report TR87-881.

- http://en.wikipedia.org/wiki/Paris_Hilton (001)
    - http://www.who2.com/parishilton.html (015)
    - http://www.answers.com/topic/paris-hilton (036)
    - http://en.wikipedia.org/wiki/Paris_(Paris_Hilton_album) (055)
- http://www.imdb.com/name/nm0385296/ (002)
    - http://www.theregister.co.uk/2005/02/21/paris_hacked/ (059)
    - http://news.com.com/Paris+Hiltons+cell+phone+hacked/2100-7349_3-5584691.html (060)
    - http://www.tv.com/paris-hilton/person/207608/summary.html (074)
- http://www.imdb.com/title/tt0412260/ (003)
    - http://video.google.com/videoplay?docid=5615212328010933613 (016)
- http://www.askmen.com/women/models_150/171_paris_hilton.html (004)
    - http://parisfacial.ytmnd.com/ (010)
    - http://www.actressarchives.com/paris/ (019)
    - http://www.flickr.com/photos/sharl/sets/72157594266743665/ (022)
    - http://www.celebrities.com/blog/?cat=6 (032)
    - http://movies.yahoo.com/movie/contributor/1804456555 (039)
    - http://web.scc.losrios.edu/carberk/discuss/msgReader$114 (045)
    - http://thehiltonfiles.com/ (046)
    - http://www.forbes.com/lists/2006/53/U3HH.html (047)
    - http://www.celebopedia.com/paris-hilton/ (049)
    - http://money.cnn.com/2005/02/21/technology/personaltech/hilton_cellphone/ (050)
    - http://music.aol.com/artist/paris-hilton/745785/main (052)
    - http://ocaoimh.ie/2007/01/13/the-paris-hilton-video/ (069)
    - http://abcnews.go.com/GMA/story?id=3143374&page=1 (077)
    - http://abcnews.go.com/GMA/story?id=3143374&page=1&CMP=OTC-RSSFeeds0312 (078)
    - http://www.washingtonpost.com/wp-dyn/articles/A45305-2005Apr11.html (079)
    - http://www.kevo.com/profile/parishilton (083)
    - http://www.allposters.com/-sp/-Posters_i1222064_.htm (090)
    - http://www.nypost.com/seven/08292006/gossip/pagesix/pagesix.htm (093)
- http://www.myspace.com/parishilton (005)
    - http://thesuperficial.com/2007/05/paris_hilton_is_going_to_jail.php (098)
- http://parishiltonrecord.com/ (006)
    - http://www.stardoll.com/en/dolls/484/Paris_Hilton_2.html (054)
    - http://www.carlsjr.com/ontv/ (063)
- http://www.youtube.com/parishilton (007)
    - http://www.amazon.com/Paris-Hilton/dp/B000GDI3SW (067)
- http://www1.hilton.com/en_US/hi/hotel/PARHITW-Hilton-Paris/index.do (008)

  - http://www.hilton-paris.com/ (013)
  - http://travel.yahoo.com/p-hotel-472934-hilton_paris-i (058)
  - http://www.hilton-paris.com/paris/ (072)
- http://www.tmz.com/category/paris-hilton/ (009)
  - http://www.tmz.com/2007/05/04/paris-ordered-to-serve-45-days/ (030)
- http://houseofwaxmovie.warnerbros.com/podcast.html (011)
  - http://www.salon.com/mwt/feature/2006/12/11/paris_hilton/ (061)
  - http://news.independent.co.uk/world/americas/article2514447.ece (097)
- http://www.parishiltononline.net/ (012)
  - http://es.wikipedia.org/wiki/Paris_Hilton (017)
  - http://www.blogtoplist.com/rss/paris-hilton.html (037)
  - http://www.parishiltonsite.net/ (041)
- http://www.starpulse.com/Supermodels/Hilton,_Paris/ (014)
  - http://www.starpulse.com/Supermodels/Hilton,_Paris/Pictures/ (087)
- http://www.parishiltonblog.org/ (018)
  - http://www.news24.com/News24/Entertainment/Celebrities/0,,2-1225-2108_2105822,00.html (051)
  - http://www.funmunch.com/celebrities/models/paris_hilton/index.shtml (088)
- http://www.parishiltonsex.org/ (020)
  - http://www.news.com.au/heraldsun/story/0,21985,21677969-661,00.html (070)
  - http://www.parishilton.net/ (071)
  - http://defamer.com/hollywood/paris-hilton/ (073)
- http://www.hollywood.com/celebrity/Paris_Hilton/1125850 (021)
  - http://www.eonline.com/celebrities/profile/index.jsp?uuid=99001ac4-3361-49d4-9f0f-b39f4f190c9e (027)
- http://www.celebritywonder.com/html/parishilton.html (023)
  - http://news.yahoo.com/s/ap/20070503/ap_en_tv/paris_hilton (066)
  - http://www.foxnews.com/story/0,2933,269890,00.html (089)
  - http://rssfeeds.usatoday.com/~r/usatoday-LifeTopStories/~3/113934257/2007-04-26-paris-hilton_N.htm (099)
- http://www.bullz-eye.com/celebritybabes/paris_hilton.htm (024)
  - http://www.publispain.com/parishilton/ (028)
- http://www.parishiltonfan.org/ (025)
  - http://www.onipblog.com/ (062)
- http://chartreuse.wordpress.com/2006/09/18/why-paris-hilton-is-famous-or-understanding-value-in-a-post-madonna-world/ (026)
  - http://www.hotflick.net/celebs/paris_hilton.html (084)
- http://www.aceshowbiz.com/celebrity/paris_hilton/ (029)

- http://movies2.nytimes.com/gst/movies/filmography.html?p_id=358949 (031)
    - http://movies.msn.com/celebs/celeb.aspx?c=488412 (057)
- http://www.youtube.com/watch?v=6Mj776YiPCU&IA (033)
    - http://www.newgrounds.com/collection/parishilton.html (064)
    - http://traffic.livevideo.com/in.php?mid=457&sid=5&lid=618 (075)
    - http://www.engadget.com/2005/02/20/paris-hiltons-hacked-sidekick-releases-unedited-tell-all/ (076)
    - http://technorati.com/tag/Paris+Hilton (091)
    - http://technorati.com/tag/Paris%20Hilton (092)
    - http://today.reuters.com/news/articlenews.aspx?type=domesticNews&storyid=2007-05-06T234211Z_01_N03396944_RTRUKOC_0_US-HILTON.xml&src=rss&rpc=22 (096)
- http://www.cnn.com/2007/SHOWBIZ/TV/05/04/paris.hilton.ap/index.html (034)
    - http://www.cnn.com/2006/SHOWBIZ/TV/09/07/parishilton/index.html (081)
- http://www.msnbc.msn.com/id/18472845/ (035)
    - http://www.msnbc.msn.com/id/18024487/ (085)
- http://www.eonline.com/news/article/?uuid=791cb3de-09cc-4fda-88ea-c80a8d5645bc (038)
    - http://www.caplakesting.com/parishiltonautopsy/index.htm (086)
- http://www.moono.com/html/paris-hilton/paris-hilton-pictures.cfm (040)
- http://news.bbc.co.uk/2/hi/entertainment/5310416.stm (042)
    - http://news.bbc.co.uk/2/hi/entertainment/6624223.stm (043)
    - http://www.boingboing.net/2006/09/03/banksy_shopdrops_500.html (094)
- http://www.celebsrate.com/name/Paris+Hilton/ (044)
    - http://www.hissandpop.com/celebrities/h/parishilton/ (082)
- http://perezhilton.com/ (048)
    - http://www.hollywoodtuna.com/?p=2421 (053)
- http://www.ipetitions.com/petition/PH21781/ (056)
    - http://www.techcrunch.com/2006/08/22/paris-hilton-storms-youtube/ (100)
- http://parishilton.bro.ro/ (065)
    - http://www.parishiltonzone.com/pictures/ (068)
- http://www.theage.com.au/articles/2005/03/31/1111862521987.html (080)
- http://www.thesun.co.uk/article/0,,4-2006480340,00.html (095)

- http://www.nextag.com/play-sandbox/search-html (001)
  - http://www.nextag.com/sandbox-sand/search-html (040)
- http://en.wikipedia.org/wiki/The_Sandbox_(play) (002)
  - http://en.wikipedia.org/wiki/User:Tinned_Elk/Sandbox_play (017)
  - http://www.answers.com/topic/the-sandbox-play (048)
  - http://www.adveractive.com/games/sbhockey.htm (059)
  - http://adsabs.harvard.edu/abs/1995Sci...268.1277G (097)
- http://www.sortprice.com/search-CQ-Outdoor_Play-Sandbox (003)
  - http://www.sortprice.com/search-MCQ-KBtoys.com-Outdoor_Play-Sandbox (043)
- http://searchcrm.techtarget.com/originalContent/0,289142,sid11_gci1151518,00.html (004)
  - https://datatracker.ietf.org/documents/LIAISON/file241.doc (049)
  - http://www.itwire.com.au/content/view/7687/53/ (050)
- http://www.searchengineguide.com/wallace/2005/0209_dw1.html (005)
  - http://ajaxian.com/archives/using-applets-to-play-outside-of-the-sandbox (015)
  - http://www.searchenginechannel.com/2006/09/how-to-play-nice-in-the-google-sandbox (054)
  - http://www.thaiirc.in.th/seo/58193.php (062)
  - http://www.arkom.co.uk/news-article.asp?id=30 (083)
  - http://www.lilengine.com/google-search-engine/google/how-to-play-nice-in-the-google-sandbox-060906/page1.html (094)
  - http://dotnetslackers.com/CSharp/re-550_VB_and_Chash_Can_t_we_all_play_in_the_sandbox_nicely.aspx (099)
- http://www.truthout.org/docs_2006/041007A.shtml (006)
  - http://www.stwr.net/content/view/1780/36/ (038)
- http://www.amazon.com/b?ie=UTF8&node=166447011 (007)
  - http://houseofplus.com/index.php?showtopic=3212&view=getlastpost (066)
  - http://amazon.com/s?ie=UTF8&rh=n%3A166447011%2Cp_4%3ASpielstabil&page=1 (087)
  - http://wpilibrary.blogspot.com/2006/06/blog-sandbox-play-time.html (096)
- http://sandbox.msn.com/ (008)
  - http://www.searchenginelowdown.com/2004/07/play-in-microsofts-msn-search-sandbox.html (082)
- http://safesand.stores.yahoo.net/ornatplaysan.html (009)
  - http://parents.berkeley.edu/recommend/where2buy/sandbox.html (089)
- http://cgi.ebay.com/Wood-Swing-Set-Slide-Sandbox-Tower-Gym-Play-Center_W0QQitemZ270122858933QQcmdZViewItem (010)
  - http://cgi.ebay.com/NEW-LITTLE-TIKES-SAND-AND-WATER-PLAY-TABLE-SANDBOX_W0QQitemZ160119658686QQcmdZViewItem (069)
- http://cdbaby.com/cd/sandbox (011)
  - http://jspwiki.org/wiki/SandBox/PlayGround (014)
  - http://visitmix.com/Blogs/News/come-play-in-the-mix07-sandbox/ (024)
  - http://www.plosone.org/article/fetchArticle.action?articleURI=info:doi/10.1371/journal.pone.0000000 (027)
  - http://www.downloadtopc.com/related/windows_media.html (060)
  - http://sfbay.listpic.com/sby/bab/335159457_little_tikes_race_car_bed_pirate_ship_climber_sandbox_play_center_140_40_san_jose_east.html (078)
  - http://mccammon.ucsd.edu/~cmura/Sandbox/frames1top_selfref.html (081)
  - http://www.raphkoster.com/gaming/book/6.shtml (095)
- http://answers.yahoo.com/question/index?qid=20070519122353AAja6La (012)
- http://games.slashdot.org/article.pl?sid=07/02/08/2124217 (013)

- http://kids-outdoor-activities.suite101.com/article.cfm/make_an_easy_backyard_sandbox (016)
- http://www.networkworld.com/newsletters/sec/0913sec2.html (018)
  - http://review.zdnet.com/AnchorDesk/4520-6033_16-4207069.html (039)
- http://www.burtongroupblogs.com/jamielewis/2006/07/a_sandbox_to_pl.html (019)
  - http://www.cyberrentals.com/USA/Florida/Florida-South-Central-Gulf-Coast/vacation-condo-Sanibel-Island/p139267.htm (041)
  - http://www.goin2travel.com/sundial.htm (045)
- http://videogames.netscape.com/story/2007/03/28/play-in-a-virtual-sandbox (020)
- http://mail.python.org/pipermail/python-checkins/2006-November/057508.html (021)
  - http://mail.python.org/pipermail/python-checkins/2006-November/057512.html (022)
- http://www.flexwiki.com/default.aspx/FlexWiki/Play%20in%20this%20sandbox.html (023)
  - http://mediatedcultures.net/modules/wiwimod/index.php?page=Games+Nacirma+Play+and+What+They+Mean (079)
  - http://www.worldgolf.com/course-reviews/south-carolina/long-bay-club-golf-course-myrtle-beach-5055.htm (085)
  - http://www.earlychildhoodlinks.com/parents/sandandwater.htm (090)
- http://www.themoleskin.com/archives/the-sandbox-is-open-for-play/ (025)
- http://abu.aladdin.cs.cmu.edu/sandbox/published/HomePage (026)
  - http://abu.aladdin.cs.cmu.edu/sandbox/ (036)
- http://www.wackyinventions.com/12.html (028)
  - http://www.freepatentsonline.com/4515360.html (031)
- http://denver.yourhub.com/AuroraNorth/Stories/News/About-Town/Story~311752.aspx (029)
  - http://denver.yourhub.com/GreenValleyRanch/Stories/News/About-Town/Story~311754.aspx (030)
- http://www.backyardadv.com/accessories.php?c=Sandbox (032)
  - http://www.backyardadv.com/accessories.php?c=Double_Sandbox (055)
- http://moodle.edutech.org/course/info.php?id=9 (033)
  - http://www.drtoy.com/2004_3/2004_3_080.html (034)
  - http://www.sandbox.com/ (098)
- http://rwynn.com/ (035)
  - http://remarkk.com/2007/01/04/play-in-the-entrepreneurial-sandbox/ (047)
  - http://www.gkbledsoe.com/phpBB/viewforum.php?f=1&sid=537ea7612d33d9a42b06e3e793b56d4d (091)
- http://www.podcastalley.com/forum/archive/index.php/t-120857.html (037)
  - http://mail.gnome.org/archives/garnome-list/2006-March/msg00045.html (072)
- http://losangeles.craigslist.org/sfv/bab/337549706.html (042)
  - http://sacramento.craigslist.org/bab/334457475.html (067)
- http://www.seomoz.org/blog/google-sandbox-shari-thurow-comes-out-to-play (044)
  - http://everything2.com/index.pl?node=play%20in%20his%20sandbox (084)
- http://mail-archives.apache.org/mod_mbox/ws-tuscany-commits/200705.mbox/%3C20070510131316.3BDC21A9838@eris.apache.org%3E (046)
  - http://mail-archives.apache.org/mod_mbox/ws-tuscany-commits/200705.mbox/%3C20070510144248.36A251A9838@eris.apache.org%3E (064)
- http://www.manning-sandbox.com/message.jspa?messageID=30196 (051)
  - http://www.manning-sandbox.com/thread.jspa?messageID=43572 (052)
- http://discuss.mediacentersandbox.com/forums/thread/3243.aspx (053)
  - http://discuss.mediacentersandbox.com/forums/post/3364.aspx (100)
- http://shopping.yahoo.com/p:Scenery%20Solution%20Octagon%20Sandbox%2010'x10'x12:2002238732 (056)
  - http://shopping.yahoo.com/p:Scenery%20Solution%20Two-High%20Square%20Sandbox%204'x4'x12:2002238733 (057)

- http://www.digra.org/news_folder/sandbox (058)
- http://www.step2.com/product.cfm?product_id=1132&stp2ssid=18E092E1-1143-E489-2D4D7228E7A3A5D8 (061)
  - http://www.step2.com/product.cfm?product_id=1221 (076)
- http://nccsandbox.wikispaces.com/ (063)
  - http://www.xchangemag.com/articles/536/6ch4156659352.html (065)
- http://www.whitneybros.com/product.php?productid=174&cat=18&page=1 (068)
  - http://www.jacksons-camping.co.uk/kidstuff/sandbox.htm (075)
- http://mindmaps.wikispaces.com/Sandbox (070)
  - http://mlc2006.wikispaces.com/sandbox (074)
  - http://languagelinks2006.wikispaces.com/sandbox (080)
- http://www.uptimebot.com/blogsearch/play-sandbox/ (071)
  - https://www.ita.uni-heidelberg.de/internal/projects/sandbox (086)
- http://findarticles.com/p/articles/mi_hb4731/is_200607/ai_n17306861 (073)
  - http://www.shopping.com/xDN-toys--outdoor_and_playground_toys-stuff_kids_like_com (077)
- http://orgprints.org/wiki/Main/WikiSandbox (088)
  - http://robotics.eecs.berkeley.edu/wiki/pmwiki/pmwiki.php?n=Main.WikiSandbox (092)
- http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=9306758&dopt=Abstract (093)

Group 1

['clstoryurl', 'uas', 'msnbc', 'cnnstoryurl', 'bornontime', 'slide', 'weds', 'vegan', 'timenow', 'scolds', 'newsweek', 'hezbollah', 'dylan', 'builds', 'mock']

- http://www.cnn.com/2007/SHOWBIZ/TV/05/04/paris.hilton.ap/index.html (34)
- http://www.msnbc.msn.com/id/18472845/ (35)
- http://www.cnn.com/2006/SHOWBIZ/TV/09/07/parishilton/index.html (81)
- http://www.msnbc.msn.com/id/18024487/ (85)
- http://rssfeeds.usatoday.com/~r/usatoday-LifeTopStories/~3/113934257/2007-04-26-paris-hilton_N.htm (99)

Group 2

['boing', 'bbc', 'hmv', 'cds', 'xeni', 'virgin', 'specify', 'remixed', 'prankster', 'ms', 'leeway', 'defeat', 'cory', 'censorware', 'tampered']

- http://news.bbc.co.uk/2/hi/entertainment/5310416.stm (42)
- http://news.bbc.co.uk/2/hi/entertainment/6624223.stm (43)
- http://www.boingboing.net/2006/09/03/banksy_shopdrops_500.html (94)

Group 3

['podcatcher', 'salon', 'podcasting', 'attribute', 'query', 'portable', 'isgoodbrowser', 'podcast', 'variables', 'publicizes', 'invites', 'homebrewed', 'id', 'goal', 'expiration']

- http://houseofwaxmovie.warnerbros.com/podcast.html (11)
- http://www.salon.com/mwt/feature/2006/12/11/paris_hilton/ (61)

Group 4

['eiffel', 'elysees', 'li', 'span', 'miles', 'chambiges', 'travelocity', 'tower', '12px', 'orly', 'suggest', 'splendid', 'situated', 'seine', 'planner']

- http://www.hilton-paris.com/ (13)
- http://travel.yahoo.com/p-hotel-472934-hilton_paris-i (58)
- http://www.hilton-paris.com/paris/ (72)

Group 5

['large', 'yayyyyyyyyyyyyy', 'starpulse', 'shell', 'reckons', 'nut', 'hha', 'fakir005', 'eyedrain', 'dave11', 'britany', 'attends', 'sick', 'supermodels', 'ha']

- http://www.starpulse.com/Supermodels/Hilton,_Paris/ (14)
- http://www.starpulse.com/Supermodels/Hilton,_Paris/Pictures/ (87)

Group 6

['retrieved', 'who2', 'launch', 'cpm', 'chart', 'paternal', 'wikipedia', 'taura', 'sheppard', 'followed', 'hilton', 'albums', 'samples', 'began', 'vocals']

- http://en.wikipedia.org/wiki/Paris_Hilton (1)
- http://www.who2.com/parishilton.html (15)
- http://www.answers.com/topic/paris-hilton (36)
- http://en.wikipedia.org/wiki/Paris_(Paris_Hilton_album) (55)

Group 7

['su', 'fue', 'una', 'editar', 'que', 'ella', 'en', 'por', 'el', 'sentencia', 'para', 'algunos', 'se', 'es', 'y']

- http://es.wikipedia.org/wiki/Paris_Hilton (17)
- http://www.blogtoplist.com/rss/paris-hilton.html (37)

Group 8

['admin', 'toy', 'stumbled', 'shining', 'phpmyvisites', 'lynch', 'kitty', 'kinds', 'jetting', 'gumball', 'dimebag', 'darrel', 'cove', 'clone', 'blend']

- http://www.parishiltonfan.org/ (25)
- http://www.onipblog.com/ (62)

Group 9

['bodyguards', 'traveling', 'zipcode', 'silence', 'qs', 'myhollywood', 'lions', 'httpheader', 'gate', 'ddselect', 'cameos', 'starred', 'sorority', 'milestones', 'longtime']

- http://www.hollywood.com/celebrity/Paris_Hilton/1125850 (21)
- http://www.eonline.com/celebrities/profile/index.jsp?uuid=99001ac4-3361-49d4-9f0f-b39f4f190c9e (27)

Group 10

['chartreuse', 'skilton', 'merv', 'understanding', 'september', 'insightful', 'pm', 'um', 'daniela', 'cicarelli', '2006', 'marketing', 'vi', 'satan', 'mas']

- http://chartreuse.wordpress.com/2006/09/18/why-paris-hilton-is-famous-or-understanding-value-in-a-post-madonna-world/ (26)
- http://www.hotflick.net/celebs/paris_hilton.html (84)

Group 11

['4th', '8th', 'tmz', 'errormessage', 'config', 'player', 'digg', 'yahoo', 'validname', 'validemail', 'validcomments', 'rag', 'popsugar', 'popbytes', 'ovation']

- http://www.tmz.com/category/paris-hilton/ (9)
- http://www.tmz.com/2007/05/04/paris-ordered-to-serve-45-days/ (30)

Group 12

['cnet', 'scored', 'voting', 'pst', 'talkback', 'exhed', 'disagrees', 'abysmal', 'points', 'nickname', 'gamespot', '194', 'zdnet', 'techrepublic', 'mysimon']

- http://news.com.com/Paris+Hiltons+cell+phone+hacked/2100-7349_3-5584691.html (60)
- http://www.tv.com/paris-hilton/person/207608/summary.html (74)

Group 13

['classical', 'helpful', 'td', 'hr', 'blowout', '1px', 'styles', 'li', 'solid', 'amazon', '0px', 'selectedimageid', 'orders', '0pt', '0em']

- http://www.youtube.com/parishilton (7)
- http://www.amazon.com/Paris-Hilton/dp/B000GDI3SW (67)