December 12, 2017

Detecting Spam Content By Clustering Media Channels Having Similar Thumbnail Images

Bing Jian

Charles Curry

DETECTING SPAM CONTENT BY CLUSTERING MEDIA CHANNELS HAVING SIMILAR THUMBNAIL IMAGES

ABSTRACT

Disclosed herein is a mechanism for detecting spam content by clustering media channels having similar thumbnail images. In some instances, the mechanism can construct an index of video thumbnail images. For example, the index of thumbnail images can be a unique set of the most frequently used thumbnail images from the thumbnail images associated with each video in a corpus of videos. The mechanism can use the index of thumbnail images to cluster videos having thumbnail images that are the same or substantially similar to those in the index of thumbnail images. The mechanism can then, based on the video clusters, determine clusters of media channels that share the same or substantially similar thumbnail images to those in the index of thumbnail images. Upon determining clusters of media channels, the mechanism can perform further analysis, such as determining whether the content associated with a cluster of media channels contains spam content or determining whether the content associated with a cluster of media channels indicates scaled abuse at the user account level.

BACKGROUND

Content sharing platforms can receive uploaded video content, store the uploaded video content, and then provide the uploaded video content to many users, for example, by streaming the video content to multiple user devices. These content sharing platforms can, in some instances, include one or more media channels. A media channel is a mechanism for providing certain media content items and/or for providing access to media content items to users for consumption.

A user associated with a media channel can provide a custom video thumbnail that

represents the content depicted within a particular media content item that is provided by the media channel. In some instances, the same image or same set of images are repeatedly used by multiple media channels as custom video thumbnails. For example, a cluster of multiple media channels on a content sharing platform can use the same set of sexually suggestive images as custom thumbnails on their associated videos. These clusters of media channels using the same set of custom thumbnail images are likely to be associated with the same user. Additionally, these clusters of media channels using the same set of custom thumbnail images are likely the result of a scaled abuse event, where, for example, a massive hijacking of a large number of user accounts has occurred.

Thus, there is a need for a better approach to detecting spam content by clustering media channels having similar thumbnail images.

DESCRIPTION

A mechanism for detecting spam content by clustering media channels having similar thumbnail images is provided. In some instances, the mechanism can construct an index of video thumbnail images. For example, the index of thumbnail images can be a unique set of the most frequently used thumbnail images from the thumbnail images associated with each video in a corpus of videos. The mechanism can use the index of thumbnail images to cluster videos having thumbnail images that are the same or substantially similar to those in the index of thumbnail images. The mechanism can then, based on the video clusters, determine clusters of media channels that share the same or substantially similar thumbnail images to those in the index of thumbnail images. Upon determining clusters of media channels, the mechanism can perform further analysis, such as determining whether the content associated with a cluster of

media channels contains spam content or determining whether the content associated with a cluster of media channels indicates scaled abuse at the user account level.

In some implementations, the mechanism can begin by constructing an index of video thumbnail images. For example, the index can include a selected subset of video thumbnail images from the video thumbnail images associated with each video stored on a video server.

FIG. 1 shows an illustrative example of a process for constructing an index of thumbnail images from a corpus of thumbnail images. In some instances, the steps of the process shown in FIG. 1 can be implemented on a server, such as a server that manages uploaded media content items and provides media content items to user devices.

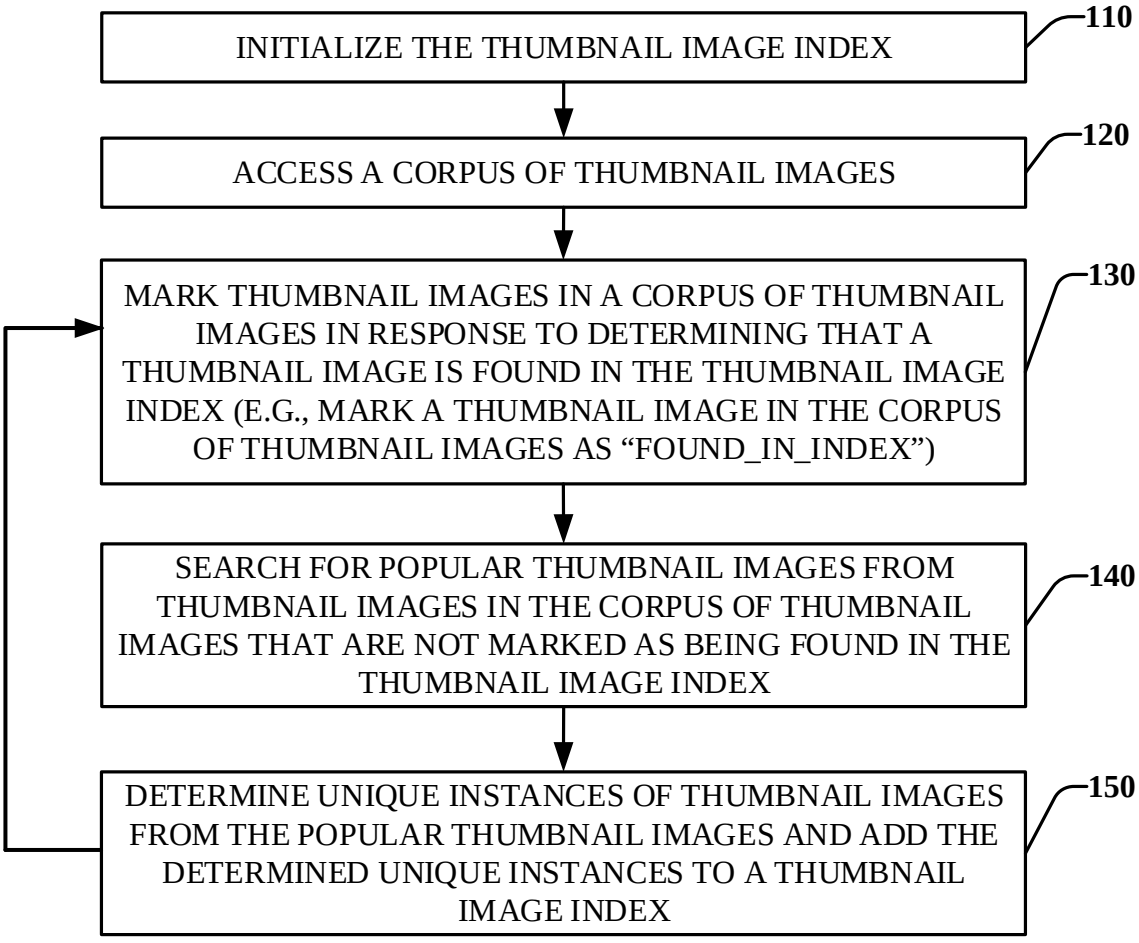

FIG. 1

Turning to FIG. 1, at step 110, the server can initialize the thumbnail image index. For example, initialization can include starting the thumbnail image index from an empty set. In another example, the server can initialize the thumbnail image index by loading a pre-built thumbnail image index containing thumbnail images associated with particular videos.

At step 120, the server can access a corpus of thumbnail images, where each thumbnail image is assigned to a video associated with a content sharing platform.

At step 130, the server can determine whether a thumbnail image in a corpus of thumbnail images is found in the thumbnail image index and, in response to determining that a thumbnail image in the corpus of thumbnail images is found in the thumbnail image index, marking or otherwise designating that thumbnail image as being found in the thumbnail image index. For example, in response to determining that a thumbnail image in the corpus of thumbnail images is found in the thumbnail image index, the server can mark the thumbnail image in the corpus of thumbnail images as "found_in_index."

In some instances, the server can reduce the number of thumbnail images for analysis by selecting a sample or subset of thumbnail images from the corpus of thumbnail images in response to determining that the size of the corpus of thumbnail images is greater than a particular threshold value (e.g., one million thumbnail images). For example, the pairwise comparison described in step 140 below can be computationally expensive if the size of the corpus of thumbnail images is very large. Accordingly, in some instances, the corpus of thumbnail images can be replaced with a sample of thumbnail images selected from the corpus of thumbnail images.

It should be noted that the server can select a sample of thumbnail images from the corpus of thumbnail images using any suitable approach.

For example, a sample of thumbnail images can be obtained by applying a classifier to the thumbnail images in the corpus of thumbnail images to select particular types of thumbnail images. In a more particular example, a classifier that has been trained to detect thumbnail images that are likely to be deemed as racy or sexually suggestive images can be applied to each thumbnail image in a corpus of thumbnail images. It should be noted that any suitable classifier can be applied to filter thumbnail images from the corpus of thumbnail images, thereby reducing the number of thumbnail images analyzed for potential addition to the thumbnail image index. It should also be noted that a particular classifier, such as a racy classifier that has been trained to detect thumbnail images that are likely to be deemed as racy or sexually suggestive images, can be selected for application to the corpus of thumbnail images to construct a thumbnail image index of racy or otherwise inappropriate thumbnail images.

In another example, a sample of thumbnail images can be obtained by selecting thumbnail images from the corpus of thumbnail images that are associated with recently uploaded videos. In a more particular example, the server can select a sample of thumbnail images can be selected from the corpus of thumbnail images for videos that have been uploaded after a particular time (e.g., only thumbnail images associated with videos that have been uploaded within the last month). It should be noted that filtering for thumbnail images associated with recently uploaded videos can be applied to the corpus of thumbnail images to construct a thumbnail image index of trending thumbnail images.

In yet another example, a sample of thumbnail images can be obtained by randomly selecting thumbnail images from the corpus of thumbnail images for further analysis.

It should be noted that any suitable criteria can be used to select a thumbnail image from the corpus of thumbnail images (e.g., popularity, recency, image category, etc.). It should also

be noted that any suitable combination of filters can be applied to select a thumbnail image from the corpus of thumbnail images.

Alternatively, in response to determining that the size of the corpus of thumbnail images is less than a particular threshold value, the server can maintain the corpus of thumbnail images and proceed to step 130.

At step 140, the server can search for popular thumbnail images from those images in the corpus of thumbnail images that have not been marked as being found in the thumbnail image index. For example, the server can skip the analysis of thumbnail images in the corpus of thumbnail images that have been marked as "found_in_index" and determine which thumbnail images are deemed popular thumbnail images from the remaining thumbnail images in the corpus. In a more particular example, the server can perform a pairwise comparison of the thumbnail images in the corpus of thumbnail images that are not marked "found_in_index" to determine popular images for potential addition to the thumbnail image index. It should be noted that popular images can include those images which are used as a thumbnail image for at least a given number of videos (K). In another more particular example, the server can apply any suitable approach, such as similarity clustering, to determine whether two thumbnail images in the corpus of thumbnail images (that are not marked "found_in_index") are the same or substantially similar to one another.

It should be noted that, in some instances, the threshold value (K) for determining whether a thumbnail image is a popular thumbnail image can be modified. For example, for each iteration of steps 130 through 150, the threshold value (K) can be decreased at each iteration – e.g., where a thumbnail image is deemed a popular thumbnail image in response to being associated with an image cluster having a size greater than 100 in a first iteration and

where a thumbnail image is deemed a popular thumbnail image in response to being associated with an image cluster having a size greater than 50 in a subsequent iteration.

At step 150, the server can determine unique instances of thumbnail images from the popular thumbnail images identified at step 140 and add the determined unique instances of thumbnail images from the popular thumbnail images to the thumbnail image index. For example, as mentioned above, the server can determine popular thumbnail image clusters from the corpus of thumbnail images and can select one thumbnail image from each image cluster for addition to the thumbnail image index.

An illustrative example of a first iteration of steps 110 through 150 of FIG. 1 is shown in FIG. 2 below.

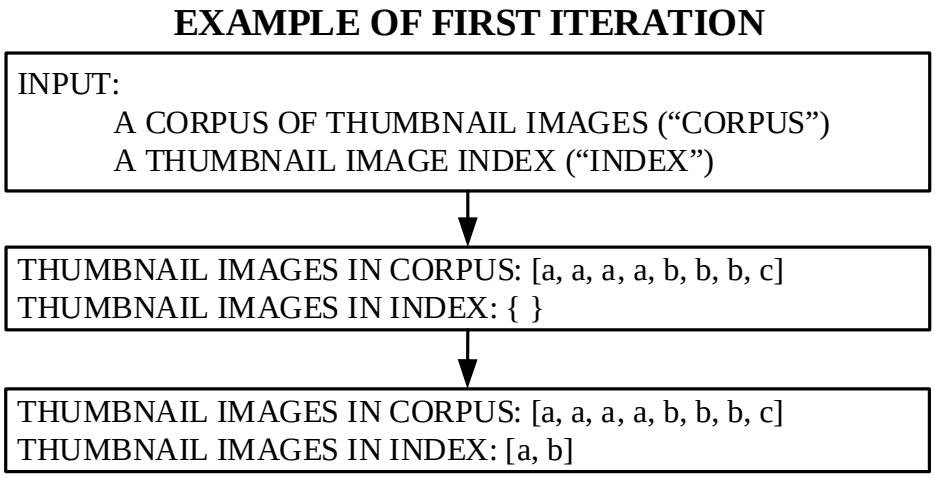

FIG. 2

As shown in FIG. 2, an exemplary corpus of thumbnail images can include thumbnail images [a, a, a, a, b, b, b, c] and the thumbnail image index can be initialized to be an empty set. Accordingly, in this example, the server can skip the marking or designation of step 130 as there will be no thumbnail images in the corpus that are also contained in the thumbnail image index. Upon performing a pairwise comparison of the thumbnail images in the corpus of thumbnail

images to determine popular images and determining unique instances of the popular images, the server can determine that images a and b are deemed popular thumbnail images in the corpus and unique instances of images a and b are stored in the thumbnail image index.

An illustrative example of a second iteration of steps 110 through 150 of FIG. 1 is shown in FIG. 3 below.

EXAMPLE OF SECOND ITERATION

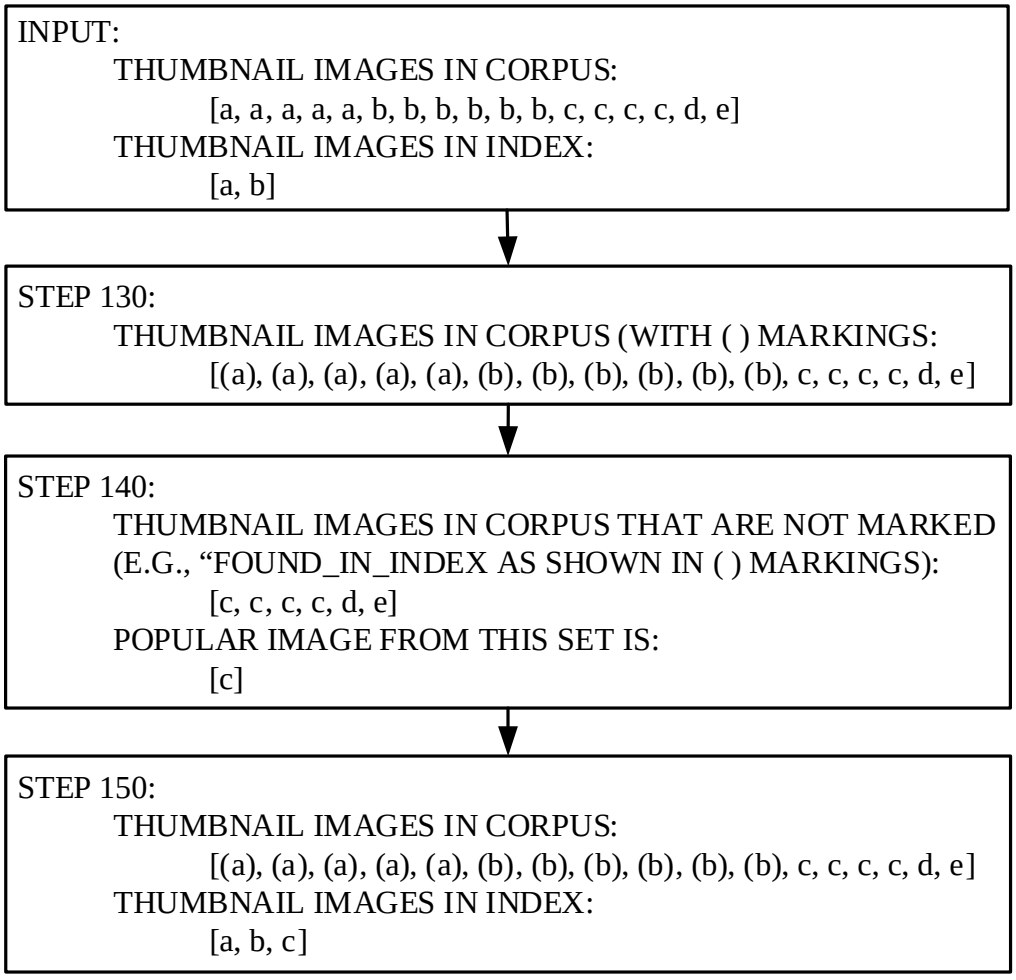

FIG. 3

As shown in FIG. 3, the corpus of thumbnail images includes thumbnail images [a, a, a, a, a, b, b, b, b, b, b, c, c, c, c, d, e] and the thumbnail image index includes thumbnail images [a, b]. It should be noted that the corpus of thumbnail images can continue to grow as videos and

other media content items are uploaded to the content sharing platform and the thumbnail image index can continue to grow through iterations of steps 130-150 of FIG. 1. In the second iteration, the server can access the corpus of thumbnail images and compare the thumbnail images in the thumbnail image index with the corpus of thumbnail images. Those thumbnail images in the corpus of thumbnail images that are found in the thumbnail image index can be marked (e.g., marked as "found_in_index"). For example, as shown in FIG. 3 above, thumbnail images in the corpus of thumbnail images that are found in the thumbnail image index have been designated with a "( )" mark. The server can continue by determining which thumbnail images in the corpus of thumbnail images are not marked as being found in the thumbnail image index (e.g., not marked "found_in_index" or with the ( ) in FIG. 3) and determining popular images from the remaining images. As shown in FIG. 3, upon performing a pairwise comparison of the thumbnail images in the corpus of thumbnail images that have not been marked as being found in the thumbnail image index to determine popular images and determining unique instances of the popular images, the server can determine that image c is deemed a popular thumbnail image in the corpus and a unique instance of image c is stored in the thumbnail image index.

An illustrative example of a third iteration of steps 110 through 150 of FIG. 1 is shown in FIG. 4 below.

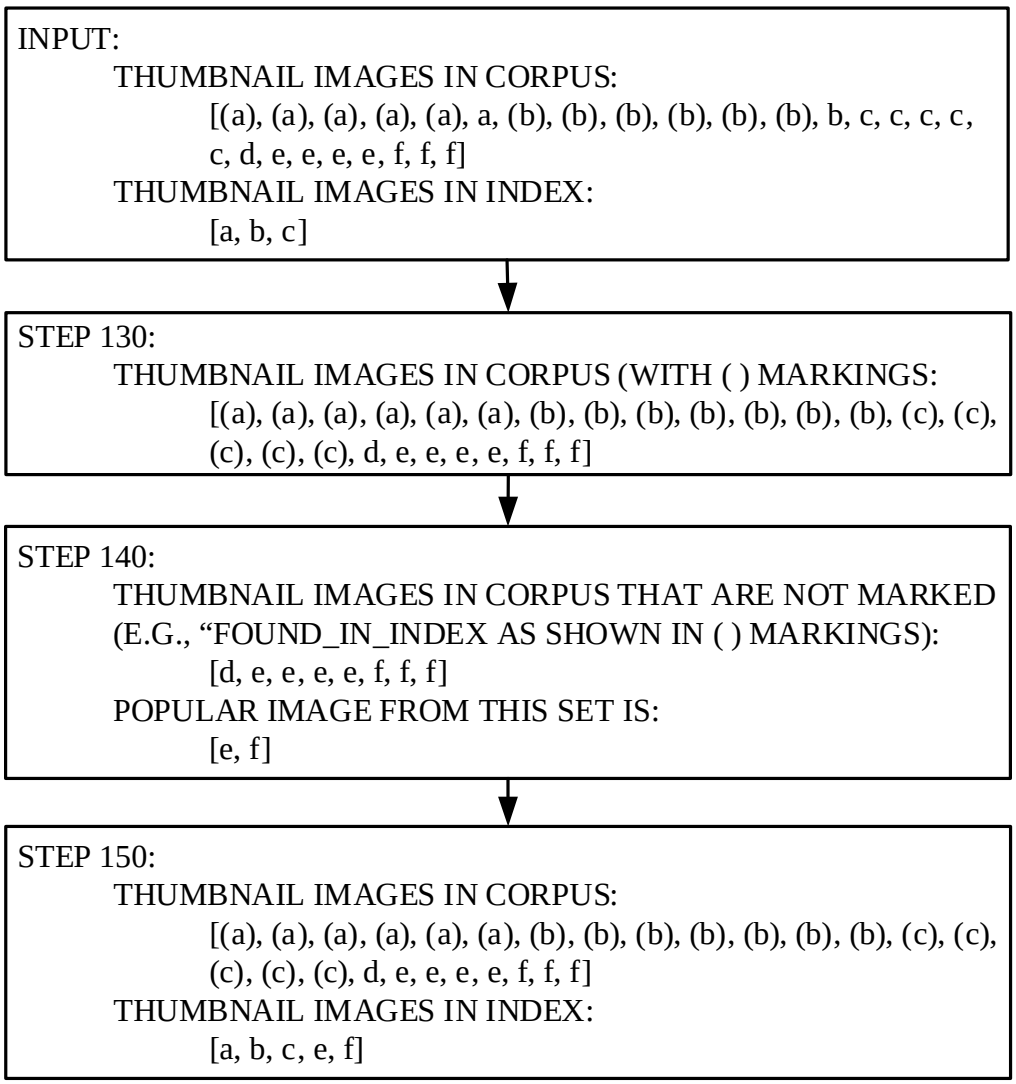

FIG. 4

As shown in FIG. 4, the corpus of thumbnail images includes thumbnail images [(a), (a), (a), (a), (a), a, (b), (b), (b), (b), (b), (b), b, c, c, c, c, c, d, e, e, e, e, f, f, f] and the thumbnail image index includes thumbnail images [a, b, c]. Note again that the corpus of thumbnail images can continue to grow as videos and other media content items are uploaded to the content sharing platform and the thumbnail image index can continue to grow through iterations of steps 130-150 of FIG. 1. In the third iteration, the server can access the corpus of thumbnail images and compare the thumbnail images in the thumbnail image index with the corpus of thumbnail

images. Again, those thumbnail images in the corpus of thumbnail images that are found in the thumbnail image index can be marked in the corpus of thumbnail images (e.g., marked as "found_in_index"). For example, as shown in FIG. 4 above, thumbnail images associated with newly uploaded videos that are found in the thumbnail image index have been marked in the corpus of thumbnail images – e.g., thumbnail images a, b, c, c, c, c, and c. The server can continue by determining which thumbnail images in the corpus of thumbnail images are not marked as being found in the thumbnail image index (e.g., not marked "found_in_index" or with the ( ) in FIG. 4) and determining popular images from the remaining images. As shown in FIG. 4, upon performing a pairwise comparison of the thumbnail images in the corpus of thumbnail images that have not been marked as being found in the thumbnail image index to determine popular images and determining unique instances of the popular images, the server can determine that images e and f are deemed popular thumbnail images in the corpus and unique instances of images e and f are stored in the thumbnail image index.

As illustrated in FIGS. 2-4 above, the server can repeat steps 130 through 150 of FIG. 1 to continue to add new thumbnail images to the thumbnail image index and continue to mark those thumbnail images that have been added to the thumbnail image index in the corpus of thumbnail image (e.g., "found_in_index"). For example, the server can continue repeat steps 130 through 150 of FIG. 1 until no additional thumbnail images are added to the thumbnail image index. In another example, as described above, the server can continue to decrease the value of K number of videos in the definition of popular thumbnail images. In a more particular example, in a first iteration of steps 130 through 150 of FIG. 1, the server can define popular thumbnail images as those images which are used as a thumbnail image for at least 100 videos, thereby marking image clusters of size greater than 100 from the corpus as being found in the thumbnail

image index and, for each cluster, a representative thumbnail image is selected and added to the thumbnail image index. In continuing this example, the subsequent iterations of steps 130 through 150 of FIG. 1, the server can continue to decrease the value K number of videos in the definition of popular thumbnail images, thereby continuing to substantially remove large image clusters from analysis by marking these image clusters in the corpus as being found in the thumbnail image index.

It should be noted that, at the end of each iteration of steps 130 through 150 of FIG. 1, a thumbnail image index can be constructed that contains unique instances of popular or frequently used thumbnail images. For example, duplicate or near-duplicate thumbnail images should not be included in the thumbnail image index. Accordingly, the server can construct a thumbnail image index containing a unique set of thumbnail images that covers the most frequently used thumbnail images by repeatedly analyzing thumbnail images associated with an ever-growing set of thumbnail images in a corpus of thumbnail images.

Upon obtaining the thumbnail image index, the server can determine clusters of media channels that share the same or substantially similar thumbnail images to those in the thumbnail image index.

FIG. 5 shows an illustrative example of a process for detecting spam content by clustering media channels having the same or similar thumbnail images to those in the thumbnail image index.

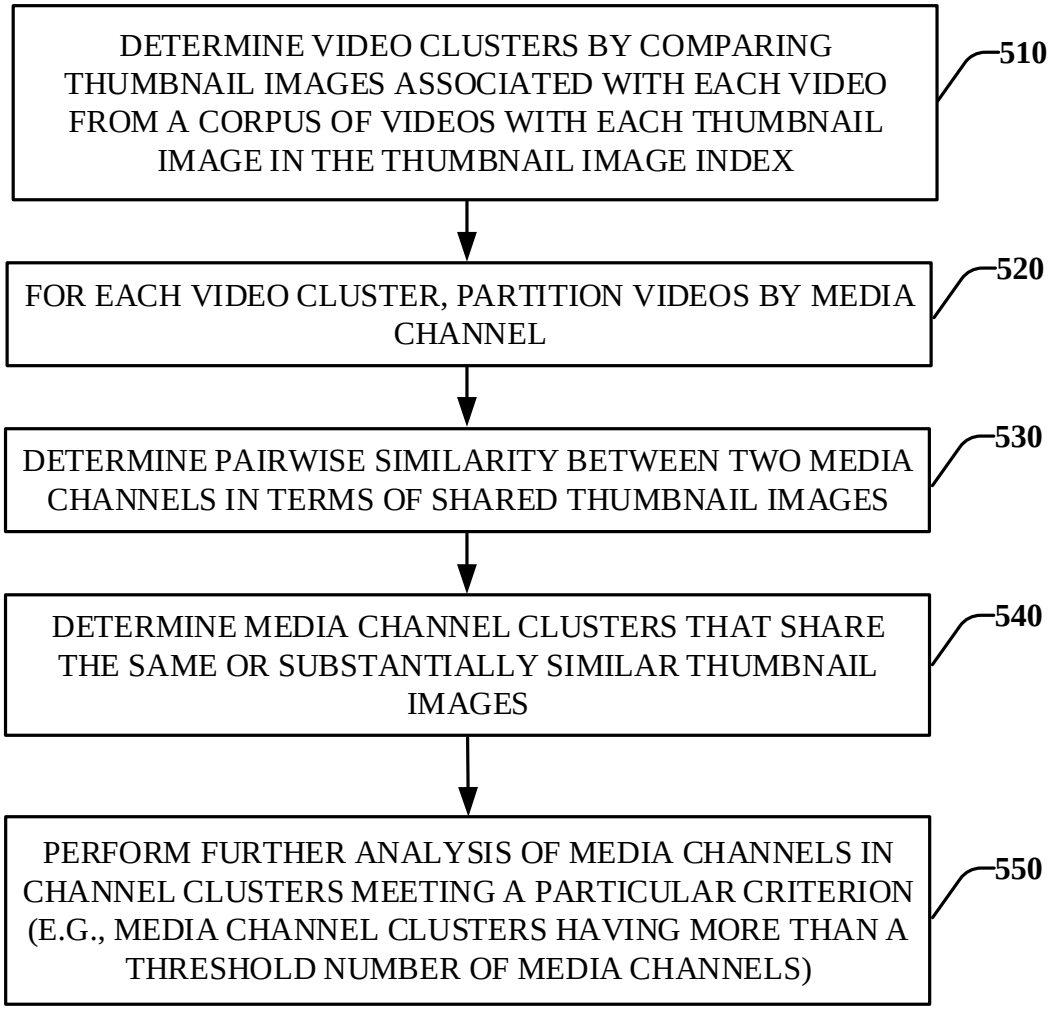

FIG. 5

Turning to FIG. 5, at step 510, the server can compare thumbnail images from a corpus of videos against the thumbnail image index to determine whether there is a match. A match can be determined if two images are either the same or substantially similar (e.g., nearly duplicate). For example, the server can use the thumbnail image index to determine clusters of videos, where in each cluster, videos have thumbnail images that are the same or substantially similar to one of the images in the thumbnail image index.

At step 520, within each video cluster, the server can further partition videos by media channel. For example, a cluster may have N number of video that each use a particular

thumbnail image that is the same or substantially similar to a thumbnail image in the thumbnail image index, where these N videos may be associated with M different media channels.

At step 530, with these clusters, the server can measure the pairwise similarity in terms of shared thumbnails between two channels. For example, the server can define a vector space model, where the "documents" are media channels and the "terms" are the reused thumbnail images that are the same or substantially similar to those in the thumbnail image index.

At step 540, upon computing the pairwise similarity between two channels, the server can cluster media channels that share a particular number of the same or substantially similar thumbnail images to those in the thumbnail image index.

At step 550, based on clusters of media channels that are greater than a particular number of channels, the server can determine whether to perform further analysis on these media channels. For example, in some instances, the server can review the content associated with the media channels in a particular cluster and determine whether the media channels contain spam content. In another example, in some instances, the server can review the content associated with the media channels in a particular cluster and determine whether there are indications of scaled abuse at the user account level.

Accordingly, a mechanism for detecting spam content by clustering media channels having similar thumbnail images is provided.