A Two Player Game To Combat Web Spam

Michelle Goodstein* **Virginia Vassilevska**

June 1, 2007
CMU-CS-07-134

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a novel approach to combating web spam. In the spirit of Luis von Ahn's games with a purpose, we propose using a two player game to identify spam pages within search results. Our game asks users to classify a page as either highly relevant to a query or not relevant to a query, with the option of passing. We use data from the game as the input to a simple voting algorithm which determines whether a page is spam. We show that the best strategy for users playing the game for fun is to answer truthfully, and that spammers have difficulty obstructing the game.

* This research was sponsored by the Henry Luce Foundation through a Clare Booth Luce Graduate Fellowship and by National Science Foundation (NSF) grant no. CCR-0122581. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Henry Luce Foundation, the NSF or the US government.

Keywords: Web spam, Games with a purpose, Algorithms

1 Introduction

Web site owners can profit greatly if their pages achieve high rankings for popular search queries [7, 19]. Users tend to click pages that are ranked highly, and are unlikely to look far past the first page; instead, they will refine their query [19]. For this, and many other reasons, search engines try to provide the most relevant pages for queries at the top of the results page. Webmasters want to be ranked as highly as possible [19, 7]. **Web spam** occurs when webmasters manipulate their sites to take advantage of search engines' ranking algorithms. Their goal is to make their sites appear higher in results than their relevancy to a query merits. Web spam is abundant, even among the top query results: an example for query "nokia motorola" follows at the end of this section.[1]

Current methodologies for combating web spam results in an arms race: researchers race to create new algorithms to detect web spam, while **spammers** [webmasters engaging in deceptive practices to affect their pages' rank] work on ways to get by these techniques [7]. Little work has been done to provide theoretical performance guarantees for a web spam detection scheme. Our spam detection scheme uses a (hopefully) fun game to detect web spam, and uses the information gathered as votes for whether a page is spam or not. Even though a simpler scheme would be to directly collect data by letting users vote as they search, we will demonstrate that such a voting scheme is problematic. Our game will allow us to gain the benefits of voting without its problems.

This paper provides a brief overview of current research, explains some of the issues surrounding web spam detection, and illustrates one possible spam detection scheme with provable performance guarantees. In Section 2, a detailed overview of the spam detection problem is provided. Related work, and a comparison of terminology, occur in Section 3. An overview of our spam detection scheme, an interactive game, appears in Section 4. In Sections 5 and 6, we provide proofs that the game is strategy-proof for players desiring high scores and that we can detect and prevent all others from interfering with the scheme.

FREE RINGTONES - FREE RINGTONES
... FREE SANTO RINGTONES LOGOS NOKIA MOTOROLA FREE RINGTONES RING TONES FREE ... POLYPHONIC RINGTONES ON MY MOTOROLA T FREE SEAN PAUL RINGTONES FOR NOKIA ...

2 Overview of the problem

Most search engines (e.g. Google, MSN and Yahoo) already have good algorithms for ranking pages. However, these algorithms occasionally make mistakes, such as ranking a page higher than the average user would want it to be. Much research is devoted to improving page ranking and spam detection, with much of the focus on **link spam**. Link spam is a type of spam where a target page's rank is increased by creating many other pages that point to each other and to the target. Less time is spent on **content analysis**, where an algorithm evaluates the actual content of a page

[1] Sixth Google result on 2/6/07, from www.directory. lmc.edu/public_facilities_viewindividual-52.php

to determine whether it is spam or not. Most algorithms, even those that analyze page content, are designed to be used to create a ranking. Little work is done on how to modify a ranking that has already been generated. Our goal is, given an already prepared ranking of webpages, to identify pages that are likely spam and remove them from the ranking. Our approach involves collecting votes from a sample of individuals. These votes should tell us whether a particular page is spam or not with respect to a query. When a sufficient number of people have responded (e.g. 100) and enough have voted the page spam (e.g. 99% of the voters), we decide that the page must be spam and remove it from the ranking.

We begin with some definitions and notation.

Definition 2.1. *Let $\mathcal{Q}$ be a fixed query, and let N represent the number of search results for $\mathcal{Q}$. Define $R_{\mathcal{Q}}$ as the $N \times 1$ vector representing a ranking of result pages for $\mathcal{Q}$.*

Alternatively, $R_{\mathcal{Q}}$ is a permutation of the N pages produced by a search engine's ranking algorithm.

Definition 2.2. *Let $C_{\mathcal{Q}}$ be a nonnegative $N \times 2$ matrix such that $C_{\mathcal{Q}}[\mathcal{P}, 0]$ represents votes for page $\mathcal{P}$ as relevant with respect to query $\mathcal{Q}$, and $C_{\mathcal{Q}}[\mathcal{P}, 1]$ is the number of votes for $\mathcal{P}$ as irrelevant with respect to query $\mathcal{Q}$.*

Our goal is to create a new ranking $R'_{\mathcal{Q}}$ by removing some pages from $R_{\mathcal{Q}}$ using information from $C_{\mathcal{Q}}$. $R'_{\mathcal{Q}}$ has the property that with high probability, no page $\mathcal{P}$ in $R'_{\mathcal{Q}}$ is ranked highly but has a large number of web spam votes in $C_{\mathcal{Q}}$. In a more algorithmic sense, for each query $\mathcal{Q}$, our input is $C_{\mathcal{Q}}$ and $R_{\mathcal{Q}}$ and our output is $R'_{\mathcal{Q}}$, where we impose a post-condition on $R'_{\mathcal{Q}}$. To clarify this post-condition, we will review and introduce some terminology.

Definition 2.3. *A **trusted agent** is a person who tries to truthfully classify page-query pairs and passes when unsure.*

This is analogous to a truth-revealing strategy in mechanism design [16]. Given a set of possible outcomes, a truth-revealing strategy for a player is to report true preferences for each outcome. We can interpret preferences as classifications of page-query pairs. We will use the terms *t*ruth-revealing and *h*onest interchangeably in this paper. By definition, a trusted agent cannot have a stake in any page-query pair; for example, a trusted agent cannot be a spammer. Other adversarial criteria are set out in Section 6.

Definition 2.4. *We will consider agreement among at least 90% of users to be an **overwhelming majority** of users.*

Remark 1. *A page is **ranked highly** if it is among the top 10% of the results for a query.*

Definition 2.5. *A web page $\mathcal{P}$ is considered to be **web spam** if it is ranked highly in $R_{\mathcal{Q}}$ but an overwhelming majority of trusted agents believe it is not relevant to $\mathcal{Q}$.*

Definition 2.6. *Let $\hat{R}_{\mathcal{Q}}$ be the ranking generated by deleting from $R_{\mathcal{Q}}$ those pages that are voted irrelevant by an overwhelming majority in $C_{\mathcal{Q}}$, where only feedback from trusted agents is used in the formation of $C_{\mathcal{Q}}$.*

We now present a definition for $R'_{\mathbb{Q}}$ and refine our post-condition for our algorithm.

Definition 2.7. *Let $R'_{\mathbb{Q}}$ be the ranking generated by deleting from $R_{\mathbb{Q}}$ those pages that are voted irrelevant by an overwhelming majority in $C_{\mathbb{Q}}$.*

We will show in Section 5 that the strategy with the most gain is to provide our game with honest answers, which leads us to expect that R' and $\hat{R}$ will be similar, since non-adversarial players can be expected to play truthfully.

3 Related Work

To motivate our definition, we will compare several other definitions from the literature. As this is still an emerging field of study, there are several definitions currently in use for web spam, with no single established version.

Definitions from the literature

- Gyongyi and Garcia Molina [10] define web spam as "deliberate human action...meant to trigger an unjustifiably favorable relevance...for some web page". In another work, Gyongyi, Garcia-Molina and Pedersen [12] define web spam as "hyperlinked pages on the World Wide Web that are created with the intention of misleading search engines." The authors present an algorithm named TrustRank. Similar in methodology to PageRank [15], a commonly used ranking algorithm in web search that propagates the popularity of a page using the link graph, TrustRank works by propagating trust through the web graph by following links. The only component of human feedback is the initial *s*eed set evaluation; after that, their algorithm is automated. In their technical report on link spam alliances, Gyongyi and Garcia-Molina [9] focus on web spam as the result of creating content "with the main purpose of misleading search engines and obtaining higher-than-deserved ranking in search results." Gyongyi et al. [11] use a similar definition when defining *s*pam mass, a metric of how much the PageRank of a page $\mathcal{P}$ is affected by inlinks from spam pages.
- Da Costa Carvalho et al. [6] describe spam pages as those pages which benefit from inlinks made intentionally in order to artificially inflate the importance of pages. This restricts the definition of spam to link spam. Their paper focuses on the link graph model of the web, but at the (internet) domain name level instead of at the page level. It attempts to detect suspicious links so that PageRank [15] can ignore these links.
- Fetterly, Manasse and Najork [7] designate web spam as "pages that exist only to mislead search engines into (mis)leading users to certain web sites....the SEO[search engine optimization]-generated pages are intended only for the search engine, and are completely useless to human visitors." Statistical ways of analyzing URLs, host names, the web graph and individual page content are examined here, each as a different automated technique for detecting web spam. Ntoulas, Najork, Manasse and Fetterley [14] define web spam as "the injection of artificially-created pages into the web in order to influence the results from search engines, to drive traffic to certain pages for fun or profit". They focus on different automated methods of web spam detection based solely on page content.
- Wu, Goel and Davidson [18] define web spam as "behavior that attempts to deceive search engine ranking algorithms". They further specify that the behavior involves "manipulating web

page features on which search engines' ranking algorithms are based". This behavior results in spam pages. Their algorithm is a modified version of TrustRank. Krishnan and Raj [13] propose a link-based system that propagates distrust.

Some of these papers define web spam in terms of actions by a webmaster, and spam pages as the result of those actions; other definitions merely describe web spam as the results of webmasters' behavior. The distinction is minute; it is sufficient to restrict ourselves to one or the other. Each of these papers, then, relies at some point on measurable criteria, since the original behavior and intent of the webmaster is not available. However, their criteria cannot match their original definition, meaning that they cannot measure what they defined to be spam. Our definition differs by immediately offering measurable criteria as it relates to user feedback.

In all cases, the algorithms previously proposed are automated. If human feedback is solicited at all, it is in the early seed set stage [12, 18, 3, 4]. Also, without provable performance guarantees, nothing prevents spammers from adapting. Spammers' adaptations require algorithm designers to change their algorithms, leading to the "arms race"As an example, Topical Trustrank is an attempt to improve TrustRank and make it more spam-resistant.

Our work differs from previous work in several ways. First, we solicit user feedback at several points. By collecting data from people, we can use statistical sampling theory to derive whether most people think a page is likely spam. Our algorithm thus suceeds in detecting web spam, as defined above in terms of public opinion. We can use statistical sampling theory to achieve a good approximation of global opinion of a page's relevance to a query.

Secondly, just because an automated algorithm decides a page is relevant to a query does not mean a human will agree; there may be factors visible to the human eye that a computer cannot easily pick up on that allows a human to (correctly) classify a page as spam where a program cannot. The only adaptation we offer an adversary is to make a page's content more relevant to a query.

4 A web spam detection scheme

Motivated by the work of [17] in using two player games to solve interesting AI problems, we investigate a similar approach for web spam. Von Ahn and Dabbish created the ESP Game, a fun two player game that induces users to label images. They not only help researchers working on computer vision, currently a hard AI problem, but also improve image search. Our aim is to also employ a fun two player game. We will collect votes from the population on whether a webpage is spam with respect to a query. Our ultimate goal is to use a simple voting algorithm to decide whether to move a page down in the rankings. Note that our scheme can be deployed together with any of the previously mentioned automated web spam detection techniques.

There are easier ways to collect votes. For example, we could let users classify search results as highly relevant or not highly relevant while searching. This approach has some major drawbacks. It is relatively easy for spammers to keep a page in the ranking by repeating the search with the same query and voting it relevant. The largest vulnerability is that the user is in charge of picking the query and the page, and then gets to classify the page-query pair. This allows spammers to vote their pages as highly relevant, and for adversaries to try to vote legitimate pages as irrelevant. We cannot produce any meaningful performance guarantees in such a scheme, because the adversaries are simply too powerful. By using our game to generate the votes, we show how to avoid such vulnerabilities and achieve provable performance guarantees.

In the following sections we introduce our game, show how to process the votes obtained from it, prove that the game is is strategy-proof, and that under some assumptions it is immune to attacks.

4.1 A web spam game

Before we present the actual web spam game we modify the original problem statement from Section 2 to fit the game setting.

Recall the vote matrix $C_{\mathcal{Q}}$ from Section 2. $C_{\mathcal{Q}}$ represents cumulative feedback about a given query, without specifying how this feedback is gathered. There are two columns, and N rows. As before, N is the number of pages in $R_{\mathcal{Q}}$. $C_{\mathcal{Q}}[\mathcal{P}][0]$ contains the number of users who voted page $\mathcal{P}$ as highly relevant to query $\mathcal{Q}$; similarly, $C_{\mathcal{Q}}[\mathcal{P}][1]$ stores the votes for $\mathcal{P}$ as not highly relevant to $\mathcal{Q}$.

We assume that the page ranking $R_{\mathcal{Q}}$ obtained from an external source is approximately optimal. The ranking $R'_{\mathcal{Q}}$ we want to obtain from our spam detection game is supposed to be a "better" version of $R_{\mathcal{Q}}$. However, while there may be errors scattered throughout the entire original ranking, we only care about results that the user will actually see (say, the first 10, 20, 30 or 40) per query. Therefore, $R'_{\mathcal{Q}}$ only needs to be more accurate for highly ranked items.

The goal is to construct the new ranking $R'_{\mathcal{Q}}$ by removing items from $R_{\mathcal{Q}}$ that are ranked highly but most people rate as spam. If we could trust user feedback, then we could simply allow users to choose $\mathcal{Q}$ and vote on pages $\mathcal{P}$ in $R_{\mathcal{Q}}$, filling in $C_{\mathcal{Q}}$. Once we relax the assumption that users are trusted, this methodology becomes open to attacks as described previously in Section 4.

The attack mentioned in the beginning of the section relies on the ability of an untrusted user to choose the page-query pair that they want to affect. To avoid this, we would like to control $\mathcal{Q}$ and $\mathcal{P}$ so that an adversary cannot greatly affect the vote count for any particular page-query pair $(\mathcal{P}, \mathcal{Q})$. In the two player game, we randomly select a query $\mathcal{Q}$, and page $\mathcal{P}$ indexed at a random point in $R_{\mathcal{Q}}$ unknown to the user. Let a snippet s_p be defined as a small, representative piece of text from $\mathcal{P}$. [For an example of a query-snippet pair, see Figure 1]. We present two users with s_p and then ask them: is $\mathcal{P}$ highly relevant to $\mathcal{Q}$ or not? If the two users match on their answer, we can use that match as a vote, where a vote for high relevance corresponds to a vote for not web spam, and a vote for not highly relevant corresponds to a vote for web spam. A key assumption for the game is that s_p represents the page well and that the algorithm for the creation of s_p is hidden from the users. Suppose it was the same snippet used by search engines. Then there is an immediate attack for any adversary. The adversary searches for the query whose result page they wish to affect, and finds their page in the ranking; then, the adversary examines the snippet from his page that is shown in the search results, finds the relevant text in his actual page, and tweaks his page. He repeats this procedure until the snippet from his page shown in the search results looks relevant to the query, even if the page does not. This immediately leads to the following requirement: a snippet that a search engine employs cannot be used for a page-query pair, as this exposes elements of the algorithm to a potential spammer [see Section 6 for further details].

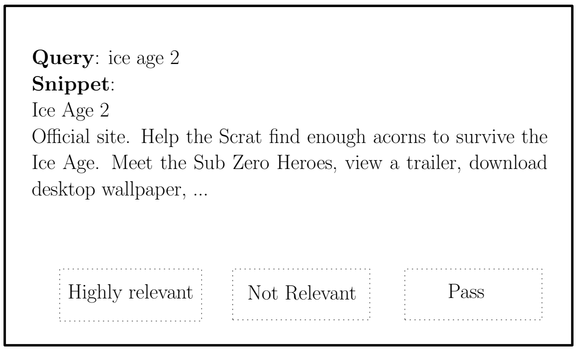

Figure 1: A potential highly relevant (test) question, with query ice age 2 and snippet text from the first Google result for "ice age 2" on 11/10/06

4.2 Game description

We design the game as a series of s independent questions. For each question, the pair of users is alloted at most t units of time, and T units of time to complete the entire game. User pairings are assigned at random at the beginning of the game, and change with each iteration. Each question to a pair consists of a query $\mathcal{Q}$, a short snippet s_p from a randomly drawn page $\mathcal{P}$ in $R_{\mathcal{Q}}$, and three options: "Highly relevant", "Not highly relevant", and "Pass". Players only get one attempt to answer the question; once they choose an option, they cannot change their minds. Users cannot see what rank $\mathcal{P}$ has in $R_{\mathcal{Q}}$ currently, or the web page URL. A visualization of the game appears in Figure 1.

We assign m points for a match, and subtract n points for a mismatch; 0 points are allotted for a pass. Without loss of generality, set $m = 1$ and $n = 1 + \varepsilon$. We shall see later that $\varepsilon > 0$ is a necessary condition for a game where the dominant strategy is to play honestly. In each round of the game, we incorporate K **test questions** (divided among relevant and non-relevant page-query pairs) for which we know the correct answer. These test questions are designed to be indistinguishable from the other questions. Any time an individual user answers some of their test questions correctly and none incorrectly, they are awarded a bonus $\mathcal{B}$ according to the number of test questions answered correctly, but are not told which K of the questions were test questions. The outcome from an example game is given in Table 1. Player 1 disagrees with a test question (question 2) and so her score is the sum of the match points, here, $1 - \varepsilon$. Player 2 never disagrees with a test question and answers at least one test question, meaning her score is $1-\varepsilon+\mathcal{B}(1)$, where $\mathcal{B}(1)$ is the value a user gets for answering exactly one test question correctly and none incorrectly.

4.3 Generating test questions

Many of the results we achieve depend on the availability of many test questions distributed roughly equally among highly relevant and not highly relevant pages. Generating not highly relevant test questions can be done rather simply. The method used to generate the query-snippet pair in Figure 2 involves permuting the words in a query $\mathcal{Q}$ together with 3 random words to form $\mathcal{Q}'$. To create the test question, we present a result from the search for $\mathcal{Q}'$, together with $\mathcal{Q}$. Experimentally,

	Q1	Q2	Q3	Q4	Q5
Test(X/R/N)	X	R	X	X	NR
Player 1	R	NR	NR	P	NR
Player 2	R	R	NR	NR	P
Match Points	$+1$	$-(1+\varepsilon)$	$+1$	0	0

Table 1: A simple scoring example. There are five questions. If a question is a test question, it has its solution (either R for relevant, or NR for not relevant) in the appropriate column; otherwise, it has an X. Player 1 and Player 2 answer either R for relevant, NR for not relevant, or P for pass.

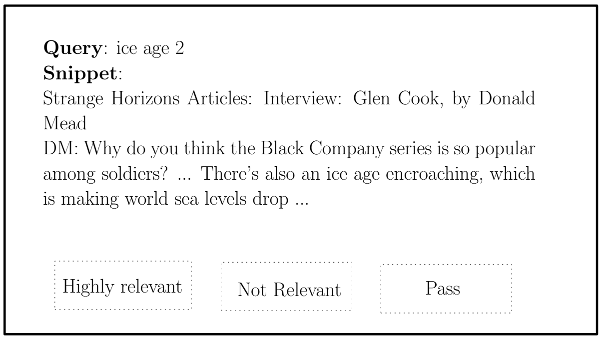

Figure 2: A potential not relevant test question, with presented query "ice age 2" and real query "company cook 2 over age ice". Snippet text from the first Google result for "company cook 2 over age ice" on 11/10/06

our implementation of this procedure created results that often contain $\mathcal{Q}$, but are also not highly relevant to $\mathcal{Q}$, as long as the original query was not a celebrity's name. We pulled the random words from a list of basic English words [1]. Figures 2 contains an example of a possible test question created with this method. The queries used were taken from the Google Zeitgeist archive [8], a collection of popular queries and trends made available by Google. We used the Google API to retrieve the snippets [2]. The snippet shown is a combination of the link text and the snippet from a standard Google search, on the first result returned. Figure 2 shows queries where we can generate test questions that are not relevant–just present the original query $\mathcal{Q}$ with the snippet for the query $\mathcal{Q}'$. In the case where this method fails to produce a clear answer, the user can still pass and get a bonus.

In addition, once we have collected a sufficient quantity of votes, we can also use questions that have a large majority (say 60-70%) of the population voting one way or another as test questions. Since we anticipate that many results taken from the top of a query $\mathcal{Q}$'s result page will be highly relevant, this should allow us to quickly amass many highly relevant test questions.

4.4 Processing the votes

The spam detection game we described provides us with relatively trustworthy votes for various page-query pairs. Here we present a simple algorithm to process these votes and decide for each given page-query pair whether the page is spam with respect to the query.

The algorithm takes as input a matrix of votes, and removes those pages that have more than 100 votes and where 100 times as many people vote a page spam as not spam. While requiring more than 100 votes is an arbitrary threshold, the criterion of 100 times as many people voting a page spam as not is actually rooted in our definitions. Recall that we only wish to eliminate those pages which an overwhelming majority ($90 - 99\%$ of the population) deemed irrelevant. Requiring 100 times as many people to vote a page irrelevant achieves an overwhelming majority. Since we are only concerned with the top few rankings $(10, 20, 30, 40)$, we can optimize to only return a subranking of the top k items. We start at a beginning index value, and are given a value k. The loop continues until k pages have made it through the loop without being removed from R'. To make sure that each page is considered at most once, we use the variable *index*, which refers to which page we are examining in the original ranking $R_{\mathcal{Q}}$ for query $\mathcal{Q}$. This procedure is presented in Algorithm 1.

The algorithm has some very attractive features. Firstly, its simplicity makes it easy to understand. Furthermore, it is very conservative; if fifty percent of the population feel one way and fifty percent another, we will leave the page in, erring on the side of keeping spam rather than eliminating legitimate content. One potential problem is our reliance on receiving many trustworthy votes for each page-query pair, which implies a need to collect a lot of trustworthy data from the game. In the next few sections we show that the data will be trustworthy. The quantity of data we collect will largely depend on how much fun the game is. We motivate our belief that our game is fun in the next section.

4.5 The fun factor

For our scheme to succeed, people must want to play the game, which means the game must be fun. There are many reasons to believe our game will be fun in practice. First, we plan to use queries from Google Zeitgeist [8], which is a source of popular and surprising queries. By playing the game, players will be exposed to queries that other people think are interesting and fun, and thus we expect the players to enjoy them as well. Secondly, by playing the game, people will know that they are helping search engines determine what is and is not spam. In addition, there is some amusement in being paired with a stranger and trying to decide how they will view a query-snippet pair.

In the cases where the queries seem relevant to the snippet, players will also have an opportunity to learn trivia, and many trivia games are considered popular. Irrelevant snippets, though, can also be amusing depending on what ways they differ from the query; see Figure 2 for an example.

5 The game is strategy-proof

In this section we analyze the game and show that the players whose goal is to maximize the number of points, when playing optimally, provide us with their honest opinions.

We borrow terminology from game theory and mechanism design in analyzing our spam detection scheme. For further definitions, see Parkes' work [16].

Algorithm 1 Variable query, returning top k elements

Require: $start\_index \geq 0$ and $k \geq 0$
Ensure: $end\_index \geq k + start\_index$
  $index = start\_index$
  On query $\mathcal{Q}$
  $R_{\mathcal{Q}}$ = the initial ranking associated with query $\mathcal{Q}$
  $C_{\mathcal{Q}}$ = the vote count vector associated with query $\mathcal{Q}$
  Initialize R' as an array of size k {Assume zero-based array}
  $numPassed = 0$
  while $numPassed < k$ **do**
    $count\_pos = C_{\mathcal{Q}}[index + numPassed][0]$
    $count\_neg = C_{\mathcal{Q}}[index + numPassed][1]$
    $p = R_{\mathcal{Q}}[index + numPassed]$
    if $(count\_neg > 100(count\_pos + 1))$ **then**
      remove $\mathcal{P}$ from R' {Algorithm rates $\mathcal{P}$ as spam}
    else
      $R'[numPassed] = \mathcal{P}$ {Algorithm rates $\mathcal{P}$ as not spam, so it is placed in R'}
      $numPassed + +$
    end if
    $index + +$
  end while
  $end\_index = index$ {In case we need another page for this query, track location relative to R's index}
  return R' and end_index

Definition 5.1. *A **strategy** represents the plan a user has for making choices in any possible situation within the game.*

Definition 5.2. *A **dominant strategy** is a strategy that maximizes the utility (here, points awarded to a user) when the strategies of other players within the game are unknown.*

Definition 5.3. *A game is **strategy-proof** if the dominant strategy is to play honestly.*

We would like to state that our game is strategy-proof, and show the necessary conditions for the parameters m, n, ε and $\mathcal{B}$ from Section 4.2.

5.1 Simple example

We begin with a small motivating example. Suppose that two users are paired. Player 2 knows all the answers, and plays honestly. Player 1 votes "relevant" with probability = $1/2$ and "not relevant" with probability = $1/2$. This implies Player 1 never passes. In this scenario, we can restrict our attention to Player 1's actions as they uniquely determine the score. For this example, we shall ignore the bonus $\mathcal{B}$.

Lemma 1. *Assume Player 1 believes Player 2 is honest and omniscient. Guessing uniformly at random is never a dominant strategy when $\varepsilon > 0$.*

Proof. Suppose $m = 1$ and $n = 1 + \varepsilon$. E[pass] $= 0$ but E[random guess] $= 1/2(s) - 1/2(s)(1 + \varepsilon) = -s\varepsilon/2 < 0 \Leftrightarrow \varepsilon > 0$. Since E[pass] $> E$[random guess], guessing uniformly at random is never a dominant strategy. □

Definition 5.4. *We define **confidence** as the probability a user thinks their answer will match an omniscient, honest partner.*

Remark 2. *Throughout this paper, we will assume that the probability a user assigns to matching his or her partner and the true probability of a match are approximately equal, and will use these definitions interchangeably.*

The dominant strategy for a user should be to play honestly when they feel confident in their answer, and to pass when they do not. To ensure this, we show the correct bounds for parameters $\mathcal{B}$ and ε.

5.2 Playing honestly, or passing, is a dominant strategy

The goal of this section is to prove our main theorem, that playing honestly when a player knows the answer, and passing otherwise, is the dominant strategy.

5.2.1 The game with no bonus when the opponent is honest

We assume Player 2 is omniscient and honest and again ignore the bonus $\mathcal{B}$. We restrict our analysis to Player 1's actions, since these determine the score for both players.

Lemma 2. *When Player 2 is honest and omniscient, the dominant strategy of Player 1 is to answer honestly when the probability that he knows the answer is $\geq \frac{(1+\varepsilon)}{(2+\varepsilon)}$, and to pass otherwise.*

Proof. Let p be the confidence Player 1 has in his answer. Now, since the opponent is playing honestly, the expected score on any question is $p(1) - (1 - p)(1 + \varepsilon)$. We want to discover when this strategy is better than simply passing (which has expected value of 0). By linearity of expectation, we can restrict our analysis to the expectation of points for one question:

$$\begin{aligned} p(1) - (1-p)(1+\varepsilon) &> 0 \\ \Leftrightarrow p(2+\varepsilon) - (1+\varepsilon) &> 0 \\ \Leftrightarrow p &> \frac{(1+\varepsilon)}{(2+\varepsilon)}, \quad \forall \varepsilon > 0 \end{aligned}$$

□

Remark 3. *This indicates that Player 1 should answer the questions when* $p > \frac{(1+\varepsilon)}{(2+\varepsilon)}$*, and pass otherwise for a positive expected utility. Note that once a choice of* ε *is made there is a concrete threshold for when Player 1 should answer questions. By choosing* ε*, we can alter the threshold* p *for when Player 1 should answer or pass.*

Definition 5.5. *Define* $p_c = \frac{(1+\varepsilon)}{(2+\varepsilon)}$ *as the* ***threshold confidence****.*

Claim 1. p_c *is the minimum probability such that for all* $p > p_c$*, the expectation of points obtained for answering a single question honestly is positive (ignoring the bonus).*

Remark 4. *Note since* $\varepsilon > 0$ *by Lemma 1,* p_c *is always greater than* $1/2$*.*

When $p > p_c$ this yields a positive expectation for the entire game. Any other strategy performs no better, as it involves more passing (which lowers the expectation), or requires guessing on questions where $p \leq p_c$. Since p_c is the minimum threshold yielding a positive expectation, any strategy involving answering with confidence $\leq p_c$ will have non-positive expectation.

5.2.2 The game with a bonus

We now consider the game with a bonus, redefining $\mathcal{B}$ as a function of the number of test questions answered correctly if none are answered incorrectly.

Lemma 3. *Let* p_c *be the confidence threshold. Suppose an honest user answers* k *test questions. Let* $\mathcal{B}(k)$ *be the bonus for answering* k *bonus questions correctly and none incorrectly. Then for answering honestly when* $p > p_c$ *and passing otherwise to be a dominant strategy, it must be the case that* $\forall k' \geq k$*,* $\frac{\mathcal{B}(k)}{\mathcal{B}(k')} \geq (1/2)^{k'-k}$*.*

Proof. Assume the contrary. Fix a player, and consider those k of the test questions that a player knows with confidence $p > p_c$. Suppose, on the remaining questions, the player guessed. Then his expectation is $p^k(1/2)^{k'-k}\mathcal{B}(k')$. On the other hand, the expectation of just answering the k test questions, and passing on the others, is $p^k\mathcal{B}(k)$. If $\frac{\mathcal{B}(k)}{\mathcal{B}(k')} < (1/2)^{k'-k}$ then $p^k\mathcal{B}(k) < p^k(1/2)^{k'-k}\mathcal{B}(k')$, which implies honesty is not a dominant strategy. Contradiction. $\square$

Lemma 3 shows the necessary conditions for our game to be strategy-proof. We now present a sufficient condition.

Lemma 4. *If* $\mathcal{B}(i) = \beta/p_c^i \; \forall i \in \mathcal{Z}^+$ *for some fixed* $\beta > 0$*, then Lemma 3 is satisfied.*

Proof. Recall that $p_c > 1/2$ by definition. If we can show $p_c^k\mathcal{B}(k) \geq p_c^k\mathcal{B}(k')(1/2)^{k'-k}$, then we will have shown that guessing randomly on an extra $k' - k$ questions will not give an advantage over an honest player who passes on those questions. We know $\mathcal{B}(i) = \beta/p_c^i$, $k' \geq k$, and $1/2 < p_c \Rightarrow 1 < 2p_c \Rightarrow 1 > 1/(2p_c)$. Then,

$$\begin{aligned}
1 &\geq (1/(2p_c))^{k'-k} \\
(1/(2p_c))^{k'-k} &= \frac{p_c^k(1/2)^{k'-k}}{p_c^{k'}} \\
\frac{p_c^k\beta}{p_c^k} &\geq \frac{p_c^k\beta(1/2)^{k'-k}}{p_c^{k'}} \\
p_c^k\mathcal{B}(k) &\geq p_c^k\mathcal{B}(k')(1/2)^{k'-k} \\
\frac{\mathcal{B}(k)}{\mathcal{B}(k')} &\geq (1/2)^{k'-k} \quad \forall k' \geq k.
\end{aligned}$$

□

Remark 5. *$B(i) = \beta/p_c^i$ implies that a player never increases his expectation of a bonus by guessing randomly on questions where the honest player would pass.*

Note that W.L.O.G. $p \geq 1/2$, since otherwise the confidence in the other answer is $\geq 1/2$. We can now prove our main theorem.

Theorem 5.1. *Let $\mathcal{B}(i) = \beta/p_c^i$ be the bonus a player receives for answering $i > 0$ test questions, all correctly. Then playing honestly when $p \geq p_c$ and passing otherwise is a dominant strategy*

Proof. Fix any game. Let k represent the number of test questions the honest player would answer. Any dishonest strategy must employ some linear combination of three strategies over the game: answering questions the honest player passed on, changing the answer of the honest player, and playing some questions honestly. By linearity of expectation, we can consider the points earned due to the bonus and points from matching separately.

In the following lemma, we demonstrate that the expectation of a bonus employing any combination of these strategies must be less than that in the honest case.

Lemma 5. *Suppose the honest player answers k test questions honestly, and passes on the rest. Suppose the dishonest players answers m test questions honestly, changes the answer on n, and answers l that the honest player skipped, $m, n, l \geq 0$ and $m + n \leq k$. Then honesty maximizes the expected number of points due to the bonus.*

Proof. Notice that other than $m+n \leq k$, we do not restrict whether $m+n+l \geq k$ or $k \geq m+n+l$ – this can vary, and allows the dishonest player full flexibility. Furthermore, answering l questions that the honest player skipped could also mean flipping the answer of the honest player. To analyze the best case for the adversary, let $\tilde{p}$ be the minimum confidence the dishonest player has, and $\hat{p}$ the maximum confidence the dishonest player has over all questions he answers. Divide the l questions the dishonest player answered that the honest player skipped into two categories, such that the dishonest player answers h honestly with maximum confidence $\hat{p}$ and j dishonestly with probability $1 - \tilde{p}$. By definition, $h + j = l$ and $h, j \geq 0$. Clearly, $p_c > \hat{p} \geq \tilde{p} \geq 1/2$, as well as $p_c > 1/2 \geq 1 - \tilde{p} \geq 1 - \hat{p}$. Also, $p_c > \hat{p} \geq 1/2$, since the honest player did not answer the question.

We are given the following set of inequalities.

$$
\begin{aligned}
m, n, j, k, l, h &\geq 0 && (1)\\
j + h &= l && (2)\\
k &\geq m + n && (3)\\
k &\geq m && (4)\\
p &\geq p_c > 1/2 && (5)\\
p_c &> \hat{p} \geq \tilde{p} \geq 1/2 && (6)\\
p_c > 1/2 &> 1 - \tilde{p} \geq 1 - \hat{p} && (7)\\
p_c &\geq 1 - p && (8)
\end{aligned}
$$

From line 5 and the fact that $k - m \geq 0$ we get

$$
\begin{aligned}
p/p_c &\geq 1 && (9)\\
(p/p_c)^{k-m} &\geq 1 && (10)
\end{aligned}
$$

Combining lines 5, 6, 7, and using the result from above, we get:

$$
\begin{aligned}
(p_c)^{n+j+h} &\geq (1-p)^n \hat{p}^h (1-\tilde{p})^j\\
(p/p_c)^{k-m} (p_c)^{n+l} &\geq (1-p)^n \hat{p}^h (1-\tilde{p})^j
\end{aligned}
$$

We rearrange terms and eventually multiply by β to get the final result.

$$
\begin{aligned}
\frac{p^k p_c^{m+n+l}}{p^m p_c^k} &\geq (1-p)^n \hat{p}^h (1-\tilde{p})^j\\
\frac{p^k}{(p_c)^k} &\geq \frac{p^m (1-p)^n \hat{p}^h (1-\tilde{p})^j}{p_c^{m+n+l}}\\
\frac{p^k \beta}{(p_c)^k} &\geq \frac{p^m (1-p)^n \hat{p}^h (1-\tilde{p})^j \beta}{p_c^{m+n+l}}\\
p^k B(k) &\geq p^m (1-p)^n \hat{p}^l (1-\tilde{p})^j \mathcal{B}(m+n+l)
\end{aligned}
$$

Finally, E[bonus for honest players] $\geq E$[bonus for dishonest players under any strategy]. □

The dominant strategy to maximize points from the bonus is to play honestly. The dominant strategy over the entire game is the strategy that maximizes the sum of points from the bonus and points from matching. We can adjust β so that the sum is dominated by the bonus points, thus ensuring that the game is strategy-proof. □

We now show that, assuming users play honestly, the probability of a match between two randomly paired players exceeds the probability of a random vote.

Lemma 6. *The probability of two rational users matching and generating a vote within C_Q exceeds the probability of a random vote.*

Proof. Let p denote the probability Player 1 is right (Player 1's confidence) and q denote the probability that Player 2 is right (Player 2's confidence). Assuming that both players are rational and choose to answer the question, the probability that they match is $pq + (1 - p)(1 - q)$, which accounts for the fact that either they are both right on the same answer, or they are both wrong on the same answer. We know that $p, q > 1/2$ since $p > p_c > 1/2$; the same argument holds for q. Without loss of generality, suppose $q \geq p$. We can rewrite this as $q = p + \delta$ for some $\delta \geq 0$.

$$\begin{aligned} pq + (1-p)(1-q) &= p(p+\delta) + (1-p)(1-p-\delta) \\ &= (2p^2 - 2p + 1) + \delta(2p - 1) \end{aligned}$$

We can now break this into pieces: $2p^2 - 2p + 1$ has a minimum at precisely $p = 1/2$. Also, $\forall p > 1/2$, $2p^2 - 2p + 1 > 1/2$. Likewise, for $p > 1/2,\ 2p - 1 > 0 \Rightarrow \delta(2p-1) > 0$ when $\delta > 0$. Putting this together, we get that $(2p^2 - 2p + 1) + \delta(2p - 1) > 1/2 + 0 = 1/2$.

□

We have shown that a match between two honest players within our game is likelier than a random vote. Theorem 5.1 allows us to assume that rational players behave honestly. Therefore, our game produces data that is more likely than votes cast uniformly at random.

6 Adversaries

We now now show that adversaries encounter difficulty affecting search rankings through our game. We will define three adversaries to make the analysis clearer. Let Sam be a spammer who wishes to move a non-relevant, spam page $\mathcal{P}$ up in the rankings for a query $\mathcal{Q}$. As a subproblem, Sam must first maintain $\mathcal{P}$'s location within the rankings for query $\mathcal{Q}$. Let Mallory be an adversary who wants to move a relevant, non-spam page $\mathcal{P}'$ down in the rankings for a query $\mathcal{Q}'$. Finally, we consider Gene to be a generic attacker, who is not interested in any specific page, but wishes to corrupt all rankings.

6.1 Sam

To boost the ranking of his page, Sam must persuade users (either bots or humans) that agree with him to play the game multiple times. In doing so, Sam (and his agents) must wait for a query to come up that is relevant to his page $\mathcal{P}$. Also, Sam must decide whether the best strategy is for him to vote the page up or down. We assume that the time restriction of t units per question prevents Sam from searching for the query, and finding what page and ranking value the snippet is associated with. Sam can check the page to see if it is his own snippet, and vote it up. However, this requires several other things to occur in tandem:

The first problem is *partner agreement*–if his page is truly spam, Sam must hope that the partner he is playing against is either his agent or an honest player who thinks $\mathcal{P}$ is relevant. Otherwise, the vote would not count and he would not affect our algorithm.

Lemma 7. *Let $\mathcal{P}$ be the page Sam wants to raise in the rankings, and let $0 \le p_s \ll 1/2$ be the fraction of the honest population who believe that $\mathcal{P}$ is not spam. Let p_m be the probability that $\mathcal{P}$ emerges within one game. Then the expected number of games Sam must play to accumulate one vote in the algorithm is $\frac{1}{p_s p_m}$, assuming that Sam does not have enough agents to affect p_s.*

Proof. This follows by linearity of expectation. The number of people who believe a page is spam or not is independent of whether a snippet of that page occurs within a game, and thus the probability of Sam both encountering someone who he agrees with and encountering a question he cares about is $p_s p_m$, so he must expect to play $\frac{1}{p_s p_m}$ games to get one match. □

Consider what would happen if Sam decided to employ agents to help achieve matches. As we are matching players at random, Sam needs many agents. Let H be the number of honest users in the game. Even if we assume a uniform distribution for how players are matched, for Sam to even have a $1/2$ chance of matching his own agent, he must introduce $(1/2 - p_s)H$ agents. Since H is a hidden parameter, as long as H is sufficiently large or p_s is sufficiently low, it is difficult for Sam to add enough agents to affect any page's score. Also, Sam is never aware of having enough agents within the game because H is a hidden parameter. Assuming p_s is low is reasonable; otherwise the page would not be identified as spam under our scheme. We will say that Sam has an *insufficient number* of agents to affect the ranking.

Another problem Sam faces is that while Sam's strategy is effective if a page-query pair comes up often enough, Sam has to play a large number of times to get the same pair.

Lemma 8. *Let $n_s = \frac{1}{p_s p_m}$ be the expected number of games Sam must play to achieve one match on $\mathcal{P}$, and $100 n_{\mathcal{P}}$ be the number of current votes for page $\mathcal{P}$ as spam, with $n_{\mathcal{P}} \in \mathcal{R}^+$. Let $m_{\mathcal{P}}$ denote the number of users who think that $\mathcal{P}$ is not spam without Sam's votes. Then Sam must expect to need to play $n_s(n_{\mathcal{P}} - m_{\mathcal{P}})$ times to affect the ranking.*

Proof. Without loss of generality, assume $n_{\mathcal{P}} \ge 1$, otherwise there are not enough votes to throw out $\mathcal{P}$ anyway. As a preliminary attempt, assuming no one other than Sam believes his page is not spam, he must expect to play $(n_{\mathcal{P}})n_s$ games by linearity of expectation. Now, remove the assumption that no one other than Sam believes his page is not spam. Let G be a random variable representing the total number of games Sam must play to keep his page within the rankings. For his page not to be removed, he needs:

$$100np < 100(m_{\mathcal{P}} + G) \Rightarrow n_{\mathcal{P}} < m_{\mathcal{P}} + G \Rightarrow G > n_{\mathcal{P}} - m_{\mathcal{P}}$$

By linearity of expectation, since it takes n_s games for Sam to expect one match, it takes $n_s(G) > n_s(n_{\mathcal{P}} - m_{\mathcal{P}})$ games to amass enough votes to keep his page in the ranking. □

It is reasonable to expect that as long as we have many page-query pairs, n_s will be a large number and thus Sam will have a hard time exceeding it. Also, if $n_{\mathcal{P}}$ is large, which might occur in the case that many people have strong opinions, this can only make Sam's work harder.

The final problem Sam encounters is the difficulty of recognizing and comparing the snippet. We can embed either the snippet, query, or both, within an image (possibly using a CAPTCHA) to make such comparisons difficult for computers. Then, the attacker needs to know the snippet. If the game uses a separate snippet from a conventional search engine, it can be difficult to discover what the snippet associated with the game is. Sam's job is very difficult: he has to play $\frac{1}{p_m}$ times just to encounter a snippet related to a page he is interested in, and even if that snippet is encountered, recognizing it as relevant is hard for a bot.

6.2 Mallory

Mallory's attack is the opposite of Sam's attack, since she is trying to lower the rank of a legitimate page. Like Sam, Mallory faces a partner agreement problem, where she must agree with her partner for the vote to count.

Lemma 9. *Let $\mathcal{P}'$ be the page Mallory is interested in lowering in the rankings, and $0 \leq p'_s << 1/2$ denote the fraction of the honest population who believe that this page is spam. Let p'_m represent the probability that a snippet from $\mathcal{P}'$ constitutes a question within one game. Then n'_s, the expected number of games Mallory must play to accumulate one vote in the algorithm, is $\frac{1}{p'_s p'_m}$, assuming she does not have enough agents to affect p'_s.*

Proof. Equivalent to Lemma 7. □

Just like with Sam, Mallory must employ a large number of agents to affect the outcome, and has the problem of partner agreement. She must introduce $(1/2 - p'_s)H$ agents. Once more, we assume p'_s is sufficiently low as otherwise the page would not likely be classified as spam under our algorithm. She also is expected to have insufficient number of agents.

Mallory also has a difficult time throwing a page out of the ranking since our algorithm is set up conservatively and keeps a spam page in than remove a good one.

Lemma 10. *Let n'_s be the expected number of games Mallory must play in order to achieve one match on $\mathcal{P}'$, and $100n'_{\mathcal{P}}$ be the number of current votes for page $\mathcal{P}'$ as spam, with $n'_{\mathcal{P}} \in \mathcal{R}^+$. Let $m'_{\mathcal{P}}$ denote the number of users who think that $\mathcal{P}'$ is not spam without Mallory's votes. Then Mallory must expect to need to play $100(m'_{\mathcal{P}} - n'_{\mathcal{P}})n'_s$ times to affect the algorithm.*

Proof. We use a similar derivation to the one in Section 6.1. Once more, let G' be a random variable representing the number of games Mallory must play to affect our algorithm. In order for a page to be thrown out, $100n'_{\mathcal{P}} + G' > 100m'_{\mathcal{P}} \Rightarrow G' > 100(m'_{\mathcal{P}} - n'_{\mathcal{P}})$. By linearity of expectation, Mallory must play $n'_s G' > 100n'_s(m'_{\mathcal{P}} - n'_{\mathcal{P}})$ games in order to affect the game. □

If $m'_{\mathcal{P}}$ is fairly large and $n'_{\mathcal{P}}$ fairly small in comparison, which is to be expected if the page is truly not spam, then Mallory has many obstacles to overcome. She also faces the same issues of recognizing and comparing the snippet that Sam faces.

6.3 Gene

Gene's attack is to corrupt the ranking. For this, Gene should be always voting dishonestly, in an attempt to promote or preserve web spam pages within the rankings, and to remove legitimate pages from the ranking. Gene faces many problems as well.

Anytime Gene attempts to vote a page up, he encounters all the problems Sam does; likewise, whenever Gene tries to vote a page down, he encounters all the problems Mallory does. In essence, by protecting against Sam and Mallory, we also protect against Gene, since Gene simply has their combined interests. With respect to any one question, Gene can be classified as a Sam or a Mallory, and is subject to the guarantees we provide.

In addition to all other roadblocks that Gene inherits from Sam and Mallory, since he is attempting to disagree with rational players, Gene will also disagree with the test questions. Since we assume that humans cannot differentiate between the test and non-test questions, then surely neither can bots without a large advance in natural language processing techniques. Therefore, Gene should have a history of doing extraordinarily poorly on the test questions. We can adapt our algorithm to not count any of Gene's votes after a sufficiently bad history on test questions has been amassed, as he will be considered a rogue player.

6.4 Key assumptions

Any attack the adversaries use must exploit one of the key assumptions.

Users cannot quickly find answers outside of the game We could imagine players launching searches on the shown query to brute force search for the snippet. This is unlikely to succeed though. First, each question is allocated a maximum amount of time t, with the entire game allocated only time T, so the user can be assumed to not have enough time to search every page returned by a search engine. Also, by varying where we question people about pages for the entire API [in the case of Google, this is the first 1000 results] it is also harder for the user to guess where the snippet is pulled from.

Using a different snippet from a standard search engine is another important aspect of the game. By doing so, a user cannot simply launch 100 searches for the query for each of the 10 intervals [assuming 1000 results maximum on an API] and then do a brute force search for the snippet. Instead, they would have to open each page, at the cost of 1000 requests versus 100, and on average, we can time it so that there is not enough time to do this.

A user's confidence well approximates the true probability of two players matching. We define confidence in terms of a user's belief about the state of the game. While this is a valid game theoretic definition, it does require assuming that users are aware of what they know, and can approximate how accurate they believe their guess to be. Nevertheless, this assumption is not fatal to our game because users who play honestly but are poor judges of the relevance of web pages are expected to get large negative scores, and their matches can be regarded as not trustworthy.

Bots perform poorly at natural language processing [NLP]. This assumption is necessary to prevent all the attackers.

The snippet is representative of the page. If this is not true, then Sam could have an indirect attack. If we are not careful in the way we choose the snippet $s_{\mathcal{P}}$, Sam can try to alter his web page so that $s_{\mathcal{P}}$ looks highly relevant to queries but the page is still web spam. This can be fixed easily by not using the same snippet as a normal search engine. By doing this, the expectation that Sam is able to easily discover his game snippet is low, and thus he cannot easily modify his snippet to cheat the game. This does not affect Mallory or Gene.

The test questions are indistinguishable from the rest of the game, and many are available. This is necessary to motivate the bonus calculations and expectation, and to ensure that the game is strategy-proof for players who care about points. Assuming that there are a smaller set of test questions as compared to non-test, a smart adversary who is hoping that we naively use the uniform distribution could write a bot that simply plays many games and scrapes the screen as it goes. Any query-snippet pair that repeats would be a test question.

This can be rectified in many ways. One way is to use a non-uniform distribution on the non-test questions, possibly biasing a small set so that repeats do not necessarily indicate test questions. Another way is to gradually have questions move into the test category when we have enough feedback to know what the "right answer" is, and have questions leave the test category when they have been shown a large amount of time. Finally, we can create a rule that each user can only see a query-snippet pair once as a test question; subsequent views are non-test. All of these strategies make it more difficult for adversaries to try to determine the test set. Alongside these strategies is the fact that $\mathcal{B}$ is individual and not group based, so we can present different test sets to different users [this may make it easier, for example, to allow each user to see each query-snippet pair only once as a test, by keeping a precalculated list of "next test" that is combined to form a game for the users]. Of our adversaries, this primarily affects Gene, since it would allow Gene to vote honestly on test questions and dishonestly on all others.

The query and page are out of the adversaries control. There is no way around this without manipulating the actual game.

The users are paired at random. The only thing the adversaries can do is flood the game with bots, and hope that they add enough bots so that, with high probability, the bots will be paired and can collude. However, bots should play poorly against the test questions, since one key assumption behind this game is that bots cannot process natural language well. We can therefore identify likely bots by looking at their history versus the test questions. This assumption is important for Mallory, Gene and Sam.

The players do not have enough time to research the URL of the page or its location within the ranking. We can fortify this assumption by placing text within images or even CAPTCHAS (especially short, easier to read elements like queries). Also, by drawing from even further down the ranking than we care about, using all 1000 pages [assuming the Google API] in $R_{\mathbb{Q}}$, it becomes time consuming for the adversary to determine where a page is located–they might have to perform 100 searches to find a page. Even then, if we are using a snippet that is separate from the snippet used in web search, it is non-trivial to discover the rank. This mostly matters for Mallory and Sam.

People enjoy playing the game In order for the game to be useful, people have to like playing it and enough people have to play it so that we can extract meaningful data. As described in Section 4.5, we do not anticipate this being a problem.

7 Conclusions and further work

In this paper we presented a two-player game approach to combating web spam. We showed it is strategy-proof and that the information obtained from it can be used to find spam pages. We provided one of the first schemes we are aware of with provable performance guarantees.

There are many advantages to our game. First of all, the likelihood of an attack by an adversary is quite low. Because of this, the votes we do gather from the game are trustworthy. As long as the game is secure, the votes themselves can be considered to come from trusted sources, and a simple voting algorithm that compares quantities of votes for and against a page being spam can be used. As we only ask players to vote a binary choice between spam and not spam, we do not encounter any voting paradoxes.

One major drawback to the game is its dependence on a secret test set. If an adversary can detect which questions are test, and which are not, he can break the game by behaving honestly on test questions, and dishonestly on all others. Some ideas to prevent this were explored in Section 6.4. Also, as described in Section 4.3, we believe it is easy to generate new test questions which an adversary would not have already classified. Further work will focus on implementing the game, testing our assumptions and assessing our success in removing spam from the rankings.

8 Acknowledgments

The authors are extremely grateful to Manuel Blum and Luis von Ahn for their insightful ideas and thoughtful commentary as the work was developed.

Added in press: The following paper relevant to this work has appeared[5].

References

[1] http://ogden.basic-english.org/words.html. Accessed on 11/10/06.

[2] http://code.google.com/apis/soapsearch/.

[3] L. Becchetti, C. Castillo, D. Donato, S. Leonardo, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *AIRWEB*, 2006.

[4] A. Benczúr, K. Csalogány, and T. Sarlós. Link-based similarity search to fight web spam. In *AIRWEB*, 2006.

[5] James Caverlee, Steve Webb, and Ling Liu. Spam-resilient web rankings via influence throttling. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Long Beach, 2007.

[6] Andrew Luiz da Costa Carvalho, Paul-Alexandru Chirita, Edleno Silva de Moura, Pavel Calado, and Wolfgang Nejdl. Site level noise removal for search engines. In *WWW*, 2006.

[7] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *WebDB*, 2004.

[8] Google. Queries pulled from the Google Zeitgeist archive, located at http://www.google.com/press/zeitgeist/archive.html. Accessed on 11/10/06.

[9] Z. Gyongyi and H. Garcia-Molina. Link spam alliances. Technical report, Stanford University, 2005.

[10] Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. In *AIRWEB*, 2005.

[11] Z. Gyongyi, H. Garcia-Molina, P. Berkhin, and J. Pedersen. Link spam detection based on mass estimation. In *VLDB*, 2006.

[12] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB*, 2004.

[13] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *AIRWEB*, 2006.

[14] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW*, 2006.

[15] L. Paige, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[16] David Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, University of Pennsylvania, May 2001.

[17] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *CHI*, 2004.

[18] B. Wu, V. Goel, and B. Davidson. Topical trustrank: Using topicality to combat web spam. In *WWW*, 2006.

[19] Baoning Wu and Brian D. Davison. Identifying link spam farm pages. In *WWW*, 2005.