Data Mining Techniques for Web Spam Detection

Jian Pei*  Bin Zhou*
Zhaohui Tang†  Dylan Huang**
*Simon Fraser University
†Microsoft AdCenter

Outline

• Information retrieval from the web
• Spam tricks
• Spam detection techniques
• Summary and future directions
A Small Survey

- Please raise your hands if you did NOT access internet in the past 7 days
- How do you find the conference web page?
- Please raise your hands if you did NOT use any search engine in the past 7 days

Why Are Search Engines Useful?

- Retrieve practically useful information from the web
  - What is Vancouver?
- Attract potential customers and users
  - Search map of Vancouver
  - Hotels and accommodations in Vancouver
  - City tour
  - ...
Look at This Page

Extracted from [Ntoulas et al. WWW'06]
Web Spam

• Increasing exposure on the World Wide Web may achieve significant financial gains for the web site owners!
  – The increasing importance of search engines to commercial web sites has given rise to a phenomenon called “Web Spam”
• Web Spam: tricks misleading search engines to obtain higher-than-deserved ranking

Basics of Web Search

• Keyword search
  – What are the documents matching query “Vancouver history” the best?
  – TFIDF
• Link-based ranking
  – Among all websites containing keywords “Vancouver” and “history”, how they should be ranked?
  – PageRank, HITS
Keyword Search

• In full text retrieval, all words in a document are considered to be keywords
• Search engines typically allow query expressions formed using keywords and the logical connectives and, or, and not
  – Ands are implicit, even if not explicitly specified

Relevance Ranking

• Term frequency
  – Frequency of occurrence of query keyword in document
• Inverse document frequency
  – How many documents the query keyword occurs in
    • Fewer \( \Rightarrow \) give more importance to keyword
• Hyperlinks to documents
  – More links to a document \( \Rightarrow \) document is more important
TF-IDF

- Term frequency/Inverse Document frequency ranking
- Let \( n(d) \) = number of terms in the document \( d \)
- \( n(d, t) \) = number of occurrences of term \( t \) in the document \( d \)
- Relevance of a document \( d \) to a term \( t \)

\[
TF(d, t) = \log \left( 1 + \frac{n(d, t)}{n(d)} \right)
\]

The log factor is to avoid excessive weight to frequent terms

- Relevance of document to query \( Q \)

\[
r(d, Q) = \sum_{t \in Q} \frac{TF(d, t)}{n(t)}
\]

Relevance Ranking Using Terms

- Most systems also consider
  - Words that occur in title, author list, section headings, etc. are given greater importance
  - Words whose first occurrence is late in the document are given lower importance
  - Very common words (stop words) such as “a”, “an”, “the”, “it” etc are eliminated
  - Proximity: if keywords in query occur close together in the document, the document has higher importance than if they occur far apart

- Documents are returned in decreasing order of relevance score
  - Usually only top few documents are returned, not all
Similarity Based Retrieval

- Similarity based retrieval - retrieve documents similar to a given document
- Similarity may be defined on the basis of common words: e.g. find $k$ terms in $A$ with highest $TF(d, t) / n(t)$ and use these terms to find relevance of other documents

Vector Space Model

- Define an $n$-dimensional space, where $n$ is the number of words in the document set
- Vector for document $d$ goes from origin to a point whose $i^{th}$ coordinate is $TF(d, t) / n(t)$
- The cosine of the angle between the vectors of two documents is used as a measure of their similarity
Relevance Using Hyperlinks

• The number of documents relevant to a query can be enormous if only term frequencies are taken into account
• Using term frequencies makes “spamming” easy
  – E.g. a travel agency can add many occurrences of the words “travel” to its page to make its rank very high
• People often look for pages from popular sites
• Idea: use popularity of Web site (e.g. how many people visit it) to rank site pages that match given keywords
  – Problem: hard to find actual popularity of site

• Use the number of hyperlinks to a site as a measure of the popularity or prestige of the site
  – Count only one hyperlink from each site (why?)
  – Popularity measure is for site, not for individual page
    • But, most hyperlinks are to root of site
    • Also, concept of “site” is difficult to define since a URL prefix like cs.sfu.ca contains many unrelated pages of varying popularity

• Refinements
  – When computing prestige based on links to a site, give more weight to links from sites that themselves have higher prestige
    • Definition is circular
    • Set up and solve system of simultaneous linear equations
PageRank

\[ PR(a) = q + (1-q) \sum_{i=1}^{n} \frac{PR(p_i)}{C(p_i)} \]

- Simulate a user navigating randomly in the web who jumps to a random page with probability \( q \) or follows a random hyperlink with probability \( 1-q \)
- \( C(a) \) is the number of outgoing links of page \( a \)
- Page \( a \) is pointed to by pages \( p_1 \) to \( p_n \)

Relevance Using Hyperlinks

- Connections to social networking theories that ranked prestige of people
  - E.g. the president of the U.S.A has a high prestige since many people know him
- Someone known by multiple prestigious people has high prestige
Rethinking Search Engines

- High recall, low precision
  - Many mildly relevant or irrelevant documents may be returned
  - “Too much can easily become as bad as too little”
- Low or no recall, often when combinations of keywords are used
- Results are highly sensitive to vocabulary
  - A search engine does not know “XML data” is “semi-structured data”
- Results are single web pages
  - How to find information spread over various documents, e.g., a survey on the latest XML initiatives

HITS: Capturing Authorities & Hubs

- Intuition
  - Many rivals, such as Toyota and Honda, do not cite each other on the Internet
  - Pages that are widely cited (i.e., many in-links) are good authorities
  - Pages that cite many other pages (i.e., many out-links) are good hubs
  - Authorities and hubs have a mutual reinforcement relationship
- The key idea of HITS (Hypertext Induced Topic Search)
  - Good authorities are cited by good hubs
  - Good hubs point to good authorities
  - Iterative reinforcement …
HITS: Strength and Weakness

- Advantages: Rank pages according to the query topic
- Disadvantages
  - Does not have anti-spam capability: One may add out-links to his own page that points to many good authorities
  - Topic-drift: One may collect many pages that have nothing to do with the topic — by just pointing to them
  - Query-time evaluation: expensive

Improvements on HITS

- SALA [Lemple & Moran, WWW’00], a stochastic algorithm, two Markov chains, an authority and a hub Markov chains, less susceptible to spam
- Weight the links [Bharat & Henzinger SIGIR’98]: if there are k edges from documents on a first host to a single document on a second host, give each edge an authority weight of 1/k, …
- Handling topic drifting: Content similarity comparison, or segment the page based on the DOM (Document Object Model) tree structure to identify blocks or sub-trees that are more relevant to query topic
Link Spam

- PageRank
  \[ PR(p, G) = d \sum_{p_i \in M(p)} \frac{PR(p_i, G)}{OutDeg(p_i)} + \frac{1-d}{N} \]
- Link spam refers to deliberately build auxiliary pages and links to boost the PageRank or other link-based ranking score of the target page.
- Those structures are referred to as link spam farms

Term Spam

- TFIDF
  - Given a web page \( p \) and a search query \( Q \)
  \[ TFIDF(p, Q) = \sum_{t \in p \cap Q} TF(t) \times IDF(t) \]
- Term spam refers to tricks that tailor the contents of text fields to make spam pages relevant for some queries
- The primary way to increase the score is to increase the frequencies of keywords within some specific text fields of the term spam pages
Web Spam Taxonomy

- Term spam
  - Add many keywords into one page
  - Make those keywords invisible but searchable
- Link spam
  - Construct links to mislead search engines
- Both tricks are often used together

Data Mining and Spam Detection

- Classification approaches
- PageRank-like approaches
- Spam mass and spamicity approaches
Classification and Prediction

- **Classification**: predict categorical class labels
  - Build a model for a set of classes/concepts
  - Classify whether a page is web spam
- **Prediction**: model continuous-valued functions
  - Predict the economic growth in 2008

A Two-step Process

- **Model construction**: describe a set of predetermined classes
  - Training dataset: tuples for model construction
    - Each tuple/sample belongs to a predefined class
  - Classification rules, decision trees, or math formulae
- **Model application**: classify unseen objects
  - Estimate accuracy of the model using an independent test set
  - Acceptable accuracy → apply the model to classify tuples with unknown class labels
Model Construction

![Diagram of model construction process]

**Training Data**

<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Years</th>
<th>Tenured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Ass. Prof</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>Mary</td>
<td>Ass. Prof</td>
<td>7</td>
<td>Yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Prof</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Asso. Prof</td>
<td>7</td>
<td>Yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Ass. Prof</td>
<td>6</td>
<td>No</td>
</tr>
<tr>
<td>Anne</td>
<td>Asso. Prof</td>
<td>3</td>
<td>No</td>
</tr>
</tbody>
</table>

**Classification Algorithms**

IF rank = 'professor' OR years > 6 THEN tenured = 'yes'

Model Application

![Diagram of model application process]

**Testing Data**

<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Years</th>
<th>Tenured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Ass. Prof</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>Merlisa</td>
<td>Asso. Prof</td>
<td>7</td>
<td>No</td>
</tr>
<tr>
<td>George</td>
<td>Prof</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Asso. Prof</td>
<td>7</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Unseen Data**

(Jeff, Professor, 4)

Tenured? **Yes**
Decision Tree

- A node in the tree – a test of some attribute
- A branch: a possible value of the attribute
- Classification
  - Start at the root
  - Test the attribute
  - Move down the tree branch

Training Dataset

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Basic Algorithm ID3

- Construct a tree in a top-down recursive divide-and-conquer manner
  - Which attribute is the best at the current node?
  - Create a node for each possible attribute value
  - Partition training data into descendant nodes
- Conditions for stopping recursion
  - All samples at a given node belong to the same class
  - No attribute remained for further partitioning
    - Majority voting is employed for classifying the leaf
  - There is no sample at the node

Which Attribute Is the Best?

- The attribute most useful for classifying examples
- Information gain and gini index
  - Statistical properties
  - Measure how well an attribute separates the training examples
Entropy

- Measure homogeneity of examples

\[ \text{Entropy}(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \]

- \( S \) is the training data set, and \( p_i \) is the proportion of \( S \) belong to class \( i \)

- The smaller the entropy, the purer the data set

Information Gain

- The expected reduction in entropy caused by partitioning the examples according to an attribute

\[ \text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \]

- \( \text{Value}(A) \) is the set of all possible values for attribute \( A \), and \( S_v \) is the subset of \( S \) for which attribute \( A \) has value \( v \)
Example

\[ \text{Entropy}(S) = - \left( \frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94 \]

\[ \text{Gain}(S, \text{Wind}) = \text{Entropy}(S) - \sum_{S_c \in \{\text{Weak, Strong}\}} \frac{|S_c|}{|S|} \text{Entropy}(S_c) \]

\[ = \text{Entropy}(S) - \frac{8}{14} \text{Entropy}(S_{\text{Weak}}) - \frac{6}{14} \text{Entropy}(S_{\text{Strong}}) \]

\[ = 0.94 - \frac{8}{14} \times 0.811 - \frac{6}{14} \times 1.00 = 0.048 \]

Extracting Classification Rules

- Each path from the root to a leaf \( \rightarrow \) an IF-THEN rule
  - Each attribute-value pair along a path forms a conjunction
  - The leaf node holds the class prediction
  - IF age = "\(<=30" \) AND student = "no" \( \rightarrow \) THEN buys_computer = "no"
- Rules are easy to understand
Bagging

- Given a set $S$ of $s$ samples, generate a sequence of $k$ independent bootstrap training sets
- Construct a sequence of classifiers $C_1, C_2, \ldots, C_k$ by using the same classification algorithm
- To classify an unknown sample $X$, let each classifier predict or vote
- The bagged classifier $C^*$ counts the votes and assigns $X$ to the class with the “most” votes

Boosting Technique

- Assign every example an equal weight $1/N$
- For $t = 1, 2, \ldots, T$ Do
  - Obtain a classifier $C(t)$ under $w(t)$
  - Calculate the error of $C(t)$ and re-weight the examples based on the errors. Samples incorrectly predicted have bigger weight
- Output a weighted sum of all the classifiers, with each classifier weighted according to its accuracy on the training set
Spam Detection by Classification

- Use a set of spam web pages as a training data set
- Train a classification model (e.g., a decision tree)
- Apply the classification model to combat web spam

Heuristic Feature Selection

- Web page top domains
- Languages
- Number of words (body and title)
- Average word length
- Anchor words
- Visibility of content
- Repeating keywords
- The most common keywords
- N-gram likelihood
- [Ntoulas et al. WWW'06]
Web Page Top Domains

![Bar chart showing the percentage of spam by top domains.]

Languages

![Bar chart showing the percentage of spam by language.]

J. Pei, B. Zhou, Z. Tang, and D. Huang: Data Mining Techniques for Spam Detection
Number of Words

Average Word Length
Fraction of Anchor Words

- Anchor words: words for hyperlinks

Visibility of Content
Repeating Keywords

Most Common Keywords
Using C4.5 to Combine Features

• Using bagging and boosting

<table>
<thead>
<tr>
<th>class</th>
<th>recall</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>82.1%</td>
<td>84.2%</td>
</tr>
<tr>
<td>non-spam</td>
<td>97.5%</td>
<td>97.1%</td>
</tr>
</tbody>
</table>

Table 1: Recall and precision of our classifier

<table>
<thead>
<tr>
<th>class</th>
<th>recall</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>86.2%</td>
<td>91.1%</td>
</tr>
<tr>
<td>non-spam</td>
<td>98.7%</td>
<td>97.8%</td>
</tr>
</tbody>
</table>

Table 2: Recall and precision after bagging

<table>
<thead>
<tr>
<th>class</th>
<th>recall</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>spam</td>
<td>82.1%</td>
<td>84.2%</td>
</tr>
<tr>
<td>non-spam</td>
<td>97.5%</td>
<td>97.1%</td>
</tr>
</tbody>
</table>

Table 3: Recall and precision after boosting.

SpamRank: Ideas

• Supporters of an honest (non-spam) page should not be overly dependent on one another

• The PageRank of the supporters of an honest page should follow a power law distribution as if a sample of the whole web

• Link spammers have a limited budget – boosting utility is important for supporters of spam pages

• [Benczur et al. AIRWeb’05]
SpamRank: A Three-Step Method

- Phase 1: select the supporters of each page by a Monte Carlo simulation
- Phase 2: pages are penalized if their supporters do not follow power law distribution in PageRank histogram
- Phase 3: compute SpamRank as PageRank personalized on the vector of penalties

PageRank versus SpamRank
TrustRank: Ideas and Method

- Honest pages often point to honest pages and seldom point to spam pages
- Use a set of known honest pages as the seed set
  - Assign high trust scores to those pages
- Propagate the trust scores via out-links to unknown web pages – a PageRank computation procedure
- When the TrustRank converge, pages with high TrustRank scores are honest pages
- Critical issue: the seed set must be good and balanced
- [Gyongyi et al. VLDB’04]

PageRank versus TrustRank

- Good pages
PageRank versus TrustRank

- Bad pages

Precision and Recall
Topical TrustRank

• General TrustRank has a bias towards heavily represented communities in the seed set
• Use pages in well-maintained topic directories such as dmoz Open Directory Project as the seed set
  – Partition the seed set into topics
• Compute TrustRank score vectors on topics
• [Wu et al. WWW’06]

TrustRank versus Topical TrustRank
Spam Farms

• The set of pages supporting a spam page
• Three components
  – A single target page to be boosted by the spammer
  – A reasonable number of boosting pages that deliberately push the ranking of the target page
  – Some external links accumulated from pages outside the spam farm
• [Gyongyi and Garcia-Molina, VLDB’05]
Spam Alliance

- A spam farm may boost multiple target pages

Irregular Spam Alliance
Questions Remained

• How can we derive spam farms in the real web?
• A spam page may play both link spam and content spam tricks?
• Is spamming as simple as black-and-white?

A Spamicity Approach

• Use spamicity to measure how likely a web page is spam
• Efficient spamicity-based link spam detection methods
• Efficient spamicity-based term spam detection methods
• [Zhou et al. SDM’08]
Page Farm Model

- Typically, link spam is a local activity.
  - Where does $PR(p, G)$ come from?

$$PR(p, G) = d \sum_{p_i \in M(p)} \frac{PR(p_i, G)}{OutDeg(p_i)} + \frac{1 - d}{N}$$

- ($\theta,k$)-page farm of page $p$: the minimal set of pages contributing to a $\theta$ portion of $PR(p, G)$ and each page has a distance to $p$ at most $k$
  - According to [Zhou and Pei, SDM'07], when $\theta \geq 0.8$ and $k \geq 3$, the farms captures the local environments of web pages accurately

Utility-based Link Spamicity

- Given a page $p$, its page farm Farm($p$) captures its local link structures
- Farm($p$) should try to achieve the PageRank of $p$ as high as possible
- The utility of Farm($p$) is the ratio of the PageRank of $p$ against the maximum PageRank that can be achieved
Optimal Spam Farms

\[ ULSpam(p) = \frac{PR(p)}{PR_{\text{max}}(|V|, |E|)} \]

- ULSpam(p) can be used as a measure on the likelihood that p is link spam
  - It is an objective measure
  - It also works for those disguised link spam
Link Spam Detection Scenarios

• When the whole web graph is available
  – Search engine companies
  – Parties who have the access to data (e.g., by crawling the web)
  – But, the maintenance of the data is a big issue

• When the whole web graph is unavailable
  – Online spam detection (e.g., intelligent web browsers)
  – Efficient spam detection (e.g., only want to label a small set of pages)
    • Out-links: parsing the content of the page
    • In-links: querying web search engines using link search queries

Efficient Link Spam Detection

• Given a link spamicity threshold and a web page
  – Determine whether the link spamicity of the page is greater than or equal to the threshold

• Major calculation costs
  – Search engine querying load
  – Web page out-link parsing load
Local Greedy Search Method

- Page contribution
  \[ PCont(v, p) = \begin{cases} 
  PR(p, G) - PR(p, G \setminus \{v\}) & (v \neq p) \\
  \frac{1}{N} & (v = p)
  \end{cases} \]

- Path contribution
  - Consider a path
  \[ p = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_n \rightarrow p \]
  \[ LCont(P, p) = \frac{1}{N} d^{n+1} (1 - d) \prod_{i=0}^{n} \frac{1}{\text{OutDeg}(v_i)} \]

- Page contribution and path contribution
  - \( PCont(v, p) \) can be calculated efficiently by summing up \( LCont(P, p) \)

- A local greedy search method
  - Given a target page \( p \), greedily add pages with the highest page contribution to \( p \) into the farm \( Farm(p) \)
  - The procedure stops until \( Farm(p) \) achieves a \( \theta \) portion of the PageRank score of \( p \)

Monotone Greedy Search Method

- The local greedy search method needs to extract the whole farm so as to calculate the link spamicity

- A critical observation: If pages are added in the page contribution descending order, the utility of adding new pages to improve the PageRank of the target page decreases monotonically

- A monotone greedy search method
  - Given a target page \( p \), greedily add a page to the current farm \( Farm(p) \) which makes the largest improvement on \( PR(p) \)
  - The iteration continues until the link spamicity is lower than the link utility threshold, or all the pages within distance to \( p \) up to \( k \) are in the farm
Utility-based Term Spamicity

- If page \( p \) is term spam, to be relevant to a search query \( Q \), \( p \) should try to achieve the TFIDF score as high as possible.
- The keywords in page \( p \) can be treated as the targeted keywords to which the builder of the page wants to make \( p \) relevant.
- Utility-based term spamicity
  \[
  UT_{Spam}(p) = \frac{TFIDF(p, Q)}{TFIDF_{max}(p)}
  \]
- \( UT_{Spam}(p) \) can be used as a measure on the likelihood that \( p \) is term spam
  - It is an objective measure

Char-Based Term Spamicity

- Keyword stuffing detection
  - Page body, page title, page meta tags, page anchor text
  - \( H_i(p) \) (i=1,2,3,4): the ratio of the total number of keywords in each field against the number of distinct keywords in each field
- Invisible keywords detection
  - Set the keywords to have the same color as the page body
  - \( H_5(p) \): the ratio of the number of invisible keywords in the body against the total number of keywords in the body
- Page URL keywords detection
  - Embed spam keywords in the URL address of the page.
  - \( H_6(p) \): the ratio of the total length of keywords in the URL against the total length of the URL
- Characteristics-based term spamicity
  \[
  CT_{Spam}(p) = \sqrt{\frac{\sum_{i=1}^{6} H_i(p)^2}{6}}
  \]
Efficient Term Spam Detection

• Given a term spamicity threshold and a web page
  – Determine whether the term spamicity of the page is greater than or equal to the threshold

• Major calculation costs
  – Web page keyword parsing load
  – Search engine querying load
  – IDF scores of keywords

Data Set

• The webspam-UK2006 data set, released by Yahoo! Research Barcelona
• 8,239 pages are labeled manually, either “spam” or “normal”
The Effectiveness of Spamicity

![Graph showing the number of pages against spamicity]

Content Spam Detection

![Graphs showing the number of pages in different categories]
Comparisons of Three Spamicities

Scalability
Summary

• Web spam hurts information retrieval quality on the web
  – Link spam
  – Content spam

• Can data mining techniques help in web spam detection?
  – Classification approaches
  – PageRank-like approaches
  – Spam mass and spamicity approaches

Future Directions

• Effectiveness
  – More accurate spam detection?

• Efficiency
  – Scalable and online spam detection?

• PageRank is not all about web information retrieval
  – Spam detection for other ranking methods?
  – Spam detection for search of other types of data, e.g., images, videos, news, shopping, …
References (1)


References (2)


References (3)


References (4)


References (5)


References (6)


References (7)


References (8)


References (9)


References (10)


References (11)


References (12)


References (13)
