Technical Disclosure Commons

Defensive Publications Series

January 27, 2016

PERSONALIZED CONTENT RECOMMENDATIONS

Wei Zhang

Xiaohong Gong

PERSONALIZED CONTENT RECOMMENDATIONS

ABSTRACT

A recommendation system uses a graph mining based approach to cluster web pages with similar contents and provide related content recommendations. The system receives a request from a user to access a first web page. The system then retrieves a list of secondary web pages that are associated with the first web page in an entity graph. The system calculates a similarity score between a secondary web page and the first web page. The similarity score between the first and a secondary web pages is calculated based on overlapping entities between the first web page and the secondary web page. The system then ranks the secondary web pages based on ascending similarity scores. Further, the system provides the list of secondary web pages, based on their similarity scores, to the user as an recommendation.

PROBLEM STATEMENT

Nowadays, a key challenge for publishers is to help users find interesting contents to read so that publishers can entice users to stay on their web sites for longer durations. Many publishers' web sites have a section called "recommended for you" or "more like this," in which a list of uniform resource locators (urls) are recommended to an user. However, the current approaches for recommending content to users are only weakly personalized. For instance, some web sites simply provide a list of most popular urls, which takes an approach that is not tailored to the user. Some web sites group urls together based on their metadata and provide a list of urls that contain the same metadata as the url that the user is currently viewing. This approach is

slightly more personal than listing trending web sites, but there are further opportunities to customize content recommendations.

DETAILED DESCRIPTION

The systems and techniques described in this disclosure relate to a personalized content recommendation system that uses a graph mining-based web page clustering-and-recommendation approach. The system can be implemented for use in an Internet, an intranet, or another client and server environment. The client device can be any electronic device such as a desktop computer, a laptop computer, a set-top box, a mobile device, a smartphone, a tablet, a handheld electronic device, a wearable device, etc.

Fig. 1 illustrates an example method 100 that uses graph mining based approach to cluster web pages with similar contents and to provide related content recommendations. The graph may be an entity graph storing nodes and edges for a variety of topics. The method 100 can be performed by the recommendation system. The system receives 110 a request to access a first web page. A user may generate a request to access the first web page on a web browser or an application. The user may generate the request from the user's electronic device, e.g., mobile phone, tablet, computer, laptop, or wearable device. The requested first web page is linked to a uniform resource locator (url). The system fetches the contents of the url from a server and displays or otherwise annunciates it to the user at the user's electronic device.

The system analyzes the content of the first web page and maps the contents to a list of entities in an entity graph. The entities in the entity graph are interconnected through structured relationships and are uniquely identified by their machine identifiers (MID). By representing information as a graph of linkages between unique identifiers, the system creates a knowledge

representation that is accumulative and accretive. The system uses the entities and their topicality scores as a concise and accurate representation of the website's contents. Further, the system uses graph mining techniques to build a url graph where each node is a unique url and each edge represents the similarity between two urls.

The system then retrieves 120 a list of secondary web pages and respective scores for each of the secondary web pages based on associations stored in the entity graph. The system calculates a similarity score, i.e. the edge weight between two urls. The similarity score between the first and a secondary web page is calculated based on their overlapping entities and the entities' topicality-scores. After the user requests the first web page, the system queries the url graph for related urls and their edge weights. The system obtains the related urls and ranks them by ascending edge weights. The higher the edge weights, the more closely the contents are related.

The system then provides 130 the list of secondary web pages based on their respective scores. The system uses one or more url filtering mechanisms: one is to filter out 133 urls that have contents that are overly similar, the second one is to filter out 135 urls that have too many related urls, the third one is to use an url's click through rate as a metric to filter out 137 unpopular urls, and the fourth one is to filter out 139 urls with only one entity. A secondary web page url with contents that are overly similar indicate that the secondary web page is a full or partial derivative of the first web page, thus filtering out 133 that secondary web page reduces repetition of the first web page's contents. A secondary web page url with too many related urls indicates a hub page that points to many articles, thus filtering out 135 that secondary web page improves the chances that a user will reach a related content url directly. A url with a poor

click-through rate indicates an un-enticing url, thus filtering out 137 that secondary web page allows other secondary web pages to be presented. And a url with only one entity (or a low number of entities) indicates that the webpage is not well developed, and thus filtering out 139 that secondary web page will present better-developed web pages to the user. The filtered list of secondary web pages, containing a number of top-scoring secondary web page urls, is provided as recommendations to the user on the first web page or anywhere else in the web browser.

Figs. 2-5 provide a detailed system architecture for performing the method 100 on the recommendation system.The system architecture is divided into four pipelines, 1) data processing pipeline, 2) content analyzing pipeline, 3) training pipeline, and 4) serving pipeline.

The system architecture for the data processing pipeline is illustrated in Fig. 2. The system obtains (210) urls and their associated first_seen timestamp from a data storage (205). The system then cleans (220) up the urls by canonicalizing them. Further, the system removes (230) the query terms in the urls and filters out illegal and private urls. Because the system canonicalizes urls and removes query terms in the urls, many urls may become one url after the cleaning step. The system further aggregates (230) urls that are the same and choses the earliest first_seen timestamp as the first_seen timestamp of the clean url. The system saves (250) the urls and the associated first_seen timestamp into an url storage.

The system architecture for the content analyzing pipeline is illustrated in Fig. 3. A content analyzer (310) takes an url from the url storage (element 250 from FIG. 2) as input, analyzes the content of the url, and maps the content of a web page to entities in an entity graph (305). Content filtering module filters (320) out homepages and hubpages because they are not interesting urls to recommend to users. After the entities and title information are obtained (330)

from the content, the system saves (340) the urls and their associated information in a storage for urls and associated entities.

Fig. 4(a) illustrates the system architecture for the training or graph building pipeline. Based on a url and its associated entities from the url and entities storage (element 340 from FIG. 3), the system constructs (410) a url node where a node identifier is a combination of profile_id and a unique fingerprint of the url. The entities associated with the url are selected as features, and the feature weights are the topicality scores associated with the entities. The topicality score is a measure of how closely related the url content is to the entity.

The system then filters urls (420), using at least one of three url filtering mechanisms mentioned earlier: 1) filter out urls with only one entity because a web page with reasonable contents will contain at least a few entities, i.e. things, concepts, people that exist in the real world. 2) Filter out related urls with too similar contents. If two url nodes have highly overlapping entities and similar topicality scores, it means that these two urls have very similar contents. They are not good recommendation candidates for each other. 3) Filter out any url with a large number of related urls. This type of urls are typically hub pages that summarize and point to many articles.

Based on the overlapping entities between two url nodes and their topicality scores, the system calculates the weight of an edge between two nodes. The system then builds (430) a url graph by constructing edges among all urls with overlapping entities. After the url graph is built, the system saves (440) the graph into an url graph bigtable.

Fig. 4(b) illustrates a method of constructing an edge between two url nodes. Node url1 contains three entities, topic-1, topic-2, and topic -3. The topicality scores for the three entities

are 0.9, 0.8, and 0.6 respectively. Thus, the feature weight vector for url1, denoted as W1, is [0.9, 0.8, 0.6]. Node url2 contains two entities, topic-1 and topic-3. The topicality scores for the two entities are 0.9 and 0.7 respectively. Thus, the feature weight vector for url2, denoted as W2, is [0.9, 0, 0.7]. Because the two urls have two overlapping entities, the system constructs an edge between the two nodes. The edge weight is calculated as the "dot" product of the two feature weight vectors:

$$\sum_{i=1}^{N} W1(i) * W2(i)$$

where W1(i) is the ith feature weight of url1 and W2(i) is the ith feature weight of url2. If two url nodes have completely or mostly overlapping entities and similar topicality scores, it means that these two urls have very similar contents. The system does not construct an edge between these two urls because they are not good recommendation candidates for each other.

Fig. 5 illustrates the system architecture for the serving pipeline. After a user clicks (510) on a url, the system queries (520) the url graph storage (530) for related urls and their edge weights. After the related urls are obtained (540; see also FIG. 4 element 440), the system ranks (550) them by ascending edge weights. The higher the edge weights, the more closely the contents are related. However, if two websites' contents are too closely related, the related url seems to be a hub website, or the related url indicates only one entity, they are not good recommendation candidates and the system filters out (560) those urls. Alternatively, the system pre-filters out urls that are too similar in content with the serving url, or indicate a hub url, or reference only one entity, as described with reference to FIG. 4(a) element 420 and FIG. 4(b).

In another embodiment, in addition to closely related contents, the system also filters out (560) urls with a low click through rate before serving the top-N related urls. The click through rate is defined as the number of clicks divided by the number of impressions. If an url has been recommended for a certain number of times but cumulates very low click through rate, it may be removed from the recommendation list.

After filtering out (560) urls based on various criteria, the system serves (570) the top-N related urls on the website that the user is currently viewing. Hence, the top-N related urls are served as personal content-related recommendations to the user.

The subject matter described in this disclosure can be implemented in software and/or hardware (for example, computers, circuits, or processors). The subject matter can be implemented on a single device or across multiple devices (for example, a client device and a server device). Devices implementing the subject matter can be connected through a wired and/or wireless network. Such devices can receive inputs from a user (for example, from a mouse, keyboard, or touchscreen) and produce an output to a user (for example, through a display). Specific examples disclosed are provided for illustrative purposes and do not limit the scope of the disclosure.

DRAWINGS

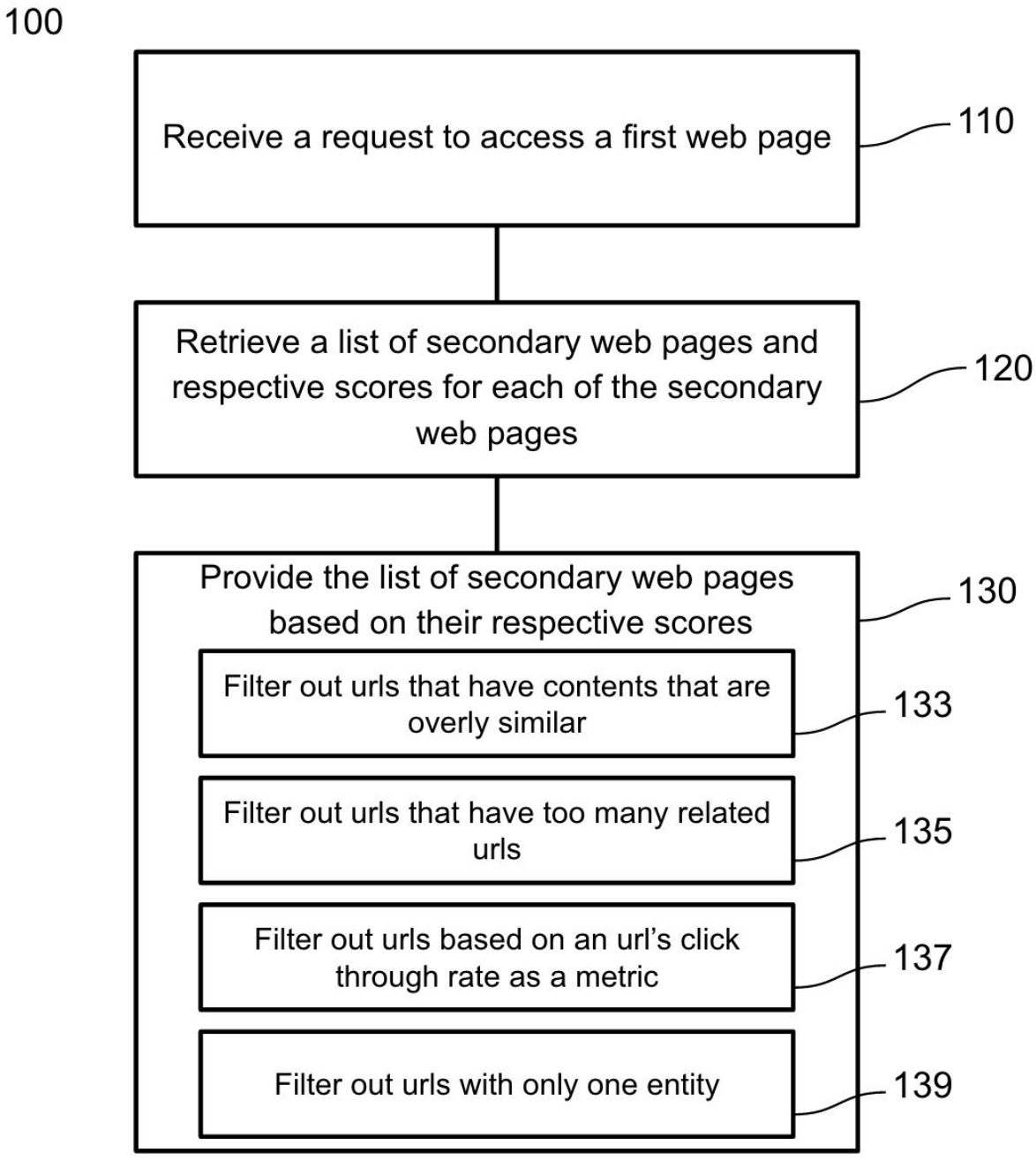

Fig. 1

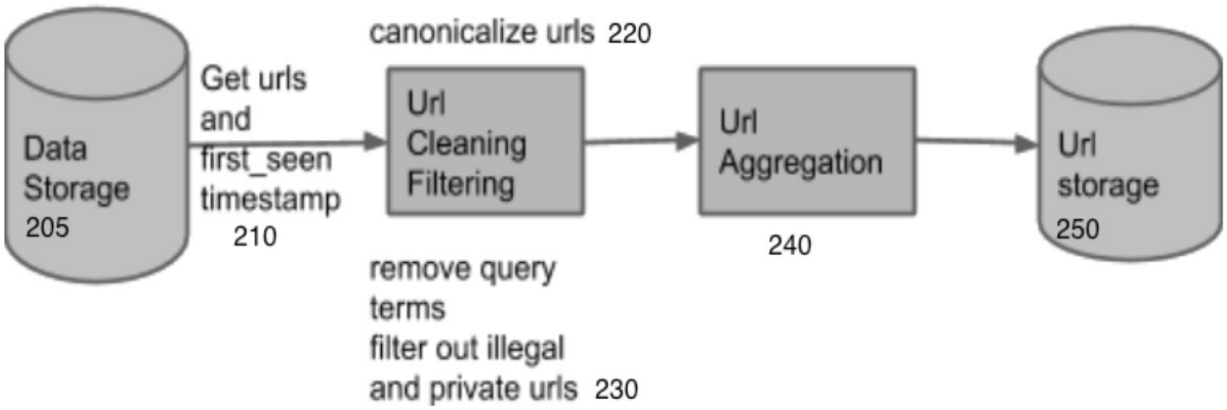
canonicalize urls 220
Get urls and first_seen timestamp 210
Data Storage 205
Url Cleaning Filtering
Url Aggregation
240
Url storage 250
remove query terms
filter out illegal and private urls 230

Fig. 2

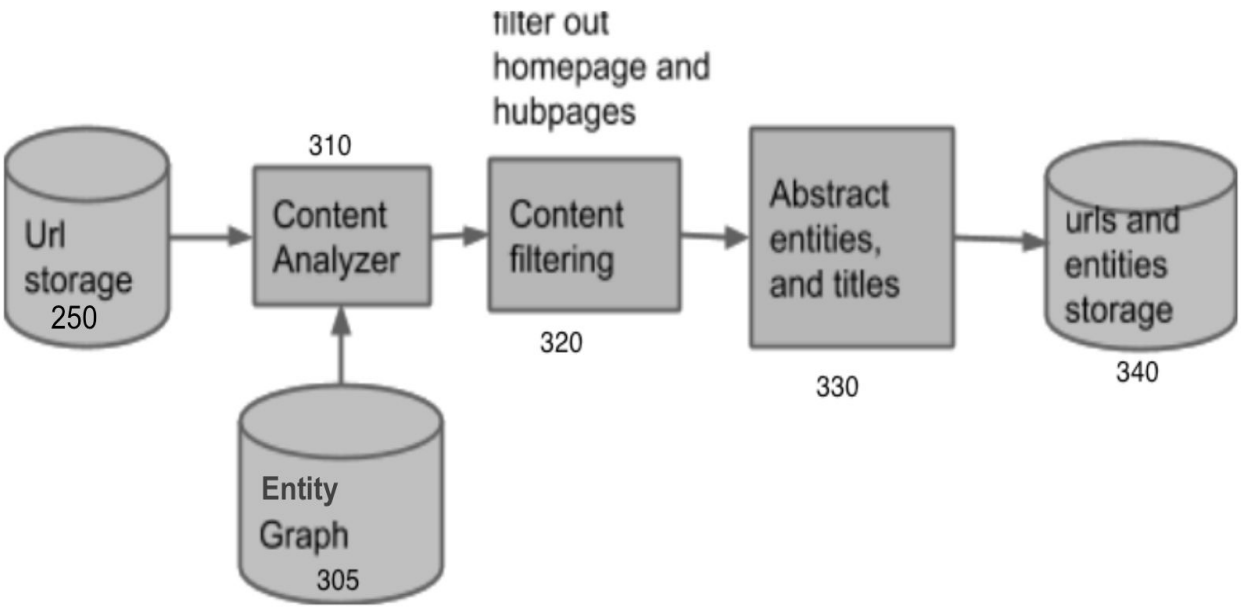
filter out
homepage and
hubpages
310
Url
storage
250
Content
Analyzer
Content
filtering
320
Abstract
entities,
and titles
330
urls and
entities
storage
340
Entity
Graph
305

Fig. 3

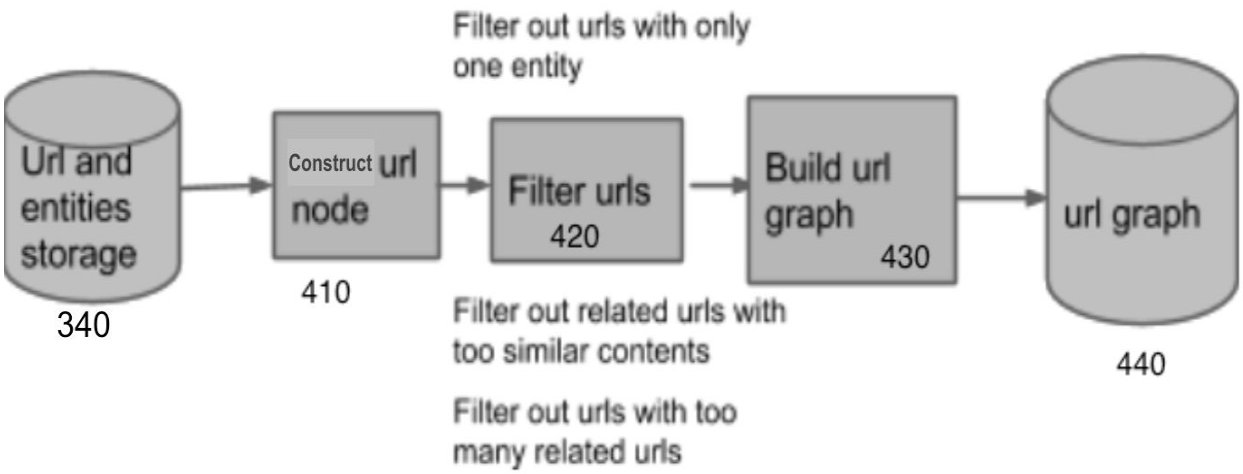
Filter out urls with only one entity
Url and entities storage
340
Construct url node
410
Filter urls
420
Build url graph
430
url graph
440
Filter out related urls with too similar contents
Filter out urls with too many related urls

Fig. 4(a)

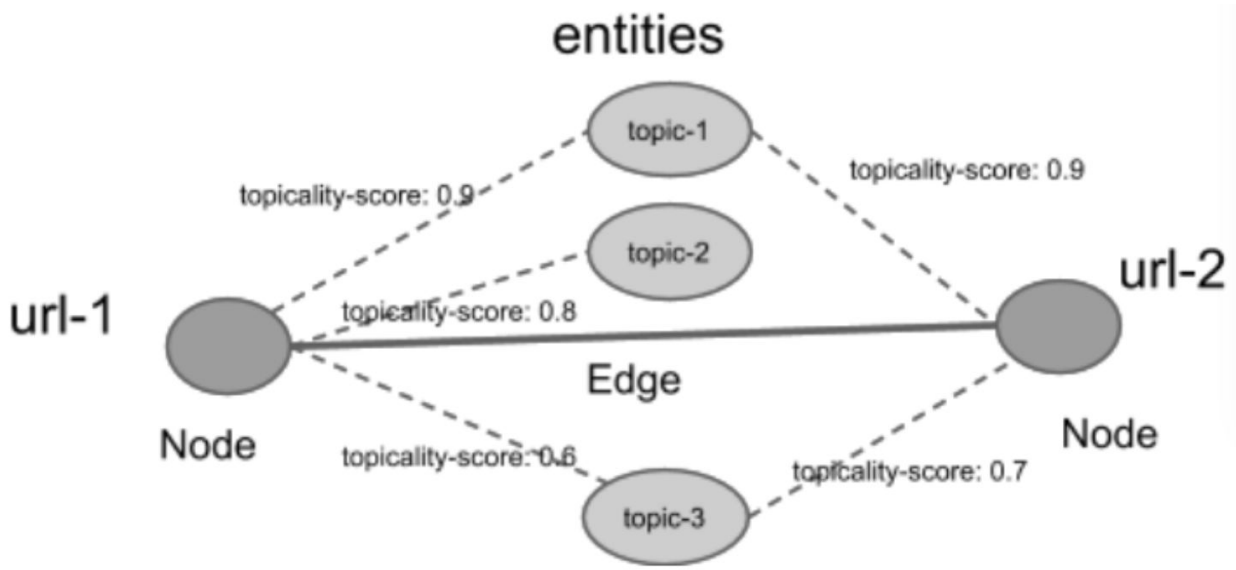
entities
topic-1
topicality-score: 0.9
topicality-score: 0.9
topic-2
url-2
url-1
topicality-score: 0.8
Edge
Node
Node
topicality-score: 0.6
topicality-score: 0.7
topic-3

Fig. 4(b)

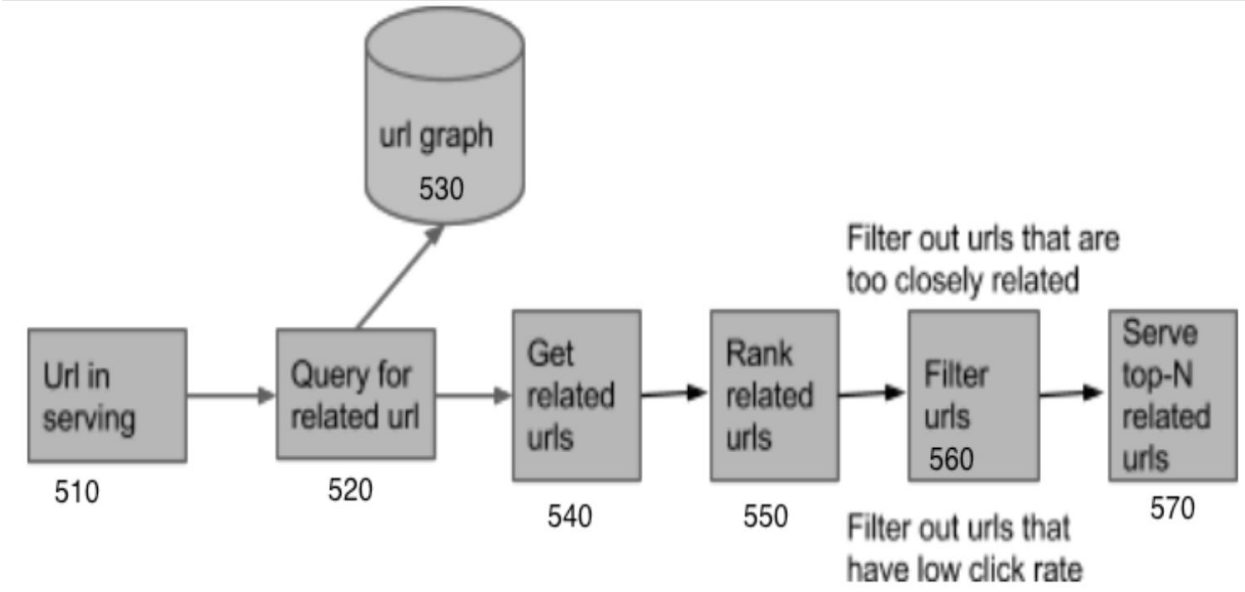
url graph
530
Filter out urls that are
too closely related
Url in
serving
Query for
related url
Get
related
urls
Rank
related
urls
Filter
urls
560
Serve
top-N
related
urls
510
520
540
550
570
Filter out urls that
have low click rate

Fig. 5