13

Security of World Wide Web Search Engines

Massimo Marchiori
Department of Pure & Applied Mathematics
University of Padova
Via Belzoni 7, 35131 Padova, Italy.
Phone: +39 49 8275972. Fax: +39 49 8758596.
Email: max@math.unipd.it

Abstract

As all the recent market surveys witness, the World Wide Web (WWW) is expanding at a phenomenal rate, both in the number of users and in the amount of available information. This has made the World Wide Web one of the key fields for companies advertisement. On the other hand, advertisement on the web depends crucially on its *visibility*, i.e. on the possibility to be noticed by as many users as possible. The backbone of information management in the WWW is given by search engines, that allow users to access the enormous amount of information present in the web. Hence, advertisement has identified search engines as the major strategic component in web advertisement: it is essential for a company to appear on top tens of search engines, when the user asks for a product in their market field. Just by their automatic nature, search engines can be fooled by artificially manipulating web pages, so to make them rank higher: this relatively new phenomenon (called *sep*, after *s*earch engine *p*ersuasion) has now become so widespread to be a great problem: Indeed, on the one hand it provokes a great loss in advertisement profits for search engines maintainers, which are unwillingly giving free advertisement to companies that are sepping; on the other hand, it makes searching in the web extremely more difficult for users, since the performances of search engines are heavily affected by the artificial sep manipulation, making their evaluation mechanisms going wrong. In this paper, we thoroughly analyze the problem of security of search engines, giving a complete panoramic, and proposing various levels of security for search engines. Practically, we propose implementations of such security shields, that can be smoothly integrated in nowadays search engines: the original evaluation mechanism of a search engine is not modified, but it is seen as a black box, and simply integrated with security modules, in the form of pre- and post-processors.

Keywords

World Wide Web, search engines, advertisement, market competition, security, reliability.

D. Gritzalis (ed.), *Reliability, Quality and Safety of Software-Intensive Systems*

1 INTRODUCTION

The number of persons using the World Wide Web (WWW) is growing at such a fast rate that WWW advertisement has rapidly become one of the hot topics of the market, for its enormous strategical importance.

This explosion both of Internet hosts and of people using the web, has made crucial the problem of managing such enormous amount of information. As market studies clearly indicate, in order to survive into this informative jungle, web users have to almost exclusively resort on search engines (automatic catalogs of the web) and repositories (human collections of links usually topics-based). In turn, repositories are now resorting themselves on search engines to keep their databases up-to-date. Thus, the *crucial* component in the information management is given by search engines.

Indeed, search engines have become so important in the advertisement market that it has become essential for companies to have their web objects listed in top positions of search engines, in order to get a significant web-based promotion. Starting with the already pioneering work of (Rhodes, 1996), this phenomenon is now boosting at such a rate to have provoked serious problems to search engines, and revolutioned the web design companies, which are now specifically asked not only to design good web sites, but also to make them rank high in search engines. A vast number of new companies was born just to make customers web pages as visible as possible. More and more companies, like Exploit, Allwilk, Northern Webs etc., explicitly study ways to rank high a web object in search engines.

We call this phenomenon *sep* (cf. Marchiori, 1996b), which is a neologism standing for *s*earch engine *p*ersuasion (incidentally, this neologism has analogies with *sep*ia, the inky secretion that is used by cuttlefishes to make their opponents blind). Sep is therefore a way to fool the evaluation mechanisms of search engines, in such a way to get "free advertisement". This initially sparse phenomenon is now so common that it is provoking serious problems in search engines, since this artificial pumping of scores has the effect of making the evaluation mechanisms of the search engine almost useless, confusing the user. A big and common problem is for instance the so called *flattening* effect, occurring when several items rank with the highest score.

Besides the degradation of performance, this is also a big economic damage for maintainers of search engines; search engines, to survive, need money from advertisement, which is either provided by banners, or, like recently OpenText has done, arriving to the point to sell "preferred listings", i.e. assuring a particular entry to stay in the top ten for some time, (cf. Wingfield, 1996).

Therefore, sep provokes a serious economic damage, which mines in a sense the same survival of search engines, since it is a form of stealing free advertisement.

In this paper we analyze the problem of security of search engines from sep attacks. We present a panoramic of the current situation, and of the kinds of sep techniques currently more used. Next, we propose several kinds of security, and explain how they can be practically implemented. A fundamental point is that we separate this problem from the problem of computing a good evaluation function. Indeed, the main problem in getting secure search engines is that their maintainers have already spent time and resources to obtain evaluation mechanisms (so called score functions) that properly evaluate the informative contents of WWW objects. The task of rebuilding from the scratch a new score function that also takes into account security issues is thus extremely expensive.

Here, we propose ways to increase the security level of search engines without actually touching such existing score function, but simply adding some components to it, in the forms of pre- and post-processors. This means that the original score function is treated as a black box, and does not need any modification at all.

Moreover, we develop the analysis of security in such a way that all the proposed forms of security can be freely *composed*, i.e. one can combine several levels of security without risks of clash. Practically, this means that a very effective level of security can be obtained by composing several different modules, each taking care of some particular security aspect.

This separation of concerns has also a big impact on software maintainment costs, since the evaluation function and all the security components are kept separate.

2 PRELIMINARIES

In general, we consider a *web structure* to be a partial function from Uniform Resource Locators (URLs) to sequences of bytes. The intuition is that for each URL we can require from the web structure the corresponding object (an HTML web object, a text file, etc.). The function has to be partial because for some URL there is no corresponding object.

In this paper we consider as web structure the World Wide Web structure $\mathcal{WWW}$.

A *web object* is a pair (*url*,*seq*), made up by an URL *url* and a sequence of bytes $seq = \mathcal{WWW}(url)$.

In the sequel, we will consider understood the $\mathcal{WWW}$ web structure in all the situations where web objects are considered.

Each search engine is usually asked to return web objects that are relevant to a certain query, returning a *ranking*, that is a sequence of web objects, ordered with respect to their relevance. For simplicity, we will consider the query as a finite string, said the *key*.

In order to produce such rankings, a search engine needs a so called *score function*, which we assume is a function taking a web object and returning a nonnegative real number, its *score*. Its intuitive meaning is that the more information, the greater the corresponding score. Note that the score functions are of course assumed to depend on a specific key, that is to say they measure the informative content of a web object with respect to a certain key. In the sequel, we will always consider the key to be understood. As said in the introduction, the phenomenon of *flattening* occurs when in a ranking the first items have all the same score.

In the paper we assume that every search engine has its proprietary *main score function*, denoted with SCORE. The security of the main score function will be improved by appropriate *pre-* or *post-processors*, that is to say respectively by first applying another function on the web objects and then using SCORE, or by using a score function that can make function calls to SCORE.

Observe that in order to have a feasible implementation, we need that all of these functions are bounded. So, we assume without loss of generality that SCORE has an upper bound of 1.

Finally, we will often consider understood a situation of *heavy competition* (aka *market pressure*), that is to say, we assume that there are several competitors in the same market

area, each willing to have web advertisement by ranking with high scores its sites in search engines.

3 FAST UPDATE VS. FAST TUNING

Sep is intrinsically an *adaptive* process. In order to gain insights on how to get a high score, one starts with some trials (usually motivated by looking at other web objects with high score), gets the score response by the search engine, then modifies the web objects, observes the new search engine responses, and so on, until a satisfactorily high score is reached.

There is however a point that has to be taken into account: search engines do not provide immediate response. Because of the huge amount of information present on the web, each search engine needs a certain amount of time to complete a "refresh" (update) of its data.

Hence, effectiveness of sep is tightly linked with the refresh time of the search engine: the shorter the refresh time, the higher the effectiveness of sep.

So, in order to contrast sep, a tactic would be to set a quite big refresh time.

However, the refresh time is also becoming one of the *major advertising factors* of a search engine, being an obvious index of how good is the search engine to keep itself updated. Hence, in this last period there has been a rush to diminish the refresh time, which now ranges in the best cases from 1–2 months to 1–2 weeks, (cf. Sullivan, 1996).

So, we are faced with a bad situation that enforces sep: on the one hand, the refresh time is getting shorter and shorter due to the market pressure; on the other hand, a shorter refresh time makes sep more and more effective.

4 SPAMDEXING

The easiest, and more common, form of sep it the so called *spamdexing*, that is to say the artificial repetition of relevant keys in a web object, in order to increase its relevance. This, as said, has led to a bad performance degradation of search engines, since an increasingly high number of web objects is designed to have an artificially high textual content. In addition, spamdex is also easy to perform, and so it has rapidly become the primer form of sep adopted by companies. The phenomenon is so serious that search engines like Infoseek and Lycos have introduced penalties to face spamdexing, (see e.g. Murphy, 1996; Sullivan, 1996; Venditto, 1996).

To test the effectiveness of spamdexing, we have set up a fake commercial web object. This web object was submitted for inclusion in search engines, but had no other web object of ours pointing to it. This way, we were sure that every access to this web object was due either to a search engine or to a user using it, and not to users wandering by our site.

The initial result was that, not surprisingly, the web object got no hits at all in the first two months. Then, we modified the web object using spamdexing, just repeating twenty times each relevant keywords. The result was that, after a period of stale of roughly two weeks, due to the time refresh interval of search engines, our web object immediately got a huge boost of hits, that went on week after week. The situation is reported in Figure 1,

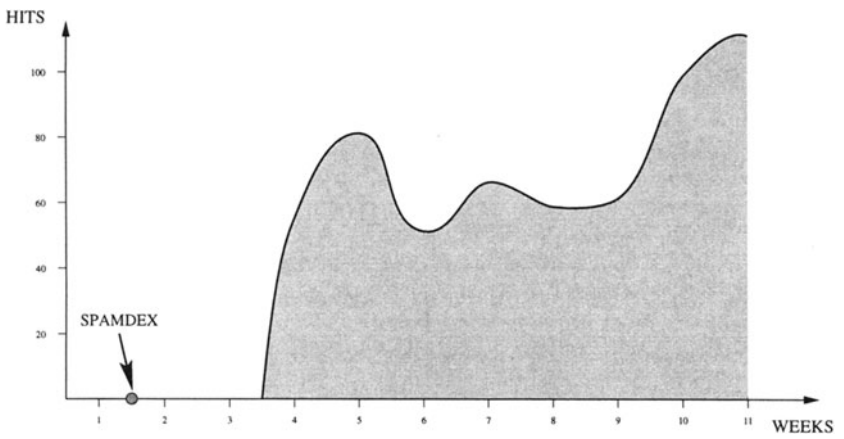

Figure 1 Effectiveness of Spamdexing.

showing the number of hits of our commercial with the evolving of time, where accesses by search engines have been filtered out (i.e., hits of search engines spiders have been ignored in the count).

4.1 Penalties

Spamdexing is the first sep technique to have been developed, and as such it is rather well known by search engine maintainers. To date, as said before, two search engines, Infoseek and Lycos, have tried to combat this phenomenon by introducing *penalties*: Once a key has too many repetitions in a web object, its score is penalized, e.g. by setting it to zero.

The penalties approach is rather drastic, and it is however an extremely poor choice for at least two reasons.

The first concerns security, and it is related to the adaptive nature of sep. Penalties only set a bound on the number of repetitions, so they only compress the range of possible high scores. Consequently, this means that in the short period, naïvely spamdexing web objects are penalized, but as soon as this is realized, these web objects are retuned to stay below the penalty border. Thus, penalties do not have the effect of avoiding flattening, but just to amplify it!

This is confirmed by a practical study of ours: among the search engines, Lycos and Infoseek are those with statistically the *highest* occurrence of flattening.

The second reason concerns the reliability of the score function, and will be examined in the next subsection.

4.2 Impact on Reliability

So far, we haven't given a general definition of penalty, but only seen some specific instances. It is indeed possible to give a formal definition of penalty w.r.t. a generic sep phenomenon (like e.g. spamdex), based on a sophisticated notion of *stability* of the score

(cf. Marchiori, 1996b). However, this notion is rather long and technical, so for lack of space and better readability we will simply stick to the intuitive notion: first, the score of a web object is evaluated by a suitable score function; then, a penalty test is done on the web object: if it is passed, a penalty occurs, in which case the web object gets a low score; otherwise, the web object gets its original score without modifications.

Now we turn our attention to the second big problem of penalties, their reliability. This problem occurs when the sep phenomenon we want to penalize cannot be always detected with certainty: in other words, when passing a penalty test does not always imply the presence of the phenomenon. For example, consider the spamdex case. While five consecutive repetitions of a key may reasonably be almost a certainty of spamdex, a repetition of ten times of the same key in different places of the web object is not.

The situation is critical, because of the following argumentation:

1. In order to refrain from penalizing web objects which are not sepping, search engines maintainers have to make the detailed penalties specification public, but
2. they cannot pretend a generic user will ever look at their penalties specifications, and
3. having all the details of the penalties specifications public makes them completely useless from the security viewpoint.

Therefore, the rationale is that *penalties should not be used, unless the chance that a non-sepping web object is penalized is very small.*

In the particular case of spamdexing, the situation is even worst, since the bound of repetitions activating a penalty has to be low in order to be effective against spamdexing, which means that a huge number of web objects that are not spamdexing will be penalized. Indeed, empirical studies show the lack of precision of Lycos and Infoseek, (cf. e.g. Churilla, 1996; Leonard, 1996; Liu, 1996; Sullivan, 1996).

Thus, the penalties approach is extremely dangerous. The best solution, we think, is in all cases not to give too much relevance to the simple *frequency score function*, i.e. not to base the score exclusively on the (possibly weighted) number of occurrences of keywords (like, unfortunately, many search engines do, cf. Sullivan, 1996), but to use more sophisticated techniques. We will see later in Section 7 an example of how this can be obtained using a post-processor. A simple pre-processor tailored to face spamdexing can instead be obtained by using the *truncation approach*, as we will show in the next subsection.

4.3 Truncation

A simple way to partially face spamdexing is to simply ignore the occurrences of a key that are beyond a certain upper bound. This technique can be implemented as a pre-processor, that deletes an occurrence of a keyword from a web object after the repetition bound has been reached. The only problem is to set a suitable repetition upper bound: a too low one is likely to cut too much relevant information, while a too high one risks to be uneffective against spamdex. We tested this approach with WebCrawler and saw immediate benefits (the best results were obtained by setting the upper bound in a range from 5 to 7).

5 GHOST COMPONENTS

A ghost component in a web object is a part of the code where text can be inserted, that in all likelihood will never be observed by a user when viewing the web object using a web browser. For instance, HTML *comments* are ghost components, as well as META description tags.

Comments and META description tags aren't the only possible ghost components, although they are by far the most widely employed for sep, since there are many others, which can roughly be grouped into two main categories.

The first category leaks within tag attributes. For instance, the ALT tag attribute will in general not be displayed unless image graphic is disabled (or not present, like in Lynx). Also, browsers are error-tolerating with tag attributes: if an unknown tag attribute is found within a tag, it is simply ignored (this has been extremely useful for the evolution of HTML, since new features can be added to better model the layout, without losing compatibility with older browsers); thus, every tag attribute not present in the HTML specification is a ghost component.

The second category gathers together esoteric ghost components that are possible via a smart use of HTML. For instance, setting a *local ink color* equal to the paper color produces invisible text. Other examples are positioning of the spamdex far away from the visible screen size, or in the NOFRAMES part.

Why are ghost components of interest in the study of sep? The answer is that since these parts are invisible to the final user, they can be freely manipulated without any constraint, thus making easier to perform sep. Visible components, instead, affects the layout of a web object, which is itself of primary importance from the advertisement viewpoint. Consider for instance spamdex: spamdex is an artificial addition to the content of a web object, just to fool search engines, but it can ruin the layout of a web object if inserted as normal text. That's why ghost components are the perfect place for a spamdex: they make spamdex invisible to the user.

A nice confirmation of this fact stems from a study that we have performed on the structure of spamdexing web objects: we have seen that statistically over 80% of them completely concentrates the spamdex in the ghost components of a web object (!).

Note that all the aforementioned kinds of ghost components are not only "potential", but actual source of spamdex, since in our studies we have found for each of them corresponding spamdex examples in the World Wide Web.

Ghost components can be syntactically identified, hence an easy solution to face their usage for sep is to get rid of them when evaluating the score of a web object. This is of course a transitory solution, since after having realized that manipulating ghost components is no more effective, one can turn to more sophisticated forms of sep. However, such tuning will be at least a non-trivial task, since spamdex will heavily impact on the layout of the web object.

The careful reader may have noticed a problem with the above solution: we also get rid of the META description tag. Most of search engines already ignore it, however others, like Alta Vista, InfoSeek and HotBot, use this tag just to infer more relevant keys for a web object. So, this method can in these cases be applied to all the ghost components but for META tags. This, however, risks to give a less satisfactory solution, since we saw that in a number of cases there were web objects using (also) this tag to spamdexis. If we do

not want to throw away this information, we can act with penalties on the sole content of the META description tag. This is not in contradiction with what we have said previously against penalties (see Subsection 4.2), since here the chance that a non-sepping web object is penalized is negligible: The fact is that we are not applying a penalty to the whole web object, but only on an optional tag, which only use is just to help search engines. So, point 2. of the "argumentation" previously mentioned in that subsection does not hold any more: anyone using a META description tag should look at how it is used by the search engine (and thus, making the penalty description public is this time a reasonable solution).

6 THE PROBABILISTIC APPROACH

An effective way to face sep can be obtained by using random techniques. As a limit case, suppose to add a post-processor to a search engine, such that every produced ranking is randomly shuffled, and then passed to the requesting user: this readily makes sep rather useless. Of course, this is a limit case since the information of the original ranking is not taken into proper account; however, it gives some intuition on the use of random techniques for security purposes.

The idea of the *probabilistic approach* is so to lower the chance of sep below certainty: in other words, to make unsure the success of sep.

We will show how probabilistic security can be ensured with a post-processor, in such a way that *its effectiveness grows proportionally to the market pressure.*

First, we fix a "lossiness parameter" ε, with the intended meaning that two scores s_1 and s_2 are indistinguishable if $|s_1 - s_2| \leq \varepsilon$. Then, given a ranking $r_1, r_2, \ldots, r_k$, we group its elements into "clusters", gathering together indistinguishable elements, in the following way. Define the *top cluster* of a ranking $\rho = r_1, \ldots, r_k$ as $\mathcal{T}(\rho) = \{r_i \in \rho : |\textsc{Score}(r_1) - \textsc{Score}(r_i)| \leq \varepsilon\}$. Then we can split any ranking ρ into a disjoint set of clusters simply by repeatedly extracting the top cluster, that is to say the first cluster is $C_1 = \mathcal{T}(\rho)$, the second $C_2 = \mathcal{T}(\rho \setminus C_1)$, and so on, until for some j we get the empty cluster $C_j = \emptyset$. Note that the maximum number of (non empty) clusters into which a ranking $\rho = r_1, \ldots, r_k$ can be split is k, in which case every cluster is simply a singleton ($C_i = \{r_i\}$). This situation corresponds to the case where there are no indistinguishable elements in ρ.

Once we have split a ranking into several clusters, we can shuffle each cluster (accordingly to our interpretation, the elements in a cluster are equally relevant).

The shuffling can be performed in two ways: either completely randomly, or taking into account the original score of each element. In the first case, a cluster $c_1, \ldots, c_n$ is shuffled by taking a random permutation π of $1 \ldots n$, obtaining $c_{\pi(1)}, \ldots, c_{\pi(n)}$. In the second case, the permutation is not completely random: roughly speaking, the chance an element has a low rank is inversely proportional to its score, that is to say the higher the score, the less the chance that its position in the local ranking given by the cluster is lowered. The formal procedure that we have implemented to achieve this more sophisticated shuffling is in the complete documentation of this paper.

Now, let us analyze the behaviour of this approach with respect to probabilistic security. The intuition, as said, is that web objects with similar scores may be switched. This from

the security viewpoint means that under heavy competition, a sepping web object does not have the certainty to be at the top, since it can be lowered by shuffling. Even a low market pressure can be in principle balanced with a higher lossiness parameter, although the choice of ε has to be extremely careful, since there is the danger that the shuffling interferes too much with the original scores of the search engines.

The less risky choice is to set $\varepsilon = 0$, so that only elements with equal score can be shuffled; this way we are not changing the information produced by the original search engine score function.

Note that we can also decide to limit the shuffling to the top cluster only, if we are solely interested in the security of the top elements (the most relevant from the marketing viewpoint). This way, setting $\varepsilon = 0$ has the nice effect that randomization takes place *if and only if* flattening is present.

6.1 More Randomization

We can improve the probabilistic security of a system further, by making even harder the general sep adaptive process. This can be achieved by randomly changing the score function each time a key is submitted to a search engine, or by randomly changing it at some time intervals, e.g. every week. This way, the sep adaptive process is likely to be *extremely hard* even in *absence* of market pressure, because one has to reconstruct the behaviour of several different score functions randomly alternating.

This security method can still be achieved with pre- or post-processors, for instance by choosing among those presented in this paper (those of this section form already a complete family, when varying ε and the clusters to shuffle).

7 THE UNIQUE-TOP APPROACH

The effectiveness of spamdexing relies on the naïve assumption by search engines that "frequency implies relevance", i.e. that the relevance of a key is proportional to the number of times it occurs in a web object. On the other hand, there is another dual kind of approach when evaluating the relevance of a key, which can be summarized with the slogan "high percentage implies relevance". This is the so-called percentage score function: the relevance of a key is given by the percentage it appears in a web object. This approach is currently used in WebCrawler. As far as a correct measurement of the relevance of a key is concerned, the percentage score function is *per se* a rather poor approach. A first point is that it penalizes too much the relevant keys. It is extremely rare that a web object has only one relevant key, while with this approach there is an unnatural penalization of the keys. Also, the percentage score function completely discards the "frequency" information, which is nevertheless an important indicator of the relevance of a key Our tests on the pure percentage score function have shown that it is absolutely not good in order to measure the relevance of a key.

So, why should one be interested in this approach? The distinctive feature that makes it interesting is that we can extract from it a general security rationale, that is what we call the *unique-top approach*; intuitively, it means that there is at most one key giving the top score for a web object. This approach is a weak form of security: it doesn't prevent

sep, but it ensures that sep is possible *for one key only*, and not on multiple keys, thus limiting by far the sep scope (it is by now well known in web advertising the need to differentiate the keys in order to reach the widest possible audience, see e.g. Northern Webs, 1996). We said "intuitively" because this is *not yet* sufficient for real security. The problem is that a user can have complete control over his own local site (whether two sites are different or not can be inferred from their IP addresses, although there are some subtleties involved in this issue). Indeed, one can set up on his site multiple copies of the same web object, and then tune each of them for a single key. This way, one can reach the top score for k keys by producing k ad-hoc web objects. The way to overcome this problem is to push the "unique-top" principle one step further, that is to say: there is at most one key giving the top score for *all the web objects within a site*.

We will now examine how unique-top security can be obtained via post-processing.

First, we have to ensure that, on each web object, sep is possible for one key only.

A solution is to combine the main score function with the percentage approach: this can be easily done by using a linear combination of the two. That is to say, if SCORE and PERCENTAGE denote respectively the main score function and the percentage score function, then we can use as score function the linear combination

$$\alpha \cdot \text{SCORE} + \beta \cdot \text{PERCENTAGE}$$

with $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$, $\alpha + \beta = 1$.

Then, we have to ensure that sep is possible for one key only on one of all the web objects within a site.

This can be done in practice by acting with another post-processor, by simply penalizing all but one of the web objects from a same site ranking with top score. Note that although we use penalties (cf. Subsection 4.2), the situation here is relatively safe, since the chance that two web objects belonging to the same site rank high, they are not sepping, and one is not linked to the other, is *extremely small*: tests that we have performed indicate that it is at least below 1%.

Also, this extremely small chance that a non-sepping web object is penalized can be made even smaller by *randomly* choosing the web objects that have to be penalized (this effect can be achieved also via a combination of this approach with the probabilistic approach of Section 6).

Note that the unique-top post-processing also improves the general security of the web object with respect to spamdex, since with the addition of the percentage score function, spamdexing a single key does not guarantee any more a high score from simple "frequency" score functions, since now there is also the percentage component; thus, making the score of a single key higher by spamdexing implies that the score of all the other keys *must decrease*.

8 THE HYPER APPROACH

This last approach reverses in some sense the common strategies against sep that one would expect (and that we have followed so far). This approach works better if its specifics are made *public* (!), since it encourages sep instead of limiting it.

This apparent paradox is clarified once the approach is explained: the idea is that a web object, in order to get a high score, has to *advertise the competitor web objects*.

Therefore, one is faced with a dilemma: either do not advertise the competitors, which means having a low score, or getting a high score (so a good advertisement), but procuring an automatic good advertisement for the competitors too.

Thus, we have that the market will oscillate between two attractors: in the first, no one is advertising the other competitors; in the second, everyone is advertising all the other competitors. Since adding an advertisement to a competitor increments the score, the second attractor is much stronger than the first, as it is trivial to see (a formal analysis, within a specific implementation, is in the complete documentation of the paper), hence the more likely situation is that after some time everyone is performing heavy sep, with the result that the user has at its disposal a complete panorama of the market, offered just by each competitor. This way, search persuasion has the effect of reshaping the web, by considerably improving its connectivity. Indeed, as noticed in (Bray, 1996), at present the inter-connectivity is rather poor, since almost 80% of sites contain no link to other sites (!), and a relatively small number of web sites is carrying most of the load of hypertext navigation.

In the following we will describe a way to obtain this form of "hyper security" using a post-processor.

The bare idea is to add to the main score function another component, the so-called "hyper information" (denoted by HYPER) which takes into account how much advertisement to the competitors the web object is doing. Maintainers can keep details of their SCORE function hidden, but are encouraged to make public the fact they are employing hyper information (although this doesn't mean they should provide exhaustive details on how HYPER is effectively implemented).

The hyper information was first developed in (Marchiori, 1996a) with another purpose (namely, to improve score functions). Here, we will only focus on the security aspects of the hyper information. For the sake of clarity, we will first give a simplified definition of the hyper information, and then proceed to refine it.

We start by isolating the two major points in the definition of the hyper approach. They are: a) to advertise another web object, and b) the competitors.

The "to advertise another web object" stems directly from the domain we are talking of: it simply means to have in the web object a link to the other web object. The notion of "competitor" is instead subtler: The idea is to identify competitors with web objects having a high SCORE. This approximation is readily good in case of market pressure.

Thus, consider the simple case where we have only one link from a web object A to a web object B. We could thus set the hyper information of A to be SCORE(B). Thus, we add score proportionally to how much B is in competition with A.

This approach is attracting, but not correct, since it raises problems of reliability of the score. For instance, suppose that A has almost zero SCORE, while B has an extremely high SCORE. Using the naïve approach, A would rank higher than B, while it is clear that the user is interested in B and not in A.

The problem essentially is that the information pointed by a link cannot be considered as *actual*, since it is *potential*: for the user there is a *cost* to retain the information pointed by a link (click and... wait).

The solution to these two factors is: the contribution to the hyper information of a web

object at depth k is not simply its SCORE, but it is its SCORE diminished via a fading factor depending on its *depth*, i.e. on "how far" is the information for the user (how many clicks s/he has to perform).

Our choice about the law regulating this fading function is that information fades exponentially w.r.t. the depth, i.e. the contribution to the hyper information of A given by an object B at depth k is $\Phi^k \cdot \text{SCORE}(B)$, for a suitable fading factor Φ $(0 < \Phi < 1)$.

Thus, in the above example, the hyper information of A is not simply $\text{SCORE}(B)$ but $\Phi \cdot \text{SCORE}(B)$.

As an aside, note that the main score function can be seen as a special degenerate case of hyper information, since it is $\text{SCORE}(A) = \Phi^0 \cdot \text{SCORE}(A)$ (viz., the object is at "zero distance" from itself).

Now we turn to the case where there is more than one link in the same web object. So, suppose you have the situation where a web object A has links pointing to n different web objects $B_1, \ldots, B_n$.

What is the hyper information in this case? The easiest answer, just sum the contribution of every link (i.e. $\Phi \cdot \text{SCORE}(B_1) + \ldots + \Phi \cdot \text{SCORE}(B_n)$), is not feasible since we want the hyper information to be bounded.

This would seem in contradiction with the interpretation of a link as potential information that we have given earlier: if you have many links, you have all of their potential information. However, this paradox is only apparent: the user cannot get all the links at the same time, but has to *sequentially select* them. In other words, *nondeterminism has a cost*. So, in the best case the user will select the most informative link, and then the second more informative one, and so on. Suppose for example that the more informative link is B_1, the second one is B_2 and so on (i.e., we have $\text{SCORE}(B_1) \geq \text{SCORE}(B_2) \geq \ldots \geq \text{SCORE}(B_n)$). Thus, the hyper information is $\Phi \cdot \text{SCORE}(B_1)$ (the user selects the best link) plus $\Phi^2 \cdot \text{SCORE}(B_2)$ (the second time, the user selects the second best link) and so on, that is to say $\Phi \cdot \text{SCORE}(B_1) + \ldots + \Phi^n \cdot \text{SCORE}(B_n)$.

Observe that evaluating the score this way gives a bounded function, since for any number of links, the sum cannot be greater than $\frac{\Phi}{1-\Phi}$.

Also, note that we chose the best sequence of selections, since hyper information is the best "potential" information, so we have to assume the user does the best choices: we cannot use e.g. a random selection of the links, or even other functions like the average between the contributions of the each link, since we cannot impose that every link has to be relevant. For instance, if we did so, accessory links with zero score (e.g. think of the "powered with Netscape"-links) would de-value by far the hyper information even in presence of highly scored links, while those accessory links should simply be ignored (as the above method, consistently, does).

Now, we go on to refine the model against possible attacks.

8.1 More Security

Analogously to what seen for the unique-top approach, there is a problem due to the possibility of manipulating web objects within a same site.

A precaution that has to be taken is to distinguish between two fundamental types of links. Suppose to have a web object (*url*,*seq*). A link contained in *seq* is called *outer* if it has not the same domain of *url*, and *inner* in the other case. That is to say, inner links

of a web objects point to web objects in the same site (its "local world", so to say), while outer links point to web objects of other sites (the "outer world").

Now, inner links are from the sep point of view dangerous, since they are under the direct control of the site maintainer. For instance, a user that wants to artificially increase the hyper information of a web object A could set up on his site a very similar web object B (i.e. such that $\text{SCORE}(A) \approx \text{SCORE}(B)$), and put a link from A to B: this would increase the score of A by roughly $\Phi \cdot \text{SCORE}(A)$.

On the other hand outer links do not present this problem since they are out of direct control and manipulation (at least with very high chance).

Thus, when calculating the hyper function one should *ignore* the inner links. This also gives the advantage of making the implementation of the hyper information quite faster, since most of the links in web objects are inner.

Another important point concerns the same definition of link in a web object which is far from trivial. A link present in a web object is said to be *active* if the web objects it points to can be accessed by viewing (*url,seq*) with an HTML browser (e.g., Netscape Navigator or Microsoft Internet Explorer). This means, informally, that once we view P with the browser, we can activate the link by clicking over it. The previous definition is rather operational, but it is much more intuitive than a formal technical definition which can be given by tediously specifying all the possible casistics according to the HTML specification (note a problem complicating a formal analysis is that one cannot assume that *seq* is composed by legal HTML code, since browsers are error-tolerating).

Thus, the links mentioned in the paper should be only the active ones.

Yet another important issue is given by duplicate information (e.g. two links in a web object pointing to the same web object). In these cases, checks are necessary in order to avoid considering more then once the same information (for the details, see e.g. Marchiori, 1996b, 1996a).

Finally, observe that there are many different kinds of links, and each of them requires a specific treatment. For instance, local links (links pointing to some point in the same web object, using the #-command) should be readily ignored (this can be seen as an instance of the duplicated information issue seen before); frame links should be automatically expanded, i.e. if A has a frame link to B, then this link should be replaced with a proper expansion of B inside A (since a frame link is automatically activated, its pointed web object is just part of the original web object, and the user does not see any link at all); other links like source links of image tags, the background links, active links pointing to images, movies, sounds etc. should be ignored in a practical implementation of the hyper information (cf. Marchiori, 1996a), because they do not provide significant contributions (at least at the current technological level, cf. Sclaroff, 1995).

Although, as said, the hyper approach works well in presence of market pressure (and with its specification made public), we have tried to test it in the present situation, as post-processor of some major search engines, like Excite, HotBot, Lycos, WebCrawler and OpenText. The results of the evaluation has shown that in the average there has been a significant success against sep (our results show at least an 80–85% percentage of success in facing sep). One of the major issues in designing a security shield for search engines is that it should not worsen too much the behaviour of the main score function. So, being the hyper information a heavy modification of it, there may be some doubt about how this

affects the bounty of the original scores; to this respect, the hyper information behaves very well, since not only it usually does not worsen the score function, but it greatly improves it (indeed, as said before, the hyper information was initially developed for this aim, cf. Marchiori, 1996a).

REFERENCES

Bray, T. (1996). Measuring the Web. In *Fifth International World Wide Web Conference* Paris.

Churilla, R. (1996). Secrets of Searching the Web & Promoting your Website. Mentor Marketing Services.

Leonard, J. (1996). Search Engines: Where to find anything on the Net. C|net Inc.

Liu, J. (1996). Understanding WWW Search Tools. IUB Libraries, Indiana University.

Marchiori, M. (1996a). The Hyper Information: Theory and Practice. Tech. rep. 46, Dept. of Pure and Applied Mathematics, University of Padova.

Marchiori, M. (1996b). World Wide Web and Search Engine Persuasion. Tech. rep. 49, Dept. of Pure and Applied Mathematics, University of Padova.

Murphy, K. (1996). Cheaters Never Win. *Web Week*. 20 May.

Northern Webs (1996). The Search Engine Tutorial for Web Designers.

Rhodes, J. (1996). How to *Promote* Your Business Web Pages. Available at http://www.iinet.net.au/~heath/rhodes.html.

Sclaroff, S. (1995). World Wide Web Image Search Engines. In *NSF Workshop on Visual Information Management*. Cambridge, Massachusetts.

Sullivan, D. (1996). The Webmaster's Guide to Search Engines and Directories. Calafia Consulting.

Venditto, G. (1996). Search Engine Showdown. *Internet World*, 7(5).

Wingfield, N. (1996). Engine sells results, draws fire. C|net Inc.