US 20050108340A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0108340 A1

Gleeson et al. (43) Pub. Date: May 19, 2005

(54) METHOD AND APPARATUS FOR FILTERING EMAIL SPAM BASED ON SIMILARITY MEASURES

(76) Inventors: **Matt Gleeson**, San Francisco, CA (US); **David Hoogstrate**, San Francisco, CA (US); **Sandy Jensen**, Berkyley, CA (US); **Eli Mantel**, Palo Alto, CA (US); **Art Medlar**, Berkeley, CA (US); **Ken Schneider**, San Francisco, CA (US)

Correspondence Address:
BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030 (US)

(21) Appl. No.: **10/846,723**

(22) Filed: **May 13, 2004**

Related U.S. Application Data

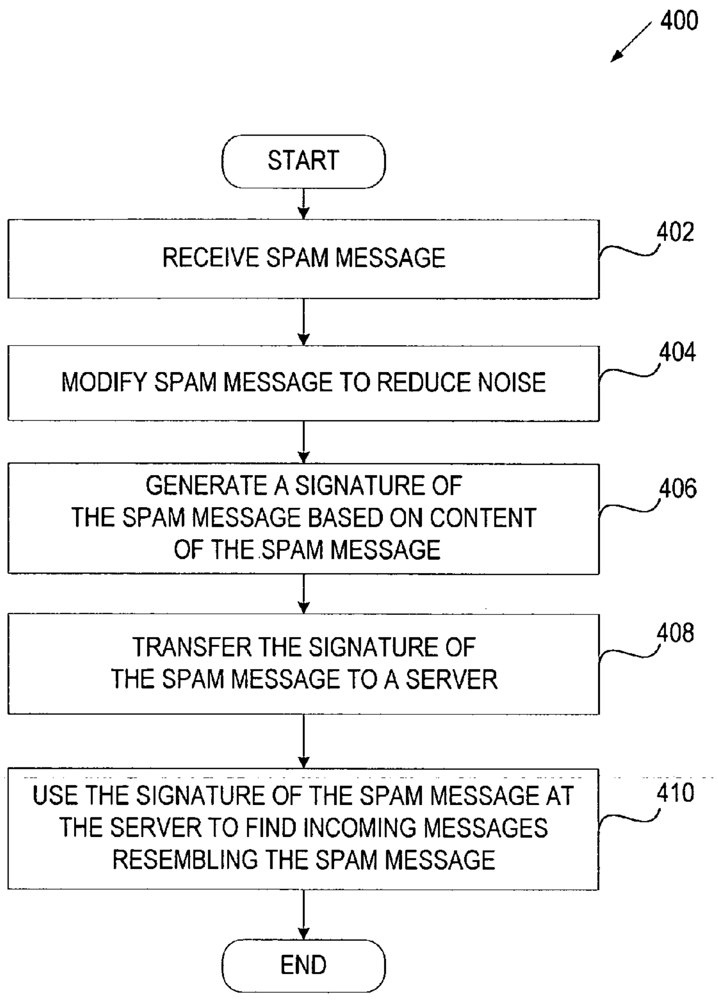
(60) Provisional application No. 60/471,242, filed on May 15, 2003.

Publication Classification

(51) Int. Cl.^{7} **G06F 15/16**
(52) U.S. Cl. **709/206**

(57) **ABSTRACT**

A method and system for filtering email spam based on similarity measures are described. In one embodiment, the method includes receiving an incoming email message, generating data characterizing the incoming email message based on the content of the incoming email message, and comparing the generated data with a set of data characterizing spam messages. The method further includes determining whether a resemblance between the data characterizing the incoming email message and any data item from the set of data characterizing spam messages exceeds a threshold.


400
START
RECEIVE SPAM MESSAGE
402
MODIFY SPAM MESSAGE TO REDUCE NOISE
404
GENERATE A SIGNATURE OF THE SPAM MESSAGE BASED ON CONTENT OF THE SPAM MESSAGE
406
TRANSFER THE SIGNATURE OF THE SPAM MESSAGE TO A SERVER
408
USE THE SIGNATURE OF THE SPAM MESSAGE AT THE SERVER TO FIND INCOMING MESSAGES RESEMBLING THE SPAM MESSAGE
410
END

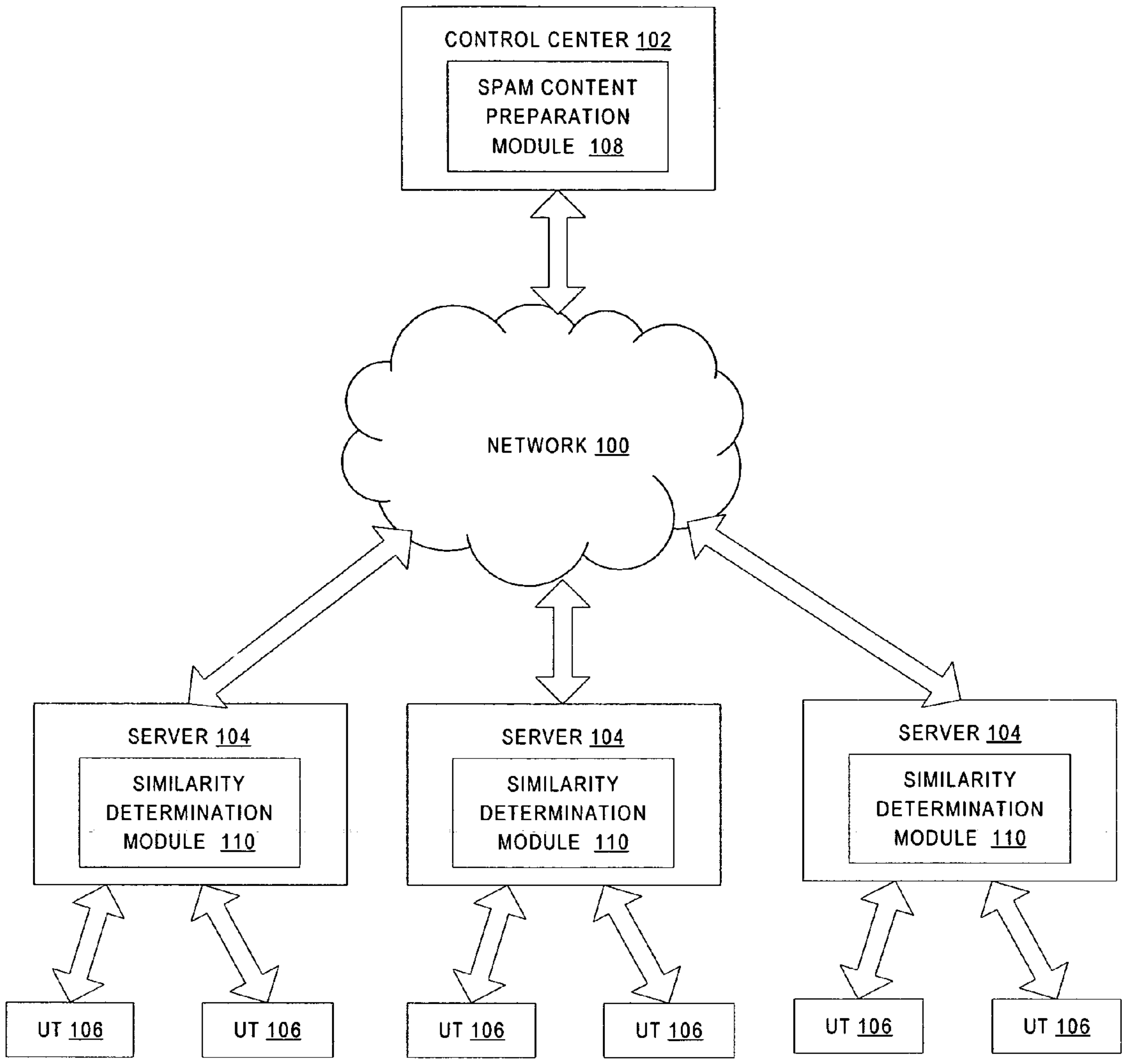

CONTROL CENTER 102
SPAM CONTENT PREPARATION MODULE 108
NETWORK 100
SERVER 104
SIMILARITY DETERMINATION MODULE 110
SERVER 104
SIMILARITY DETERMINATION MODULE 110
SERVER 104
SIMILARITY DETERMINATION MODULE 110
UT 106
UT 106
UT 106
UT 106
UT 106
UT 106

FIG. 1

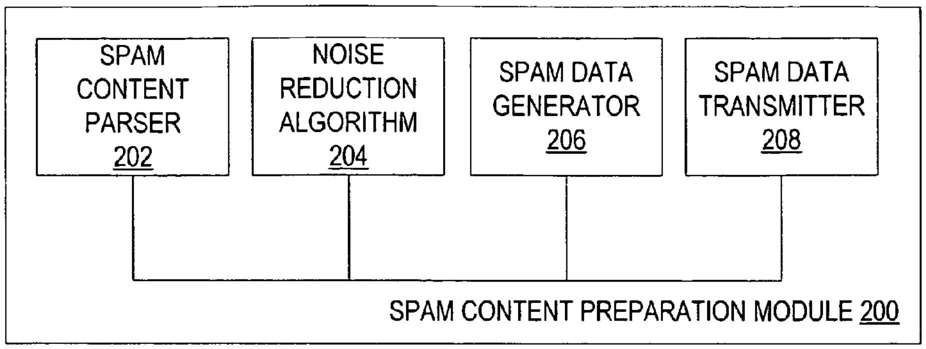

SPAM CONTENT PARSER 202
NOISE REDUCTION ALGORITHM 204
SPAM DATA GENERATOR 206
SPAM DATA TRANSMITTER 208
SPAM CONTENT PREPARATION MODULE 200

FIG. 2

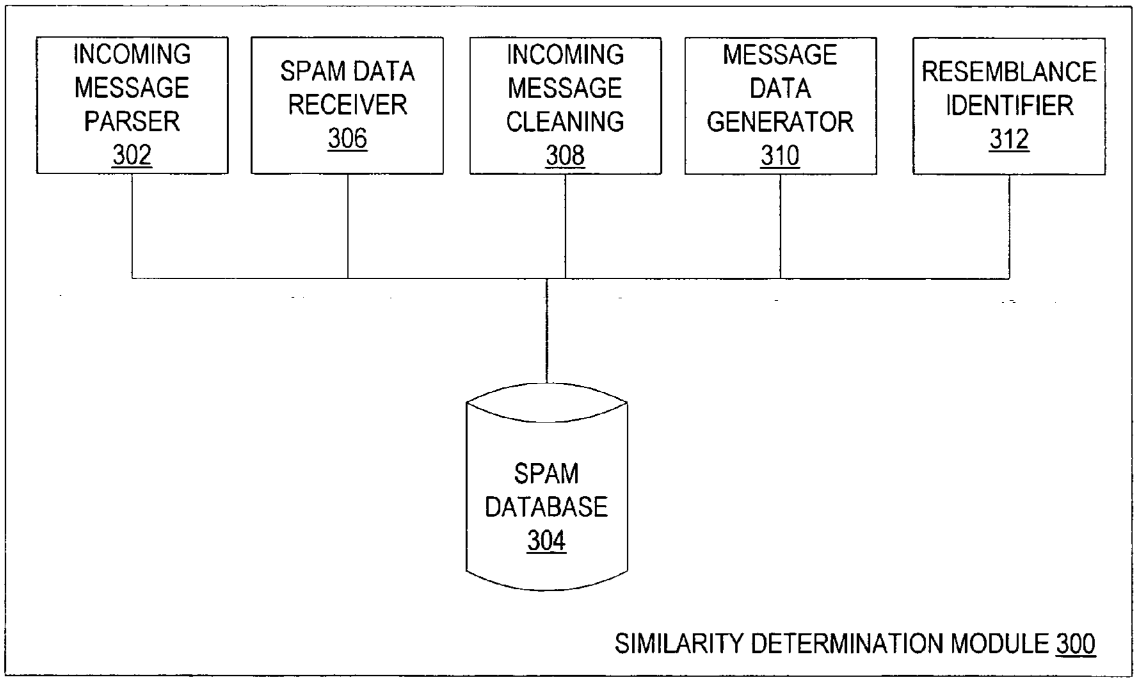

INCOMING MESSAGE PARSER 302
SPAM DATA RECEIVER 306
INCOMING MESSAGE CLEANING 308
MESSAGE DATA GENERATOR 310
RESEMBLANCE IDENTIFIER 312
SPAM DATABASE 304
SIMILARITY DETERMINATION MODULE 300

FIG. 3

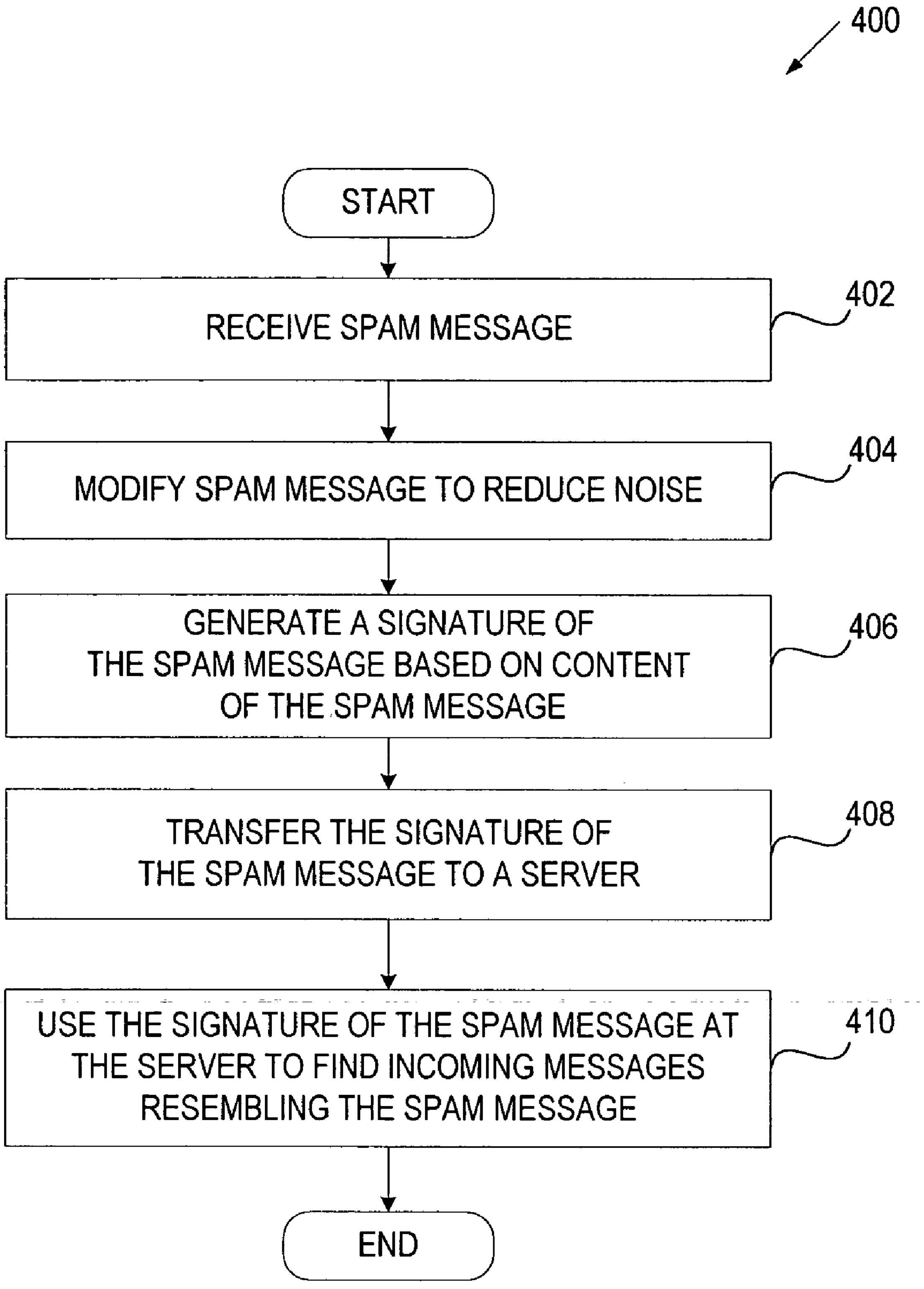

400
START
RECEIVE SPAM MESSAGE
402
MODIFY SPAM MESSAGE TO REDUCE NOISE
404
GENERATE A SIGNATURE OF THE SPAM MESSAGE BASED ON CONTENT OF THE SPAM MESSAGE
406
TRANSFER THE SIGNATURE OF THE SPAM MESSAGE TO A SERVER
408
USE THE SIGNATURE OF THE SPAM MESSAGE AT THE SERVER TO FIND INCOMING MESSAGES RESEMBLING THE SPAM MESSAGE
410
END

FIG. 4

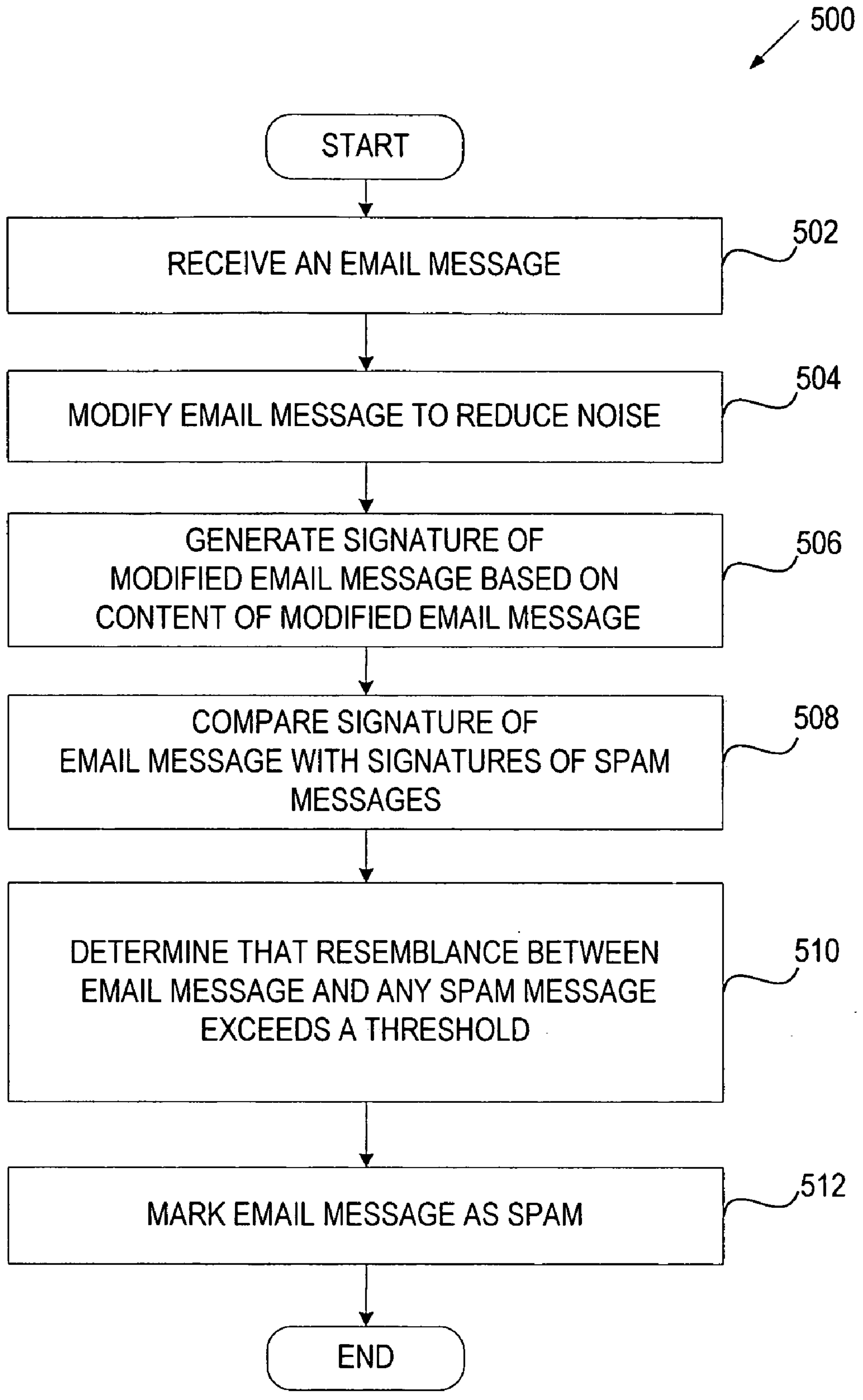

500
START
RECEIVE AN EMAIL MESSAGE
502
MODIFY EMAIL MESSAGE TO REDUCE NOISE
504
GENERATE SIGNATURE OF MODIFIED EMAIL MESSAGE BASED ON CONTENT OF MODIFIED EMAIL MESSAGE
506
COMPARE SIGNATURE OF EMAIL MESSAGE WITH SIGNATURES OF SPAM MESSAGES
508
DETERMINE THAT RESEMBLANCE BETWEEN EMAIL MESSAGE AND ANY SPAM MESSAGE EXCEEDS A THRESHOLD
510
MARK EMAIL MESSAGE AS SPAM
512
END

FIG. 5

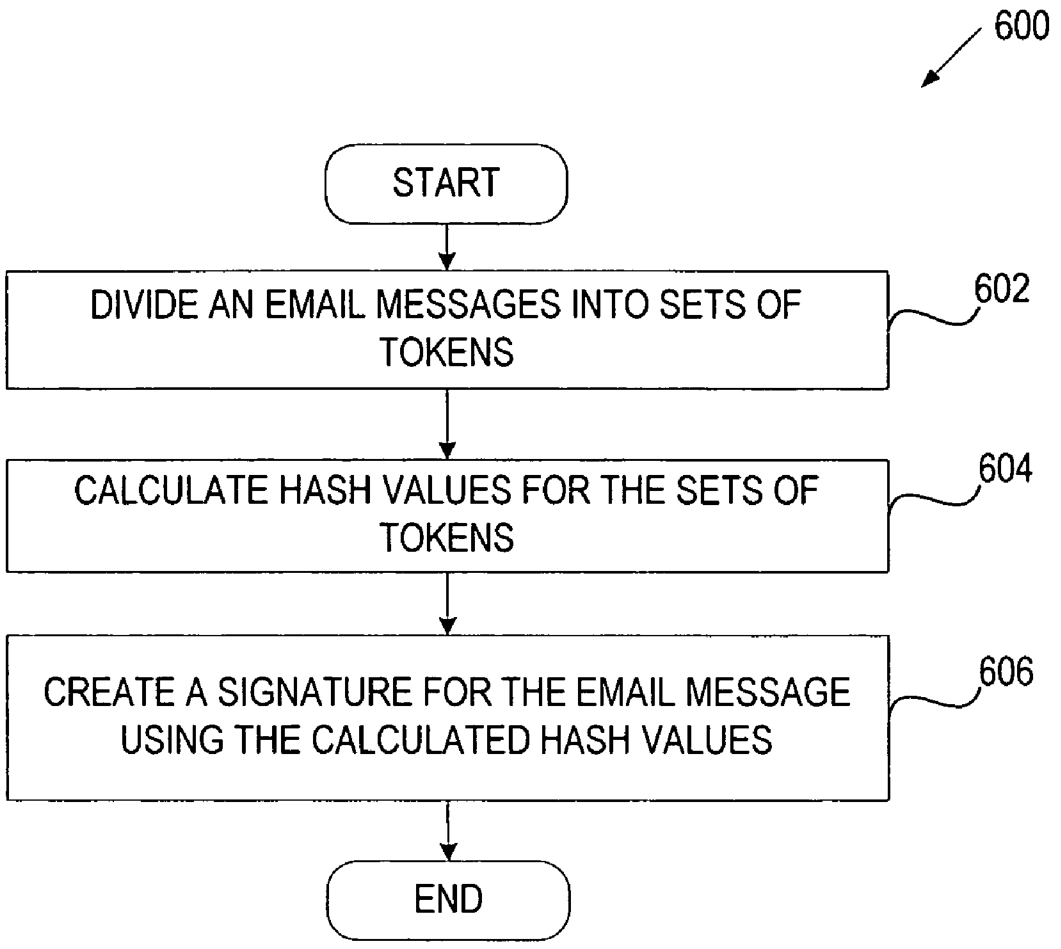

600
START
DIVIDE AN EMAIL MESSAGES INTO SETS OF TOKENS
602
CALCULATE HASH VALUES FOR THE SETS OF TOKENS
604
CREATE A SIGNATURE FOR THE EMAIL MESSAGE USING THE CALCULATED HASH VALUES
606
END

FIG. 6A

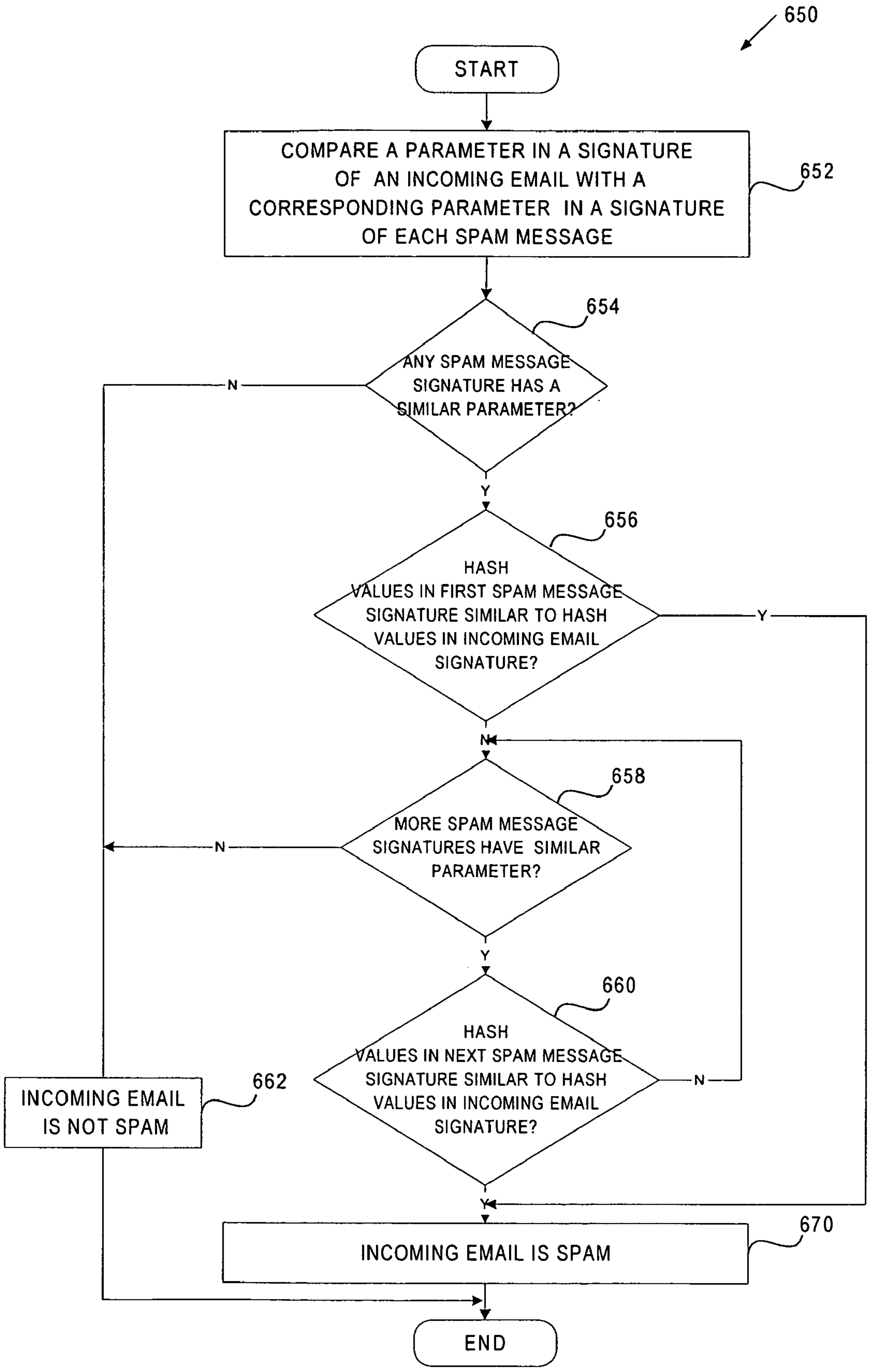

650
START
COMPARE A PARAMETER IN A SIGNATURE OF AN INCOMING EMAIL WITH A CORRESPONDING PARAMETER IN A SIGNATURE OF EACH SPAM MESSAGE
652
ANY SPAM MESSAGE SIGNATURE HAS A SIMILAR PARAMETER?
654
N
Y
HASH VALUES IN FIRST SPAM MESSAGE SIGNATURE SIMILAR TO HASH VALUES IN INCOMING EMAIL SIGNATURE?
656
Y
N
MORE SPAM MESSAGE SIGNATURES HAVE SIMILAR PARAMETER?
658
N
Y
HASH VALUES IN NEXT SPAM MESSAGE SIGNATURE SIMILAR TO HASH VALUES IN INCOMING EMAIL SIGNATURE?
660
N
Y
INCOMING EMAIL IS NOT SPAM
662
INCOMING EMAIL IS SPAM
670
END

FIG. 6B

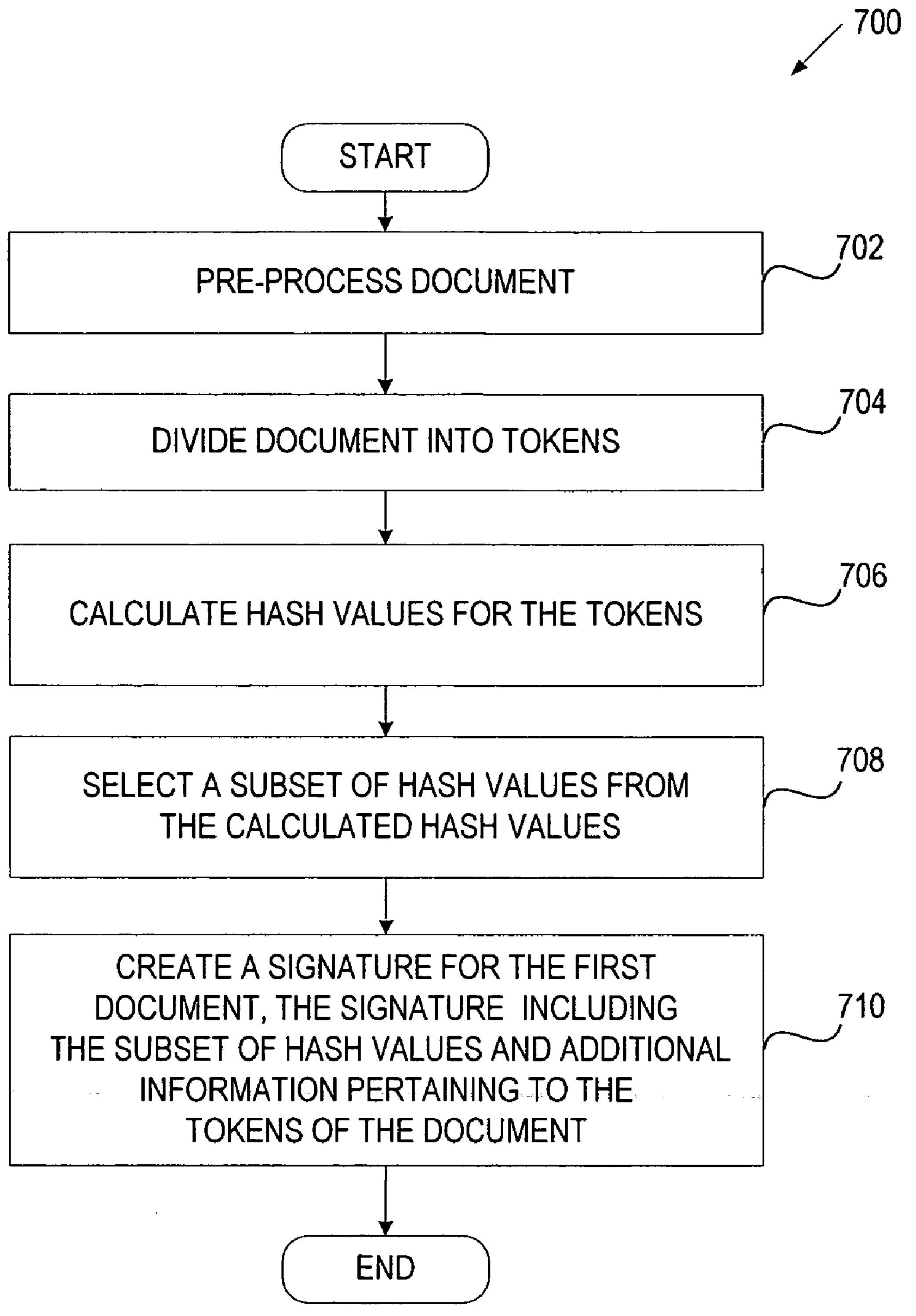

700
START
PRE-PROCESS DOCUMENT
702
DIVIDE DOCUMENT INTO TOKENS
704
CALCULATE HASH VALUES FOR THE TOKENS
706
SELECT A SUBSET OF HASH VALUES FROM THE CALCULATED HASH VALUES
708
CREATE A SIGNATURE FOR THE FIRST DOCUMENT, THE SIGNATURE INCLUDING THE SUBSET OF HASH VALUES AND ADDITIONAL INFORMATION PERTAINING TO THE TOKENS OF THE DOCUMENT
710
END

FIG. 7

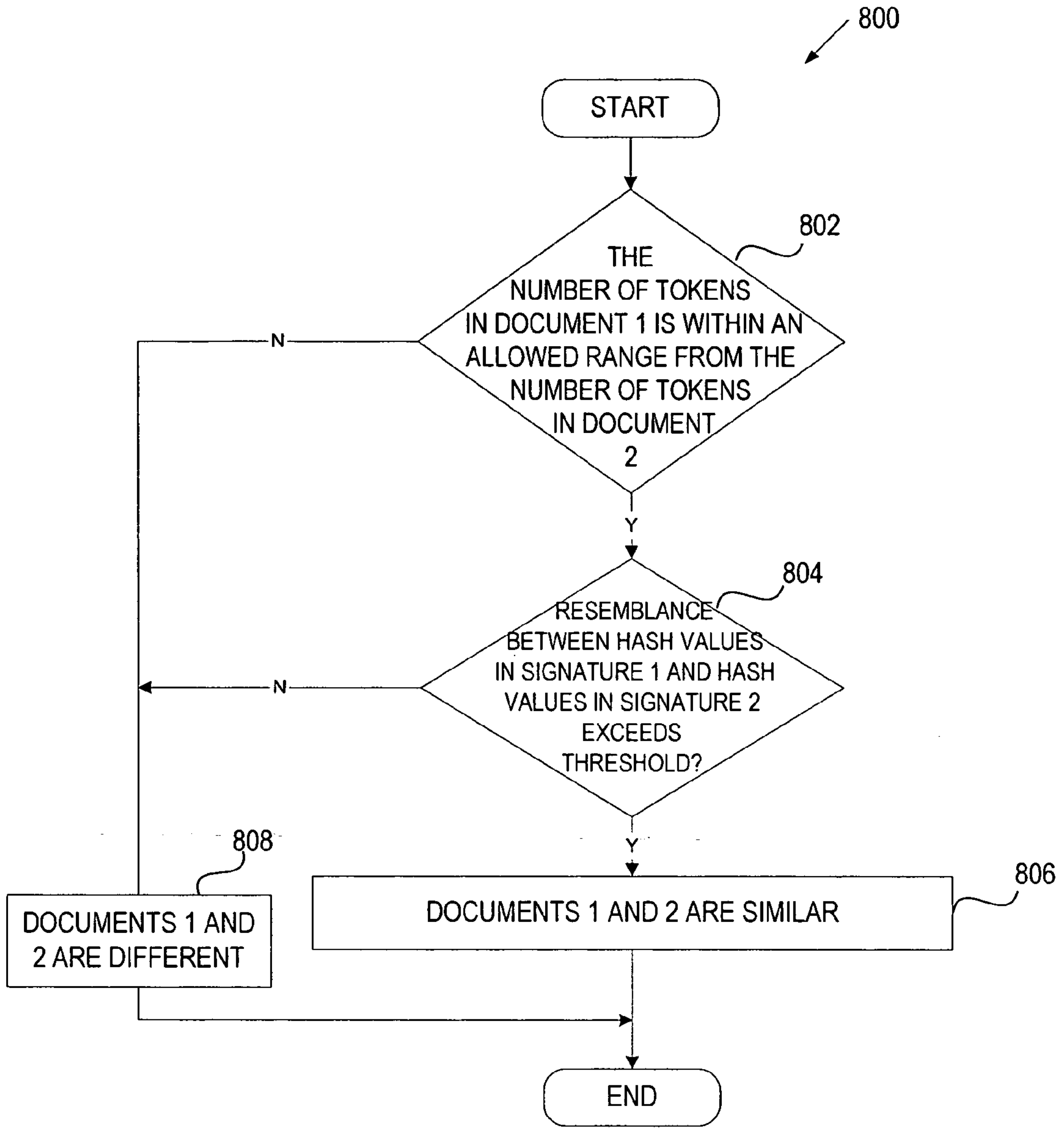

800
START
THE NUMBER OF TOKENS IN DOCUMENT 1 IS WITHIN AN ALLOWED RANGE FROM THE NUMBER OF TOKENS IN DOCUMENT 2
802
N
Y
RESEMBLANCE BETWEEN HASH VALUES IN SIGNATURE 1 AND HASH VALUES IN SIGNATURE 2 EXCEEDS THRESHOLD?
804
N
Y
808
DOCUMENTS 1 AND 2 ARE DIFFERENT
DOCUMENTS 1 AND 2 ARE SIMILAR
806
END

FIG. 8

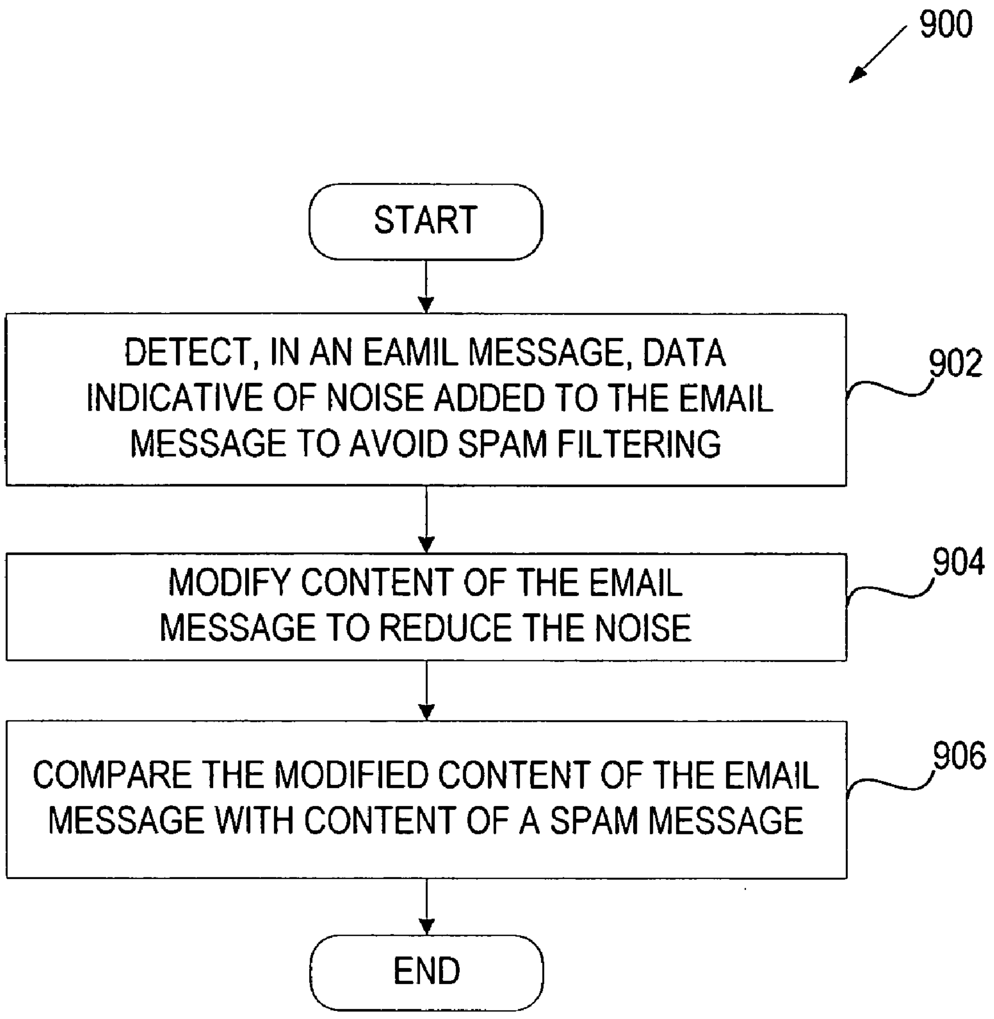

900
START
DETECT, IN AN EAMIL MESSAGE, DATA INDICATIVE OF NOISE ADDED TO THE EMAIL MESSAGE TO AVOID SPAM FILTERING
902
MODIFY CONTENT OF THE EMAIL MESSAGE TO REDUCE THE NOISE
904
COMPARE THE MODIFIED CONTENT OF THE EMAIL MESSAGE WITH CONTENT OF A SPAM MESSAGE
906
END

FIG. 9

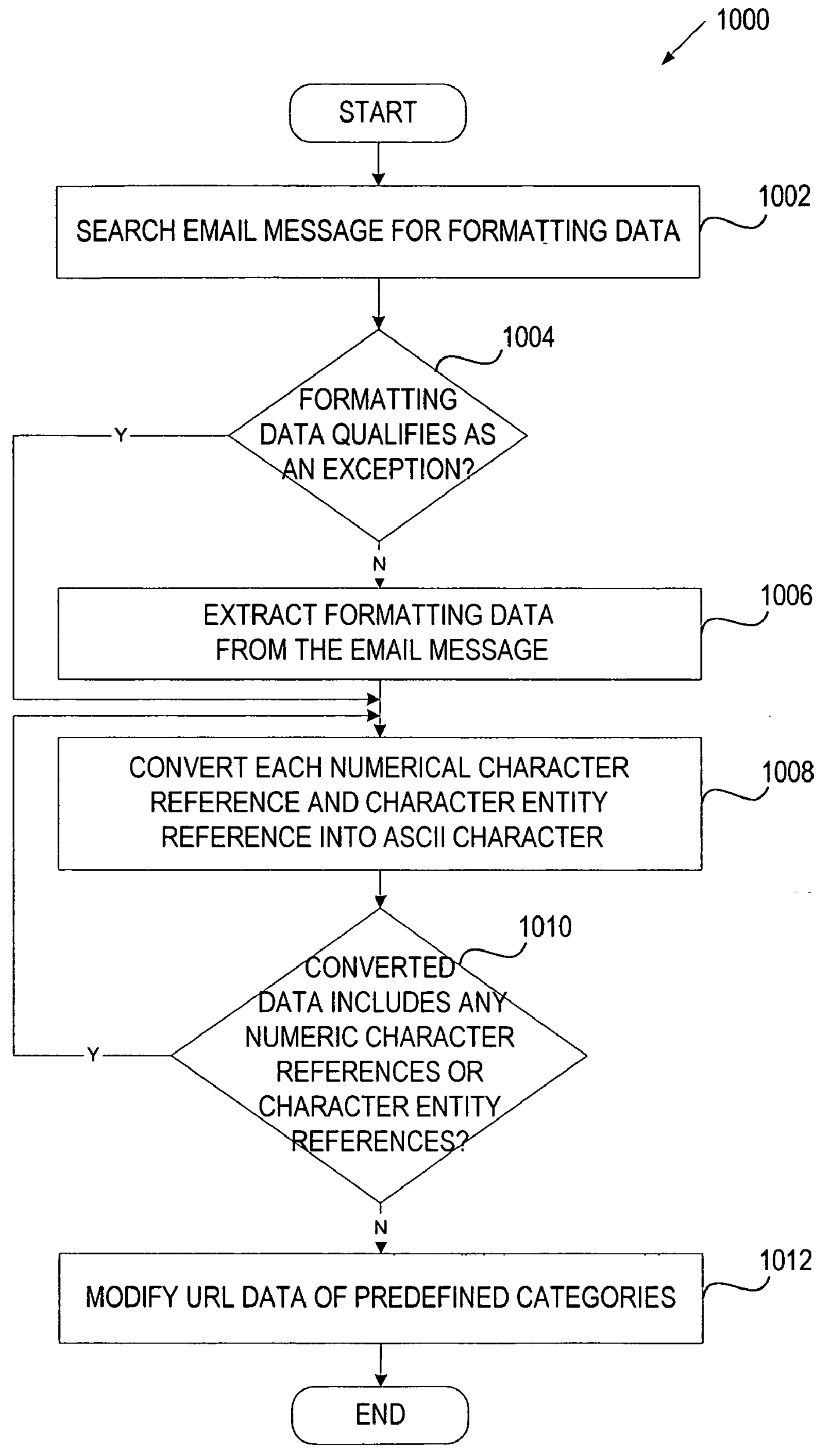

1000
START
SEARCH EMAIL MESSAGE FOR FORMATTING DATA
1002
FORMATTING DATA QUALIFIES AS AN EXCEPTION?
1004
Y
N
EXTRACT FORMATTING DATA FROM THE EMAIL MESSAGE
1006
CONVERT EACH NUMERICAL CHARACTER REFERENCE AND CHARACTER ENTITY REFERENCE INTO ASCII CHARACTER
1008
CONVERTED DATA INCLUDES ANY NUMERIC CHARACTER REFERENCES OR CHARACTER ENTITY REFERENCES?
1010
Y
N
MODIFY URL DATA OF PREDEFINED CATEGORIES
1012
END

FIG. 10

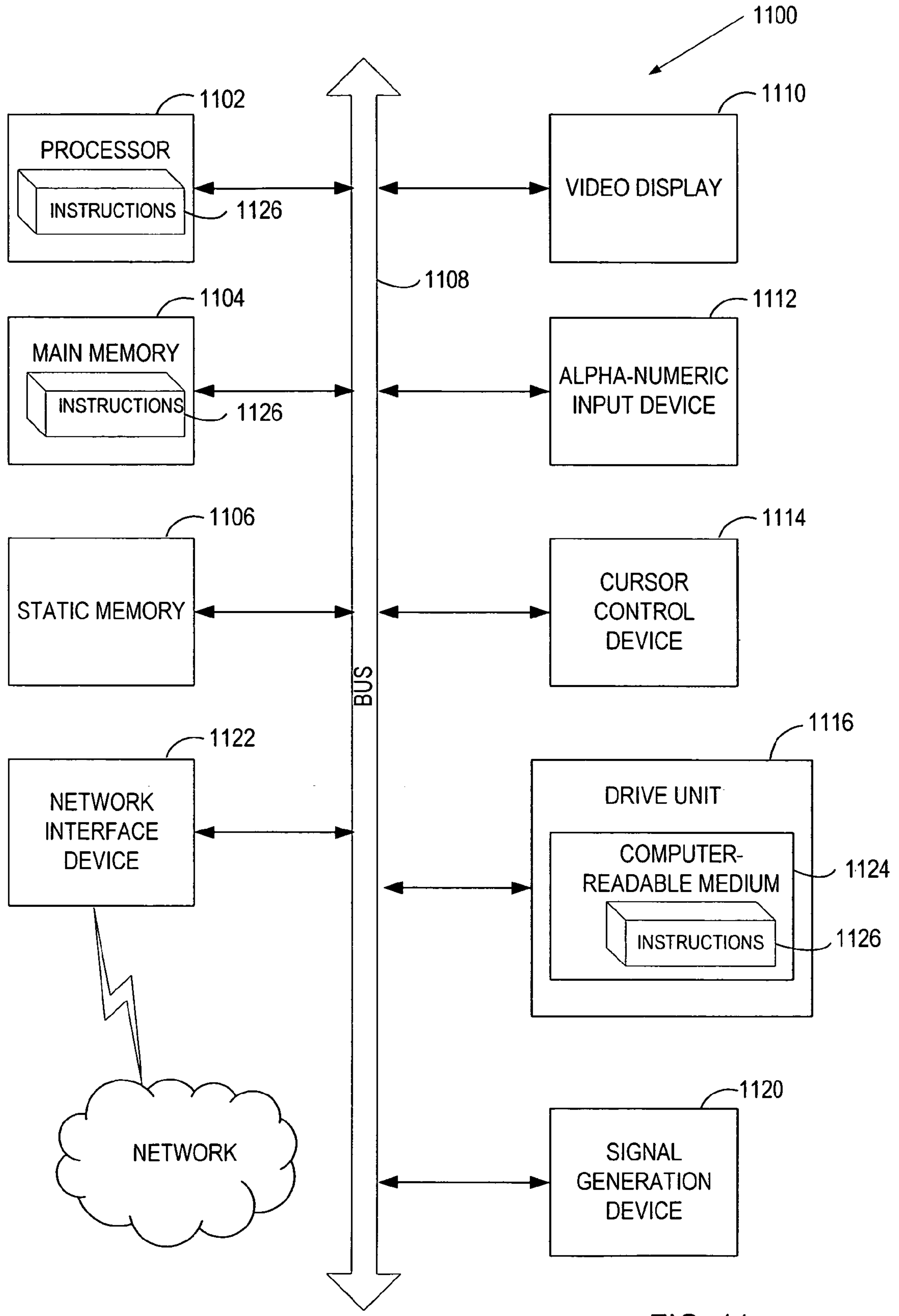

1100
1102
PROCESSOR
INSTRUCTIONS
1126
1104
MAIN MEMORY
INSTRUCTIONS
1126
1106
STATIC MEMORY
1122
NETWORK INTERFACE DEVICE
NETWORK
1108
BUS
1110
VIDEO DISPLAY
1112
ALPHA-NUMERIC INPUT DEVICE
1114
CURSOR CONTROL DEVICE
1116
DRIVE UNIT
COMPUTER-READABLE MEDIUM
1124
INSTRUCTIONS
1126
1120
SIGNAL GENERATION DEVICE

FIG. 11

METHOD AND APPARATUS FOR FILTERING EMAIL SPAM BASED ON SIMILARITY MEASURES

RELATED APPLICATIONS

[0001] The present application claims priority to U.S. Provisional Application Ser. No. 60/471,242, filed May 15, 2003, which is incorporated herein in its entirety.

FIELD OF THE INVENTION

[0002] The present invention relates to filtering electronic mail (email); more particularly, the present invention relates to filtering email spam based on similarity measures.

BACKGROUND OF THE INVENTION

[0003] The Internet is growing in popularity, and more and more people are conducting business over the Internet, advertising their products and services by generating and sending electronic mass mailings. These electronic messages (emails) are usually unsolicited and regarded as nuisances by the recipients because they occupy much of the storage space needed for the necessary and important data processing. For example, a mail server may have to reject accepting an important and/or desired email when its storage capacity is filled to the maximum with the unwanted emails containing advertisements. Moreover, thin client systems such as set top boxes, PDA's, network computers, and pagers all have limited storage capacity. Unwanted emails in any one of such systems can tie up a finite resource for the user. In addition, a typical user wastes time by downloading voluminous but useless advertisement information. These unwanted emails are commonly referred to as spam.

[0004] Presently, there are products that are capable of filtering out unwanted messages. For example, a spam block method exists which keeps an index list of all spam agents (i.e., companies that generate mass unsolicited e-mails), and provides means to block any e-mail sent from a company on the list.

[0005] Another "junk mail" filter currently available employs filters which are based on predefined words and patterns as mentioned above. An incoming mail is designated as an unwanted mail, if the subject contains a known spam pattern.

[0006] However, as spam filtering grows in sophistication, so do the techniques of spammers in avoiding the filters. Examples of tactics incorporated by recent generation of spammers include randomization, origin concealment, and filter evasion using HTML.

SUMMARY OF THE INVENTION

[0007] A method and system for filtering email spam based on similarity measures are described. According to one aspect, the method includes receiving an incoming email message, generating data characterizing the incoming email message based on the content of the incoming email message, and comparing the generated data with a set of data characterizing spam messages. The method further includes determining whether a resemblance between the data characterizing the incoming email message and any data item from the set of data characterizing spam messages exceeds a threshold.

[0008] Other features of the present invention will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0010] **FIG. 1** is a block diagram of one embodiment of a system for controlling delivery of spam electronic mail.

[0011] **FIG. 2** is a block diagram of one embodiment of a spam content preparation module.

[0012] **FIG. 3** is a block diagram of one embodiment of a similarity determination module.

[0013] **FIG. 4** is a flow diagram of one embodiment of a process for handling a spam message.

[0014] **FIG. 5** is a flow diagram of one embodiment of a process for filtering email spam based on similarities measures.

[0015] **FIG. 6A** is a flow diagram of one embodiment of a process for creating a signature of an email message.

[0016] **FIG. 6B** is a flow diagram of one embodiment of a process for detecting spam using a signature of an email message.

[0017] **FIG. 7** is a flow diagram of one embodiment of a process for a character-based comparison of documents.

[0018] **FIG. 8** is a flow diagram of one embodiment of a process for determining whether two documents are similar.

[0019] **FIG. 9** is a flow diagram of one embodiment of a process for reducing noise in an email message.

[0020] **FIG. 10** is a flow diagram of one embodiment of a process for modifying an email message to reduce noise.

[0021] **FIG. 11** is a block diagram of an exemplary computer system.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

[0022] A method and apparatus for filtering email spam based on similarity measures are described. In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0023] Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0024] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0025] The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0026] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0027] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

[0028] Filtering Email Spam Based on Similarity Measures

[0029] **FIG. 1** is a block diagram of one embodiment of a system for controlling delivery of spam electronic mail (email). The system includes a control center **102** coupled to a communications network **100** such as a public network (e.g., the Internet, a wireless network, etc.) or a private network (e.g., LAN, Intranet, etc.). The control center **102** communicates with multiple network servers **104** via the network **100**. Each server **104** communicates with user terminals **106** using a private or public network.

[0030] The control center **102** is an anti-spam facility that is responsible for analyzing messages identified as spam, developing filtering rules for detecting spam, and distributing the filtering rules to the servers **104**. A message may be identified as spam because it was sent by a known spam source (as determined, for example, using a "spam probe", i.e., an email address specifically selected to make its way into as many spammer mailing lists as possible).

[0031] A server **104** may be a mail server that receives and stores messages addressed to users of corresponding user terminals sent. Alternatively, a server **104** may be a different server coupled to the mail server **104**. Servers **104** are responsible for filtering incoming messages based on the filtering rules received from the control center **102**.

[0032] In one embodiment, the control center **102** includes a spam content preparation module **108** that is responsible for generating data characterizing the content associated with a spam attack and sending this data to the servers **104**. Each server **104** includes a similarity determination module **110** that is responsible for storing spam data received from the control center **102** and identifying incoming email messages resembling the spam content using the stored data.

[0033] In an alternative embodiment, each server **104** hosts both the spam content preparation module **108** that generates data characterizing the content associated with a spam attack and the similarity determination module **110** that uses the generated data to identify email messages resembling the spam content.

[0034] **FIG. 2** is a block diagram of one embodiment of a spam content preparation module **200**. The spam content preparation module **200** includes a spam content parser **202**, a spam data generator **206**, and a spam data transmitter **208**.

[0035] The spam content parser **202** is responsible for parsing the body of email messages resulting from spam attacks (referred to as spam messages).

[0036] The spam data generator **206** is responsible for generating data characterizing a spam message. In one embodiment, data characterizing a spam message includes a list of hash values calculated for sets of tokens (e.g., characters, words, lines, etc.) composing the spam message. Data characterizing a spam message or any other email message is referred to herein as a message signature. Signatures of spam messages or any other email messages may contain various data identifying the message content and may be created using various algorithms that enable the use of similarity measures in comparing signatures of different email messages.

[0037] In one embodiment, the spam content preparation module **200** also includes a noise reduction algorithm **204** that is responsible for detecting data indicative of noise and removing the noise from spam messages prior to generating signatures of spam messages. Noise represents data invisible to a recipient that was added to a spam message to hide its spam nature.

[0038] In one embodiment, the spam content preparation module 200 also includes a message grouping algorithm (not shown) that is responsible for grouping messages originated from a single spam attack. Grouping may be performed based on specified characteristics of spam messages (e.g., included URLs, message parts, etc.). If grouping is used, the spam data generator 206 may generate a signature for a group of spam messages rather than for each individual message.

[0039] The spam data transmitter 208 is responsible for distributing signatures of spam messages to participating servers such as servers 104 of FIG. 1. In one embodiment, each server 104 periodically (e.g., each 5 minutes) initiates a connection (e.g., a secure HTTPS connection) with the call center 102. Using this pull-based connection, signatures are transmitted from the call center 102 to the relevant server 106.

[0040] FIG. 3 is a block diagram of one embodiment of a similarity determination module 300. The similarity determination module 300 includes an incoming message parser 302, a spam data receiver 306, a message data generator 310, a resemblance identifier 312, and a spam database 304.

[0041] The incoming message parser 302 is responsible for parsing the body of incoming email messages.

[0042] The spam data receiver 306 is responsible for receiving signatures of spam messages and storing them in the spam database 304.

[0043] The message data generator 310 is responsible for generating signatures of incoming email messages. In some embodiments, a signature of an incoming email message includes a list of hash values calculated for sets of tokens (e.g., characters, words, lines, etc.) composing the incoming email message. In other embodiments, a signature of an incoming email message includes various other data characterizing the content of the email message (e.g., a subset of token sets composing the incoming email message). As discussed above, signatures of email messages may be created using various algorithms that allow for use of similarity measures in comparing signatures of different email messages.

[0044] In one embodiment, the similarity determination module 300 also includes an incoming message cleaning algorithm 308 that is responsible for detecting data indicative of noise and removing the noise from the incoming email messages prior to generating their signatures, as will be discussed in more detail below.

[0045] The resemblance identifier 312 is responsible for comparing the signature of each incoming email message with the signatures of spam messages stored in the spam database 304 and determining, based on this comparison, whether an incoming email message is similar to any spam message.

[0046] In one embodiment, the spam database 304 stores signatures generated for spam messages before they undergo the noise reduction process (i.e., noisy spam messages) and signatures generated for these spam messages after they undergo the noise reduction process (i.e., spam message with reduced noise). In this embodiment, the message data generator 310 first generates a signature of an incoming email message prior to noise reduction, and the resemblance identifier 312 compares this signature with the signatures of noisy spam messages. If this comparison indicates that the incoming email message is similar to one of these spam messages, then the resemblance identifier 312 marks this incoming email message as spam. Alternatively, the resemblance identifier 312 invokes the incoming message cleaning algorithm 308 to remove noise from the incoming email message. Then, the message data generator 310 generates a signature for the modified incoming message, which is then compared by the resemblance identifier 312 with the signatures of spam messages with reduced noise.

[0047] FIG. 4 is a flow diagram of one embodiment of a process 400 for handling a spam message. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, processing logic resides at a control center 102 of FIG. 1.

[0048] Referring to FIG. 4, process 400 begins with processing logic receiving a spam message (processing block 402).

[0049] At processing block 404, processing logic modifies the spam message to reduce noise. One embodiment of a noise reduction algorithm will be discussed in more detail below in conjunction with FIGS. 9 and 10.

[0050] At processing block 406, processing logic generates a signature of the spam message. In one embodiment, a signature of the spam message includes a list of hash values calculated for sets of tokens (e.g., characters, words, lines, etc.) composing the incoming email message, as will be discussed in more detail below in conjunction with FIG. 6A. In other embodiments, a signature of an incoming email message includes various other data characterizing the content of the email message.

[0051] At processing block 408, processing logic transfers the signature of the spam message to a server (e.g., a server 104 of FIG. 1), which uses the signature of the spam message to find incoming email messages resembling the spam message (block 410).

[0052] FIG. 5 is a flow diagram of one embodiment of a process 500 for filtering email spam based on similarities measures. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, processing logic resides at a server 104 of FIG. 1.

[0053] Referring to FIG. 5, process 500 begins with processing logic receiving an incoming email message (processing block 502).

[0054] At processing block 504, processing logic modifies the incoming message to reduce noise. One embodiment of a noise reduction algorithm will be discussed in more detail below in conjunction with FIGS. 9 and 10.

[0055] At processing block 506, processing logic generates a signature of the incoming message based on the content of the incoming message. In one embodiment, a signature of an incoming email message includes a list of hash values calculated for sets of tokens (e.g., characters, words, lines, etc.) composing the incoming email message, as will be discussed in more detail below in conjunction with **FIG. 6A**. In other embodiments, a signature of an incoming email message includes various other data characterizing the content of the email message.

[0056] At processing block **508**, processing compares the signature of the incoming messages with signatures of spam messages.

[0057] At processing block **510**, processing logic determines that the resemblance between the signature of the incoming message and a signature of some spam message exceeds a threshold similarity measure. One embodiment of a process for determining the resemblance between two messages is discussed in more detail below in conjunction with **FIG. 6B**.

[0058] At processing block **512**, processing logic marks the incoming email message as spam.

[0059] **FIG. 6A** is a flow diagram of one embodiment of a process **600** for creating a signature of an email message. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, processing logic resides at a server **104** of **FIG. 1**.

[0060] Referring to **FIG. 6A**, process **600** begins with processing logic dividing an email message into sets of tokens (processing block **602**). Each set of tokens may include a predefined number of sequential units from the email message. The predefined number may be equal to, or greater than, 1. A unit may represent a character, a word or a line in the email message. In one embodiment, each set of tokens is combined with the number of occurrences of this set of tokens in the email message.

[0061] At processing block **604**, processing logic calculates hash values for the sets of tokens. In one embodiment, a hash value is calculated by applying a hash function to each combination of a set of tokens and a corresponding token occurrence number.

[0062] At processing block **606**, processing logic creates a signature for the email message using the calculated hash values. In one embodiment, the signature is created by selecting a subset of calculated hash values and adding a parameter characterizing the email message to the selected subset of calculated hash values. The parameter may specify, for example, the size of the email message, the number of calculated hash values, the keyword associated with the email message, the name of an attachment file, etc.

[0063] In one embodiment, a signature for an email message is created using a character-based document comparison mechanism that will be discussed in more detail below in conjunction with **FIGS. 7 and 8**.

[0064] **FIG. 6B** is a flow diagram of one embodiment of a process **650** for detecting spam using a signature of an email message. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. In one embodiment, processing logic resides at a server **104** of **FIG. 1**.

[0065] Referring to **FIG. 6B**, process **650** compares data in a signature of an incoming email message with data in a signature of each spam message. The signature data includes a parameter characterizing the content of an email message and a subset of hash values generated for the tokens contained in the email message. The parameter may specify, for example, the size of the email message, the number of tokens in the email message, the keyword associated with the email message, the name of an attachment file, etc.

[0066] Processing logic begins with comparing a parameter in a signature of the incoming email message with a corresponding parameter in a signature of each spam message (processing block **652**).

[0067] A decision box **654**, processing logic determines whether any spam message signatures contain a parameter similar to the parameter of the incoming message signature. The similarity may be determined, for example, based on the allowed difference between the two parameters or the allowed ratio of the two parameters.

[0068] If none of the spam message signatures contain a parameter similar to the parameter of the incoming message signature, processing logic decides that the incoming email message is legitimate (i.e., it is not spam) (processing block **662**).

[0069] Alternatively, if one or more spam message signatures have a similar parameter, processing logic determines whether the signature of he first spam message has hash values similar to the hash values in the signature of the incoming email (decision box **656**). Based on the similarity threshold, the hash values may be considered similar if, for example, a certain number of them matches or the ratio of matched and unmatched hash values exceeds a specified threshold.

[0070] If the first spam message signature has hash values similar to the hash values of the incoming email signature, processing logic decides that the incoming email message is spam (processing block **670**). Otherwise, processing logic further determines if there are more spam message signatures with the similar parameter (decision box **658**). If so, processing logic determines whether the next spam message signature has hash values similar to the hash values of the incoming email signature (decision box **656**). If so, processing logic decides that the incoming email message is spam (processing block **670**). If not, processing logic returns to processing block **658**.

[0071] If processing logic determines that no other spam message signatures have the similar parameter, then it decides that the incoming mail message is not spam (processing block **662**). Character-Based Document Comparison Mechanism

[0072] **FIG. 7** is a flow diagram of one embodiment of a process **700** for a character-based comparison of documents. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[0073] Referring to **FIG. 7**, process **700** begins with processing logic pre-processing a document (processing block **702**). In one embodiment, the document is pre-processed by changing each upper case alphabetic character within the document to a lower case alphabetic character. For example, the message "I am Sam, Sam I am." may be pre-processed into an expression "i.am.sam.sam.i.am".

[0074] At processing block **704**, processing logic divides the document into tokens, with each token including a predefined number of sequential characters from the document. In one embodiment, each token is combined with its occurrence number. This combination is referred to as a labeled shingle. For example, if the predefined number of sequential characters in the token is equal to 3, the expression specified above includes the following set of labeled shingles:

[0075] i.a1

[0076] .am1

[0077] am.1

[0078] m.s1

[0079] .sa1

[0080] sam1

[0081] sm.2

[0082] m.s1

[0083] .sm2

[0084] sam2

[0085] am.3

[0086] m.i1

[0087] .i.1

[0088] i.a2

[0089] .am4

[0090] In one embodiment, the shingles are represented as a histogram.

[0091] At processing block **706**, processing logic calculates hash values for the tokens. In one embodiment, the hash values are calculated for the labeled shingles. For example, if a hashing function H(x) is applied to each labeled shingle illustrated above, the following results are produced:

[0092] H(i.a1)->458348732

[0093] H(.am1)->200404023

[0094] H(am.1)->692939349

[0095] H(m.s1)->220443033

[0096] H(.sa1)->554034022

[0097] H(8am1)->542929292

[0098] H(am.2)->629292229

[0099] H(m.s1)->702202232

[0100] H(.sa2)->322243349

[0101] H(8am2)->993923828

[0102] H(am.3)->163393269

[0103] H(m.i1)->595437753

[0104] H(.i.1)->843438583

[0105] H(i.a2)->244485639

[0106] H(.am4)->493869359

[0107] In one embodiment, processing logic then sorts the hash values as follows:

[0108] 163393269

[0109] 200604023

[0110] 220643033

[0111] 246685639

[0112] 322263369

[0113] 458368732

[0114] 493869359

[0115] 542929292

[0116] 554034022

[0117] 595637753

[0118] 629292229

[0119] 692939349

[0120] 702202232

[0121] 843438583

[0122] 993923828

[0123] At processing block **708**, processing logic selects a subset of hash values from the calculated hash values. In one embodiment, processing logic selects X smallest values from the sorted hash values and creates from them a "sketch" of the document. For example, for X=4, the sketch can be expressed as follows:

[0124] [163393269 200404023 220443033 244485639].

[0125] At processing block **710**, processing logic creates a signature of the document by adding to the sketch a parameter pertaining to the tokens of the document. In one embodiment, the parameter specifies the number of original tokens in the document. In the example above, the number of original tokens is 15. Hence, the signature of the document can be expressed as follows:

[0126] [15 163393269 200404023 220443033 244485639].

[0127] Alternatively, the parameter may specify any other characteristic of the content of the document (e.g., the size of the document, the keyword associated with the document, etc.).

[0128] **FIG. 8** is a flow diagram of one embodiment of a process **800** for determining whether two documents are similar. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[0129] Referring to **FIG. 8**, process **800** begins with processing logic comparing the token numbers specified in the signatures of documents 1 and 2, and determining whether the token number in the first signature is within the allowed range with respect to the token number from the second signature (decision box **802**). For example, the allowed range may be a difference of 1 or less or a ratio of 90 percent or higher.

[**0130**] If the token number in the first signature is outside of the allowed range with respect to the token number from the second signature, processing logic decides that documents 1 and 2 are different (processing block **808**). Otherwise, if the token number in the first signature is within the allowed range with respect to the token number from the second signature, processing logic determines whether the resemblance between hash values in signatures 1 and 2 exceeds a threshold (e.g., more than 95 percent of hash values are the same) (decision box **804**). If so, processing logic decides that the two documents are similar (processing block **806**). If not, processing logic decides that documents 1 and 2 are different (processing block **808**).

[**0131**] Email Spam Filtering Using Noise Reduction

[**0132**] **FIG. 9** is a flow diagram of one embodiment of a process **900** for reducing noise in an email message. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[**0133**] Referring to **FIG. 9**, process **900** begins with processing logic detecting in an email message data indicative of noise (processing block **902**). As discussed above, noise represents data that is invisible to a recipient of the mail message and was added to the email message to avoid spam filtering. Such data may include, for example, formatting data (e.g., HTML tags), numeric character references, character entity references, URL data of predefined categories, etc. Numeric character references specify the code position of a character in the document character set. Character entity references use symbolic names so that authors need not remember code positions. For example, the character entity reference &aring refers to the lowercase "a" character topped with a ring.

[**0134**] At processing block **904**, processing logic modifies the content of the email message to reduce the noise. In one embodiment, the content modification includes removing formatting data, translating numeric character references and charcater entity references to their ASCII equivalents, and modifying URL data.

[**0135**] At processing block **906**, processing logic compares the modified content of the email message with the content of a spam message. In one embodiment, the comparison is performed to identify an exact match. Alternatively, the comparison is performed to determine whether the two documents are similar.

[**0136**] **FIG. 10** is a flow diagram of one embodiment of a process **1000** for modifying an email message to reduce noise. The process may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[**0137**] Referring to **FIG. 10**, process **1000** begins with processing logic searching an email message for formatting data (e.g., HTML tags) (processing block **1002**).

[**0138**] At decision box **1004**, processing logic determines whether the found formatting data qualifies as an exception. Typically, HTML formatting does not add anything to the information content of a message. However, a few exceptions exist. These exceptions are the tags that contain useful information for further processing of the message (e.g., tags <BODY>, <A>, <IMG>, and <FONT>). For example, the <FONT> and <BODY> tags are needed for "white on white" text elimination, and the <A> and <IMG> tags typically contain link information that may be used for passing data to other components of the system.

[**0139**] If the formatting data does not qualify as an exception, the formatting data is extracted from the email message (processing block **1006**).

[**0140**] Next, processing logic converts each numerical character reference and character entity reference into a corresponding ASCII character (processing block **1008**).

[**0141**] In HTML, numeric character references may take two forms:

[**0142**] 1. The syntax "&#D;", where D is a decimal number, refers to the ISO 10646 decimal character number D; and

[**0143**] 2. The syntax "&#xH;" or "&#XH;", where H is a hexadecimal number, refers to the ISO 10646 hexadecimal character number H. Hexadecimal numbers in numeric character references are case-insensitive.

[**0144**] For example, randomized characters in the body may appear as a following expression:

[**0145**] Th&#101&#32&#83a&#118&#105n&#103 &#115R&#101&#103 is &#116e&#114&#119&#97 &#110&#116&#115&#32yo&#117.

[**0146**] This expression has a meaning of the phrase "The SavingsRegister wants you."

[**0147**] Some times the conversion performed at processing block **1008** may need to be repeated. For example, the string "&#38;" corresponds to the string "&" in ASCII, the string "&#35;" corresponds to the string "#" in ASCII, the string "&#51;" corresponds to 3 in ASCII, the string "#56;" corresponds to 8 in ASCII, and "#59;" corresponds to the string "; " in ASCII. Hence, the combined string "&#38;&#35;&#51;&#56;&#59;", when converted, results in the string "&#38;" that needs to be converted.

[**0148**] Accordingly, after the first conversion operation at processing block **1008**, processing logic checks whether the converted data still includes numeric character references or character entity references (decision box **1010**). If the check is positive, processing logic repeats the conversion operation at processing block **1008**. Otherwise, processing logic proceeds to processing block **1012**.

[**0149**] At processing block **1012**, processing logic modifies URL data of predefined categories. These categories may include, for example, numerical character references contained in the URL that are converted by processing logic into corresponding ASCII characters. In addition, the URL "password" syntax may be used to add characters before an "@" in the URL hostname. These characters are ignored by the target web server but they add significant amounts of noise information to each URL. Processing logic modifies the URL data by removing these additional characters. Finally, processing logic removes the "query" part of the URL, following a string "?" at the end of the URL.

[0150] An example of a URL is as follows:

[0151] http %3a %2f%2flotsofjunk@www.foo.com %2fbar.html?muchmorejunk

[0152] Processing logic modifies the above URL data into http://www.foo.com/bar.hmil.

[0153] An Exemplary Computer System

[0154] **FIG. 11** is a block diagram of an exemplary computer system **1100** that may be used to perform one or more of the operations described herein. In alternative embodiments, the machine may comprise a network router, a network switch, a network bridge, Personal Digital Assistant (PDA), a cellular telephone, a web appliance or any machine capable of executing a sequence of instructions that specify actions to be taken by that machine.

[0155] The computer system **1100** includes a processor **1102**, a main memory **1104** and a static memory **1106**, which communicate with each other via a bus **1108**. The computer system **1100** may further include a video display unit **1110** (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system **1100** also includes an alphanumeric input device **1112** (e.g., a keyboard), a cursor control device **1114** (e.g., a mouse), a disk drive unit **1116**, a signal generation device **1120** (e.g., a speaker) and a network interface device **1122**.

[0156] The disk drive unit **1116** includes a computer-readable medium **1124** on which is stored a set of instructions (i.e., software) **1126** embodying any one, or all, of the methodologies described above. The software **1126** is also shown to reside, completely or at least partially, within the main memory **1104** and/or within the processor **1102**. The software **1126** may further be transmitted or received via the network interface device **1122**. For the purposes of this specification, the term "computer-readable medium" shall be taken to include any medium that is capable of storing or encoding a sequence of instructions for execution by the computer and that cause the computer to perform any one of the methodologies of the present invention. The term "computer-readable medium" shall accordingly be taken to included, but not be limited to, solid-state memories, optical and magnetic disks, and carrier wave signals.

[0157] Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.

We claim:

1. The method comprising:

receiving an email message;

generating data characterizing the email message based on content of the email message;

comparing the data characterizing the email message with a set of data characterizing a plurality of spam messages; and

determining whether a resemblance between the data characterizing the email message and any data item within the set of data characterizing the plurality of spam messages exceeds a threshold.

2. The method of claim 1 further comprising:

marking the email message as spam if the resemblance between the data characterizing the email message and any data item within the set of data characterizing the plurality of spam messages exceeds a threshold.

3. The method of claim 1 further comprising:

receiving data characterizing a new spam message; and

storing the received data in a database.

4. The method of claim 1 further comprising:

upon receiving the email message, evaluating the email message for a presence of noise added to avoid spam filtering; and

modifying content of the email message to reduce the noise.

5. The method of claim 4 wherein evaluating the email message for the presence of noise comprises detecting at least one of formatting data, a numeric character reference, a character entity reference, and predefined URL data.

6. The method of claim 1 wherein generating the data characterizing the email message comprises:

dividing the email message into a plurality of tokens; and

calculating a plurality of hash values for the plurality of tokens.

7. The method of claim 6 wherein comparing the data characterizing the email message with the set of data characterizing the plurality of spam messages comprises:

finding, in the set of data characterizing the plurality of spam messages, one or more data items having additional information similar to additional information contained in the data characterizing the plurality of spam messages; and

comparing the subset of hash values in the data characterizing the email message with a subset of hash values in each found data item until finding a similar subset of hash values.

8. The method of claim 3 further comprising:

evaluating the new spam message for a presence of noise;

modifying content of the new spam message to reduce the noise; and

generating data characterizing the spam message based on content of the modified new spam message.

9. The method comprising:

receiving a spam message;

generating data characterizing the spam message based on content of the spam message; and

transferring the data characterizing the spam message to a server, the data characterizing the spam message being subsequently used to find incoming messages resembling the spam message.

10. The method of claim 9 further comprising:

upon receiving the spam message, evaluating the spam message for a presence of noise; and

modifying content of the spam message to reduce the noise.

11. The method of claim 9 wherein evaluating the spam message for the presence of noise comprises detecting at least one of formatting data, a numeric character reference, a character entity reference, and predefined URL data.

12. The method of claim 9 wherein generating the data characterizing the spam message comprises:

dividing the spam message into a plurality of tokens; and

calculating a plurality of hash values for the plurality of tokens.

13. A system comprising:

an incoming message parser to receive an email message;

a message data generator to generate data characterizing the email message based on content of the email message; and

a resemblance identifier to compare the data characterizing the email message with a set of data characterizing a plurality of spam messages, and to determine whether a resemblance between the data characterizing the email message and any data item within the set of data characterizing the plurality of spam messages exceeds a threshold.

14. The system of claim 13 further comprising:

a database to store data characterizing a new spam message.

15. The system of claim 13 further comprising:

a message cleaning algorithm to evaluate the email message for a presence of noise added to avoid spam filtering, and to modify content of the email message to reduce the noise.

16. A system comprising:

a spam content parser to receive a spam message;

a spam data generator to generate data characterizing the spam message based on content of the spam message; and

a spam data transmitter to transfer the data characterizing the spam message to a server, the data characterizing the spam message being subsequently used to find incoming messages resembling the spam message.

17. The system of claim 16 further comprising a noise reduction algorithm to evaluate the spam message for a presence of noise, and to modify content of the spam message to reduce the noise.

18. The system of claim 17 wherein the noise reduction algorithm is to evaluate the spam message for the presence of noise by detecting at least one of formatting data, a numeric character reference, a character entity reference, and predefined URL data.

19. An apparatus comprising:

means for receiving an email message;

means for generating data characterizing the email message based on content of the email message;

means for comparing the data characterizing the email message with a set of data characterizing a plurality of spam messages; and

means for determining whether a resemblance between the data characterizing the email message and any data item within the set of data characterizing the plurality of spam messages exceeds a threshold.

20. The apparatus of claim 19 further comprising:

means for receiving data characterizing a new spam message; and

a database to storing the received data.

21. An apparatus comprising:

means for receiving a spam message;

means for generating data characterizing the spam message based on content of the spam message; and

means for transferring the data characterizing the spam message to a server, the data characterizing the spam message being subsequently used to find incoming messages resembling the spam message.

22. The apparatus of claim 21 further comprising:

means for evaluating the spam message for a presence of noise; and

means for modifying content of the spam message to reduce the noise.

23. The apparatus of claim 21 wherein means for evaluating the spam message for the presence of noise comprises means for detecting at least one of formatting data, a numeric character reference, a character entity reference, and predefined URL data.

24. A computer readable medium comprising executable instructions which when executed on a processing system cause said processing system to perform a method comprising:

receiving an email message;

generating data characterizing the email message based on content of the email message;

comparing the data characterizing the email message with a set of data characterizing a plurality of spam messages; and

determining whether a resemblance between the data characterizing the email message and any data item within the set of data characterizing the plurality of spam messages exceeds a threshold.

25. The computer readable medium of claim 24 wherein the method further comprises:

receiving data characterizing a new spam message; and

storing the received data in a database.

26. The computer readable medium of claim 24 wherein the method further comprises:

upon receiving the email message, evaluating the email message for a presence of noise added to avoid spam filtering; and

modifying content of the email message to reduce the noise.

27. A computer readable medium comprising executable instructions which when executed on a processing system cause said processing system to perform a method comprising:

receiving a spam message;

generating data characterizing the spam message based on content of the spam message; and

transferring the data characterizing the spam message to a server, the data characterizing the spam message being subsequently used to find incoming messages resembling the spam message.

28. The computer readable medium of claim 27 wherein the method further comprises:

upon receiving the spam message, evaluating the spam message for a presence of noise; and

modifying content of the spam message to reduce the noise.

29. The computer readable medium of claim 27 wherein evaluating the spam message for the presence of noise comprises detecting at least one of formatting data, a numeric character reference, a character entity reference, and predefined URL data.

* * * * *