Data Sci. Eng. (2016) 1(2):101–113
DOI 10.1007/s41019-016-0012-2

INVITED PAPER

UniClip: Leveraging Web Search for Universal Clipping of Articles on Mobile

Ruihua Song[1] · **Kazutoshi Umemoto**[2] · **Jian-Yun Nie**[3] · **Xing Xie**[1] · **Katsumi Tanaka**[2] · **Yong Rui**[1]

Received: 26 April 2016 / Accepted: 29 June 2016 / Published online: 18 July 2016

Abstract In this paper we address the difficulty of clipping articles from mobile apps. We propose a service called UniClip that allows a user to save the full content of an article by snapping a screenshot part of it. UniClip leverages a huge amount of indexed web data to mine the article by starting with a snapped screenshot. We propose approaches to solve three challenges: (1) how to represent a screenshot; (2) how to formulate effective queries for retrieving a full article; and (3) how to rank the best URL at the top from multiple search result lists. Experimental results indicate that our approach is effective in achieving as high an F_1 measure as 0.905, which outperforms the best of three baseline methods by 18 points.

Keywords Universal clipping · Search · Screenshots · Article clipping · Mobile apps

✉ Ruihua Song
rsong@microsoft.com

Kazutoshi Umemoto
umemoto@dl.kuis.kyoto-u.ac.jp

Jian-Yun Nie
nie@iro.umontreal.ca

Xing Xie
xingx@microsoft.com

Katsumi Tanaka
tanaka@dl.kuis.kyoto-u.ac.jp

Yong Rui
yongrui@microsoft.com

[1] Microsoft Research Asia, Beijing, China

[2] Kyoto University, Kyoto, Japan

[3] University of Montreal, Montreal, Canada

1 Introduction

According to a report from the Internet Trends Conference held in May 2014,[1] the use of mobile devices (including smartphones and tablets) has surpassed PCs since 2010 and continues to grow (Meeker, 2014). In 2013, the number of smartphones shipped was more than three times as many as that of PCs. This report also shows that the fraction of page views via mobile devices has been rapidly increasing year by year (14 % in 2013 to 25 % in 2014). Different from PCs, the primary method of reading on mobile devices is not browser-centered, but application-driven. Information providers develop their own applications or apps. Users install apps for different purposes. It is common that a user uses several apps to access information. However, what a user reads or likes is likely to be scattered in different apps or buried by never ending updating streams. There are increasing demands for developing effective and user-friendly tools to save articles in one place from different apps.

Some note-taking apps, like OneNote, EverNote, Pocket, and EverClip2, have been developed to solve the problem. These solutions are far from satisfactory for a number of reasons. Note-taking apps usually depend on the "Share", "Copy", or "Copy the link" interfaces to receive content from other apps. However, the decisions on whether to support sharing with an app or what to share or copy are fully controlled by app developers. It is difficult for note-taking apps to facilitate partnerships with all reading related apps. For example, Facebook Paper and Klout do not support any note-taking apps. They prefer sharing articles with social networks. More issues with the current systems will be discussed in Sect. 2.1.

[1] http://www.kpcb.com/internet-trends.

In this paper, rather than getting support from all reading apps, we propose taking a screenshot whenever a user is interested in an article and leveraging search and the huge amount of Web data available to discover the full article from the Web. We call the framework UniClip (a shorthand for Universal Clipping)[2] as it is a universal way compared to the existing note-taking apps (such as OneNote, EverNote, and Pocket) that work with only partner reading apps.

We propose approaches to solve three challenges in our search by screenshots task: (1) how to represent a screenshot, (2) how to formulate effective queries to retrieve the article, and (3) how to aggregate search results of the queries. We create benchmarks to evaluate the methods and baselines. Experimental results indicate that our proposed methods are effective in discovering the URL of full article. Our best method achieves as high as 90.5 % in terms of F_1 measure and outperforms baseline keyword extraction methods by over 18 points. It is even better than competiting commercial app Sight (launched in July 2014 after we implemented the prototype of UniClip. It cannot be used now as the service was shutdown in June 2015) by over 8 % in precision and 14 % in recall.

This paper has three main contributions:

- We formulate a new research problem of search by screenshots for augmenting a part of articles on a screenshot by search, which can be used for universal clipping from mobile apps;
- We propose a general framework to solve the problem. Specifically, we propose methods to segment a screenshot into semantic blocks, labeling title and body blocks, formulating effective queries, and aggregating search results.
- We create benchmarks and choose measures to evaluate effectiveness. Experimental results show that our proposed methods are effective in discovering target URLs. They achieve large improvements over our baselines and the competitor.

The remaining parts of this paper are organized as follows: we review existing related work in both industry and academia in Sect. 2. In Sect. 3, we first formulate the core problem of UniClip and overview a general framework to solve the problem. Then we describe our proposed approaches for implementing the framework in Sect. 4. Section 5 reports our built benchmarks and conducted experiments and results. Finally, we conclude the paper in Sect. 6 and discuss future work.

[2] A demo can be visited at http://uniclip.azurewebsites.net/Uniclip.

2 Related Work

2.1 Related Work from Industry

There are many popular mobile apps that are related to reading articles or taking notes. Popular reading apps include Flipboard, Zite, Klout, Facebook Paper, and NYTimes, and note-taking apps include OneNote, EverNote, Pocket, and EverClip2. Based on our experience using these apps, we summarize that the existing ways to save an article are as follows:

(1) Sharing them to a note-taking app: Usually reading apps provide sharing to social networks and/or supported apps. For example, Zite is a popular personalized news reader. When a user is reading an article in Zite, he/she can touch the ”Share” button to share articles to Pocket, Evernote, or others. The issue is that it is difficult to facilitate partnerships with all related apps. For example, Klout and Facebook Paper do not support any note-taking apps. They provide sharing only with social networks.
(2) Copying and pasting: It is difficult if not impossible to select the whole text of an article. An alternative way is to “Copy” or “Copy the link”. Similar to sharing the article to other apps, many apps provide “Copy” or “Copy the link” options in the sharing menu. The issue is that what content to be copied is fully controlled by app developers. For example, in Zite, clicking on the “Copy” button does not copy an article, but the URL of the article. The URL might expire when users would like to read it later. EverClip2 can help get the web page if the content in clipboard is only a link. However, it cannot deal with cases in which a bookmark of an article (including title, link, image, etc.) is copied from some apps, such as TouTiao (the most popular news aggregation app in China) does.
(3) Sharing them to an email: To alleviate the not-supported issue, some note-taking apps provide a way to add notes by sending an email. For example, OneNote provides me@onenote.com. Users can share an article to the email in an app. The OneNote server will identify a user’s account by the email sender and add a note to the user’s notebook. The method seems universal; however, the apps control what content is sent. And thus the sent content is usually a link with a title and short description. Again, the link may expire in the future. It also troubles users in typing users’ account emails because the default emails may be Apple accounts that are not users’ accounts for OneNote or Pocket.

Compared to the above ways, our proposed UniClip service has three advantages: (1) Straightforwardness: the screenshot shows what a user reads and is interested in. (2) Effortlessness: it takes only one or two seconds to take a screenshot.

No further step is required. (3) Universality: apps can control what to share but users can take a screenshot of whatever they are reading anytime in any app.

Sight (sight.sc) is the most related app to UniClip. It was launched in July 2014, when we implemented our ideas into a demonstration system. Similar to our ideas, Sight allows users to take a snap of what they are reading on a mobile phone and then tries to automatically return an extracted article. As it is a commercial app, no details are available on how they mine the corresponding article. Therefore, to the best of our knowledge, we are the first to discuss the problem of search by screenshots in academia and propose solutions in detail. We also compare our approaches with Sight on effectiveness in Sect. 5.

2.2 Related Work from Academia

2.2.1 Text Reuse

Text reuse is defined as the activity whereby pre-existing written material is reused during the creation of a new text, either intentionally or un-intentionally. Not all text reuse is deemed a cardinal sin. For example, it is perfectly acceptable that texts from a news agency are re-used by journalists in the creation of newspaper articles (see, e.g., [12]), as long as the journalist or the organization they write for are subscribers of the news agency. Actually in creating benchmarks and labeling mined URLs for our work, we often find some copies with exactly the same content but different titles, or some articles having similar content but paraphrased or in different length.

Plagiarism, as an example of text reuse, has received much attention from both the academic and commercial communities [11]. Early research works (like [16,22,31,36])were motivated to detect ”unusual” similarity between programming assignments handed in by students. Later more studies (like [5,9,27]) aimed to detect plagiarism between natural language texts, in particular verbatim cut-and-paste from Web based sources and the same content but paraphrased. Some services on plagiarism detection, such as plagiarism.org, turnitin.com, and contentguard.com, are also available online.

Near-duplicate detection is devoted to finding duplicate pages or documents. Different from detecting plagiarism, the technology is used to track web evolution or improve search quality. The state-of-the-art solution is considered as the shingling algorithm in [6] and the random projection based approach in [8]. Moreover, lots of adaptive approaches were proposed to satisfy various requirements, such as [9,10,15,20,26].

Our goal is different from plagiarism detection and near-duplicate detection. We would like to find not similar or relevant but exactly the same full article by a part of an article shown in a screenshot. It is possible to use text similarity measurements proposed in plagiarism detection and near-duplicate detection to calculate the similarity between the captured screenshot and the discovered document from the web. However, we decide not to do that for performance considerations. It takes time to download candidate documents for calculating similarity. To be more efficient, we leverage search and make use of voting from multiple search result lists to rank the best matched URL at the top.

2.2.2 Long Queries

As the text on a screenshot is usually longer than an ordinary query, our work is related to three areas of long queries.

First, some studies aim to better understand a natural language long query, like the description of a query in TREC datasets, because such a query is not effective or efficient in search. Kumaran and Allan [23] use Mutual Information (MI) to select ten sub-queries and present them to the user to choose from. Bendersky and Croft [4] propose automatically finding key concepts in long queries. Some other works study query quality predictors and automatic selection of reduced queries [2,3,24]. For example, Kumaran and Carvalho [24] apply learning to rank framework in reducing long queries using query quality predictors, such as Query Clarity [13] and Query Scope [19].

Second, when a query is as long as a document, previous works define it as a separate problem called query by document [35]. The goal is to retrieve similar documents from a large-scale text corpus. Most existing indexing techniques have difficulty dealing with the document query due to high dimensionality and sparse representation of a document query. Weng et al. [35] propose a two-level retrieval solution. A document is decomposed to a compact vector and a few document specific keywords by a dimension reduction approach. They adopt locality sensitive hashing (LSH) to index the compact vectors for quickly finding a set of related documents, and re-rank documents by document specific words.

Third, phrase extraction is used to extract some representative phrases from a document, and then to find related documents containing these representative phrases. Some works [28,29] utilize statistical information to identify suitable phrases; some works [18] also leverage the relationships between phrases; some works [33,34] apply learning algorithms in the extraction process. Mihalcea and Tarau [30] propose TextRank, a graph-based ranking model, for text processing such as keyword and sentence extraction. In keyword extraction setting, this algorithm takes as input a graph whose nodes correspond to words appearing in a target document. When two words co-occur within a window of a given width W, an edge is added between the nodes of these words. This algorithm then calculates PageRank scores of nodes

for this graph and outputs N nodes having high scores as extracted keywords.

The objective of the above studies is to find similar or relevant documents, and thus not all words in the long query are required to appear in the target document. We are different because we aim to find exactly the same document corresponding to the screenshot query. Therefore, any word should appear in the document, including a stop-word, which is usually filtered out in the approaches to long queries. In this paper, we implement keyword extraction methods with different term weights using TF-IDF [7], BM25 [32], and TextRank[30] as our baselines. Experimental results confirm that our proposed methods are much better.

2.2.3 Search by a Photo

Image search has been widely researched. Lei and Yong conduct a great survey on image search over the last 20 years [40]. They divide the history into several stages: (1) Text-based stage (1970–1990), (2) Content-based stage (1990–2000), and (3) Web-based stage (2000–present). Our proposed framework takes a screenshot image as an input, but we make use of text recognized from the image, rather than the image itself, to compose queries. Moreover, different from image search whose target is usually an image, our work targets a web page containing the article.

Some works [17,37] are related to ours as they propose a retrieval system that receives multimodal queries comprising images and optional text (input by users) and returns images similar to input queries. Yeh et al. [39] utilize images captured by users' mobile devices to suggest location-oriented information to them. Our work is different because we neither use the image as a direct query nor search images as results. We leverage OCR technologies to recognize text from the image and automatically compose text queries to retrieve documents in text.

The most relevant work to ours is done by Yeh et al. [38].They develop a system called Sikuli that enables users to search a large collection of online documentation about GUI elements using screenshots. Sikuli has a database of 102 popular computer books that contain about 50k screenshots. Yeh et al. propose using visual features, surrounding text, and embedded text by OCR for indexing screenshots in the database. At the runtime, a user can specify an interested GUI element by dragging out a rectangle around it and then search it in Sikuli. Their work is similar to ours in using a screenshot for search, but they use the image because users may not know the standard names of GUI elements. Different form this purpose, we use the text on a screenshot to save users efforts in copy-and-paste. Different from our leveraged full text index, they index a small domain of books as well as screenshots. Thus their proposed visual based approaches are not applicable to our addressed problem because users may search articles in any domain.

3 System Overview

Our proposed UniClip framework allows users to save an article from any app with only one action, i.e., taking a screenshot. How UniClip works to save the full article based on the screenshot is shown in Fig. 1. First, a module called screenshot representation will process the input image to obtain text and analyze where the title and body are, if any. Then, a module called query formulation can intelligently compose queries that can potentially identify the article from the web and send the queries to a search engine. Next, a module called result aggregation can discover a URL containing the corresponding full article by aggregating the returned search result lists. Finally, we apply an existing main article extraction module [41] to extract the cleaned full article from the URL and deliver the adaptively rendered article to users.

The core problem with UniClip is search by screenshots. We formulate input/output of the problem as follows:

Input: it is a screenshot of an article that a user is reading via his/her mobile device, where an article is a written work of a specific topic. It can be any part of the whole article (e.g., the beginning part including the title or the middle part containing a few paragraphs). Even a screenshot in which some (but not all) words are partly displayed (because the lines are wider than the screen of device) is also acceptable. A screenshot is regarded as invalid input if it (a) consists of multiple articles (e.g., a screenshot of search results) or (b) does not include any text useful for identifying the article (e.g., a screenshot containing advertisements only).

Output: Given an article's screenshot as input, we aim to output a URL of exactly the same article. Note that some articles are specialized for only mobile apps and are not available on the Web. In such cases, ideal output should be empty since outputting false-positive URLs may frustrate users. As the first work, we do not discuss how to decide not to return any URL in this paper.

Our main work is done to solve three challenges of search by screenshots: how to represent a screenshot, how to compose effective queries, and how to discover the best matched URL from search result lists.

4 Our Approach

In this section, we describe detailed approaches of the three main modules.

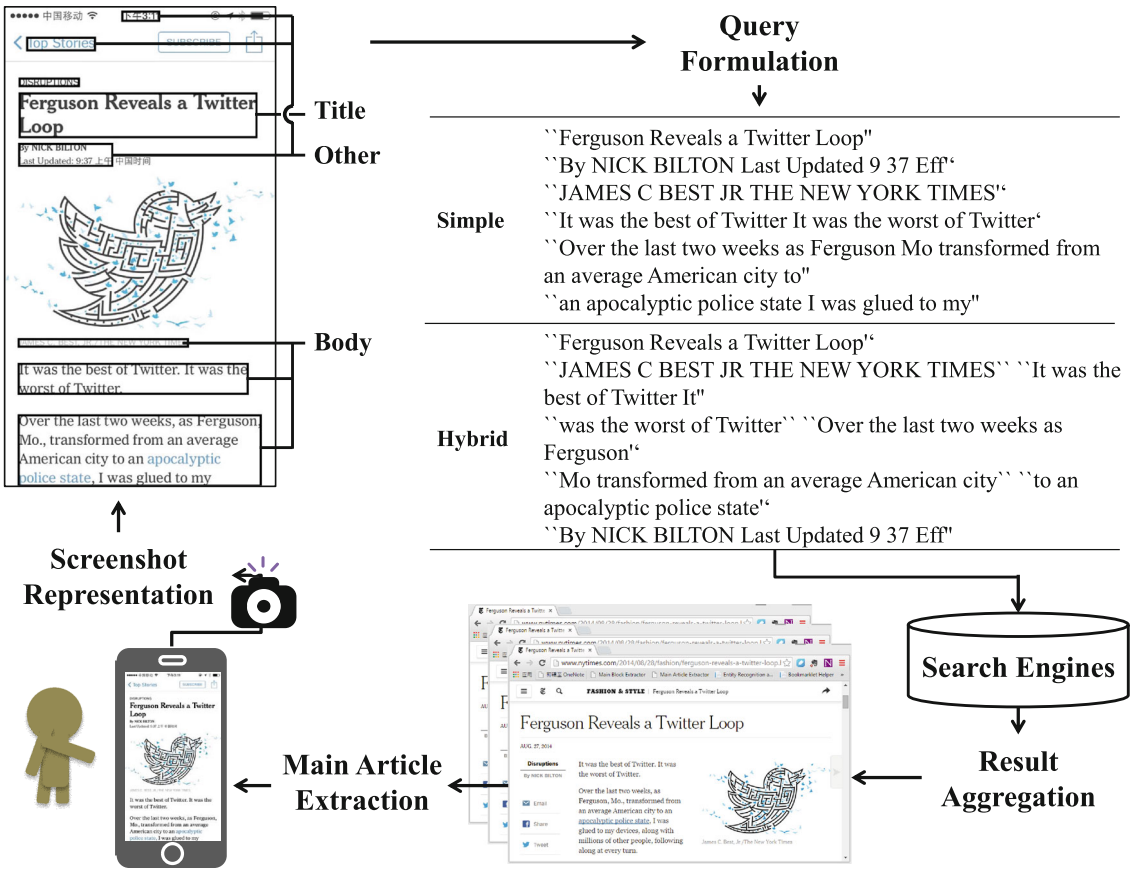

Fig. 1 How UniClip works for users

4.1 Screenshot Segmentation

Given a screenshot, we first apply Optical Character Recognition (OCR) technology [21] to identify text from the screenshot. A recognition result returned by our OCR engine consists of the detected language and a list of lines and their bounding boxes; a line contains a list of words and their bounding boxes; a word also has its bounding box and recognition confidence $\in [0, 1]$. Note that our approach is also applicable to other OCR engines if their recognition results contain information about lines.

Algorithm 1 BUILDSEGMENTS($\mathcal{L}$)

Input: a sequence of lines $\mathcal{L}$ appearing in a recognition result
Output: a sequence of candidate segments $\mathcal{S}_{\text{cand}}$
1: $\mathcal{S}_{\text{cand}} \leftarrow ()$
2: $L_{\text{cur}} \leftarrow$ pick the first element from $\mathcal{L}$; $S \leftarrow (L_{\text{cur}})$
3: **for each** line L_{new} **in** $\mathcal{L}$ **do**
4: **if** L_{cur} and L_{new} look similar and are located closely **then**
5: append L_{new} to the end of S
6: **else**
7: append S to the end of $\mathcal{S}_{\text{cand}}$; $S \leftarrow (L_{\text{new}})$
8: $L_{\text{cur}} \leftarrow L_{\text{new}}$
9: append L_{new} to the end of S
10: **return** $\mathcal{S}_{\text{cand}}$

4.1.1 Block Segmentation

To better represent a screenshot, we propose merging lines into blocks. One line can be a block, although the unit is too small to compose discriminative (usually long) queries. Take the screenshot shown in Fig. 1 as an example. The fifth line of the article title contains only the words "Loop", which is too short to retrieve the article from the Web. In the example, we hope to merge lines of a title into one block. We propose a two-phase segmentation method, whose output for the example is shown in Fig. 1 (each block is marked by a rectangle).

In the first phase, we first build candidate segments by iteratively merging adjacent similar lines. The detailed process is shown in Algorithm 1. At the fourth line in this algorithm, two lines are considered similar if they have similar heights, the distance is less than a threshold, and they have the same alignment. Take the screenshot in Fig. 1 as an example. The fifth line "Loop" is similar to the former line "Ferguson Reveals a Twitter" because they have similar heights, are close to each other and both are left aligned, so they are merged. The

Algorithm 2 REFINESEGMENTS($\mathcal{S}_{\text{cand}}$)

Input: a sequence of candidate segments $\mathcal{S}_{\text{cand}}$ built by Algorithm 1
Output: a sequence of refined segments $\mathcal{S}$
1: $\mathcal{S} \leftarrow ()$; $j_{\text{from}} \leftarrow 0$
2: **while** $i_{\text{from}} <$ Length($\mathcal{S}_{\text{cand}}$) **do**
3: $S \leftarrow ()$
4: $j_{\text{to}} \leftarrow$ last index of segments similar to $\mathcal{S}_{\text{cand}}[j_{\text{from}}]$
5: **for** $j = j_{\text{from}}$ **to** j_{to} **do**
6: append each of lines in $\mathcal{S}_{\text{cand}}[j]$ to the end of S
7: append S to the end of $\mathcal{S}$
8: $j_{\text{from}} \leftarrow j_{\text{to}} + 1$
9: **return** $\mathcal{S}$

Algorithm 3 GENERATEQUERIES($B, n_{\min}, n_{\max}$)

Input: a block B from which queries are generated
min. $n_{\min}$ and max. $n_{\max}$ length of generated queries in words
Output: a sequence of phrase queries Q
1: $Q \leftarrow ()$; $t \leftarrow$ cleaned text in B
2: **while** Length(t) > 0 **do**
3: $s \leftarrow$ select at most $n_{\max}$ words from t
4: $t \leftarrow t$[Length(s), Length(t) $- 1$]
5: **if** Length(s) $\geq n_{\min}$ **then**
6: append Quote(s) to the end of Q
7: **return** Q

sixth author line is smaller than the fifth line and they are far from each other. Thus the sixth line will be added to a new segment.

In the second phase, we refine the previous segmentation by merging adjacent candidate segments that share similar structures. This phrase is required because a paragraph is sometimes over-segmented into several candidate segments in the first phase due to the difference in height of the two lines on the boundary. We address this issue by considering features about the whole segment, e.g., font-size approximated by the average height of lines, when finding two segments that should be merged. Algorithm 2 shows the detailed process of the second phase. At the fourth line in this algorithm, two candidate segments are considered similar if (1) they share the same alignment, (2) they have similar font-size, or (3) the space between them is less than a threshold.

4.1.2 Block Attribute Prediction

Not all blocks are equally important in retrieving the full article. Article title block and body blocks are important as they are parts of the article. Some other blocks, such as ads or toolbars, are useless or even harmful for later modules. In this paper, we apply the Conditional Random Field (CRF) [25] method to label three block attributes, i.e., title, body, and others.

We extract nine features for each line as shown in Table 1. Three features are related to title, e.g., the font size (estimated by the height of bounding box) of a line as a title is usually in a larger font than body paragraphs. Three features are related to body, e.g., the presence of punctuations. The other three features are related to consistency between two adjacent lines as the lines in similar styles are likely to have the same attribute label. We discretize feature values using certain thresholds as was done in [1], which makes the features more tractable by CRF. All bins for discretization are also shown in Table 1.

Once all lines in a block are labeled by our CRF model, we determine the attribute of the block by majority voting. If more than one attribute gets the most votes, we use the attribute that appears earlier as the block attribute. An example of an attribute prediction result is shown in Fig. 1. The rectangles mark blocks. Article title and two paragraphs are correctly predicted. The caption of article image is predicted as "Body". It is acceptable as it is a part of the article. Other blocks are correctly predicted as "Other" attributes.

4.2 Query Formulation

A straightforward idea for query formulation may be extracting keywords from the screenshot text; however, given that our objective is to retrieve exactly the same article, useful information will be lost even if some words are removed or sequence information is ignored. Thus, we propose formulating phrase queries from blocks as a simple method and an advanced method that applies different strategies based on block attributes.

4.2.1 Simple Method

We propose using quoted phrase queries, i.e., enclosed continuously occurred terms with double quotation marks around them. Search engines perform the exact-match algorithm for such queries. Thus the queries can make use of all words (including stop words) in an article as well as other contextual information like term order and proximity. In experiments, we find that a query is not discriminative enough if it is too short. For example, a single word query most likely cannot represent the article and retrieve noisy documents. Thus, we restrict the minimum length of formulated queries to be σ_{qmin}. We also restrict the length of formulated queries not to be more than σ_{qmax} because search engines have their internal limitations on query length due to efficiency and do not perform the exact-match for queries that are too long. How to generate a simple query is described in Algorithm 3. As Fig. 2b shows, simple queries are composed within each block of the example screenshot.

We investigate the effect of quoted phrase queries in different lengths on Bing Search and empirically set the optimal values as $\sigma_{\text{qmax}} = 14$ and $\sigma_{\text{qmin}} = 4$. Take the screenshot in Fig. 1 as an example. From the title block, the simple query "Ferguson Reveals a Twitter Loop" is formulated. From the

Table 1 List of features used for estimating attribute of a given line

Group	Name	Description	Bins
Title	FontSize	Font size of a line approximated by the mean height of words in the line	Small, medium, large
	Confidence	Recognition confidence averaged by words in a line	Low, middle, high
	VerticalPosition	Vertical appearance position of a line	Beginning, middle, ending
Body	WordCount	Number of words appearing in a line	Below, 2, 3, 4, 5, above 5
	Punctuation	Presence of punctuations in a line (for each punctuation including “,”, “.”, “?”, and all of them)	True, false
	LetterCase	Presence of a certain letter case in line (for each style such as lower/upper-case and only-number)	True, false
Consistency	Alignment	Coherence of (any of left-, right-, and center-) alignment of successive two lines	N/A, mismatch, match
	Distance	Distance of the top position of the current line from the bottom position of the previous line	N/A, close, near, far
	Height	Difference of the height of successive two lines	N/A, similar, different

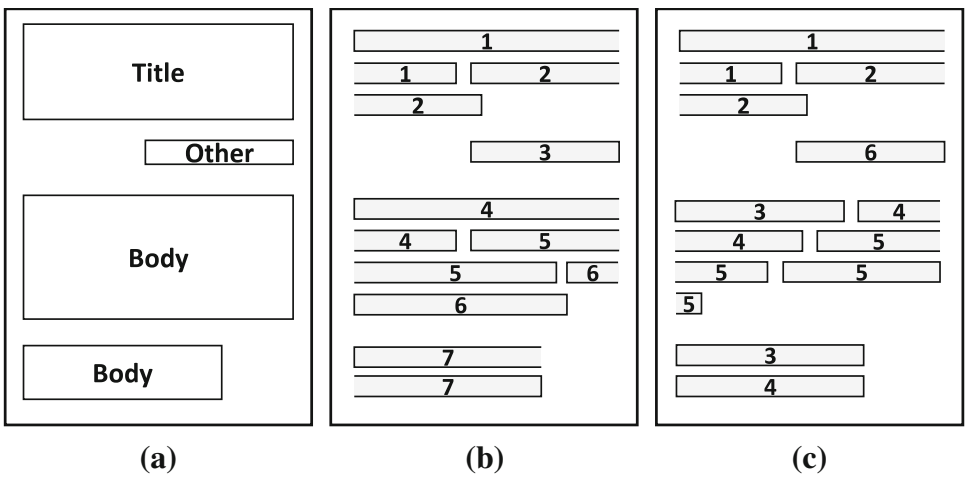

Fig. 2 Contrast of two query formulation methods. Given a segmentation result shown in the left, the Simple method formulates *simple* queries from each segment while the Hybrid method employs *hybrid* query formulation strategies in accordance with the segment attribute. **a** Attribute blocks, **b** Simple method, **c** Hybrid method

last body block, two simple queries are formulated because the number of words in this block is larger than σ_{qmax}. The first query contains the first 14 words, whereas the second query contains the remaining nine words. No simple query is generated from the block of “Top Stories” because the number of words is less than σ_{qmin}.

4.2.2 Hybrid Method

Simple queries from body blocks sometimes are not so discriminative. For example, in Fig. 3a, the first long paragraph quotes a review from Hollywood Report and the second long paragraph quotes a review from The Wrap. The queries that come from such a paragraph will retrieve the original reports or some other articles that use the quote too. Fortunately we find that it is less likely that two paragraphs from two different articles are all the same. Thus, we propose composing Compound queries rather than Simple queries for body blocks.

To formulate compound queries, we first formulate half-length simple queries by Algorithm 3, whose length is between $\sigma_{\text{qmax}}/2$ and $\sigma_{\text{qmin}}/2$, from each body block Bi as their component query set C_i. For example, from the second body block in Fig. 1, two component queries are generated:

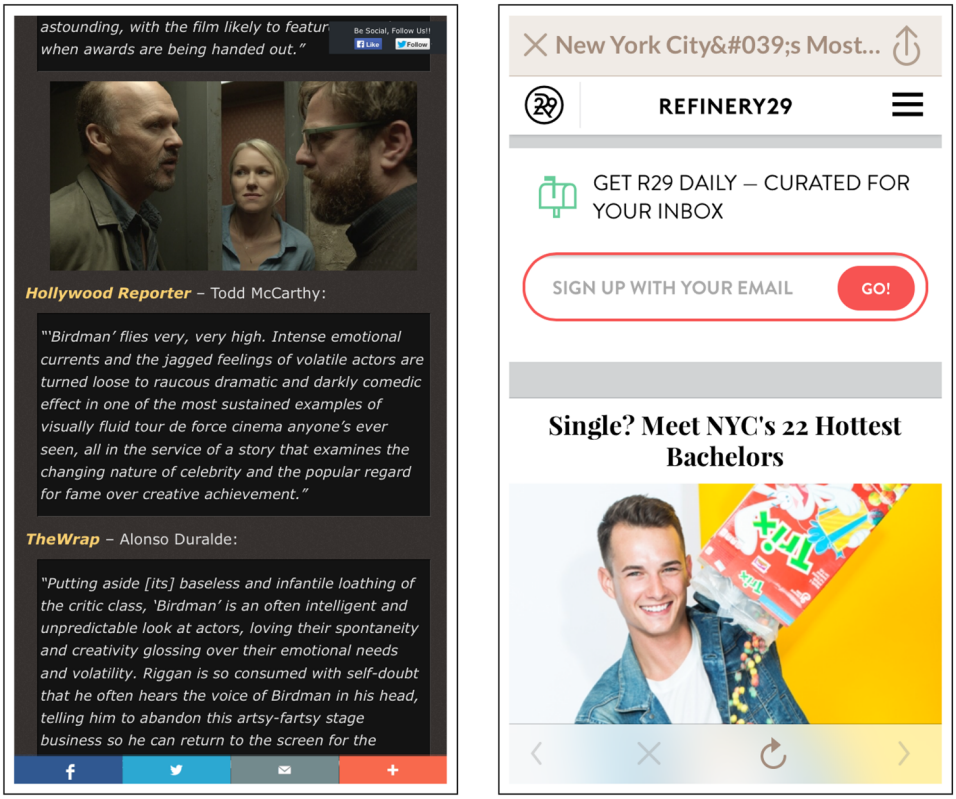

Fig. 3 Examples of screenshots. **a** On hybrid method, **b** On query weighting

"It was the best of Twitter It" and "was the worst of Twitter" given $\sigma_{\text{qmax}} = 14$ and $\sigma_{\text{qmin}} = 4$. We then formulate compound queries by (1) selecting two body blocks containing any unused component query and (2) combining a component query in the first block with a space character and another component query from the second one. For example, in the case shown in Fig. 1, two component queries, "JAMES C BEST JR THE NEW YORK TIMES" from the first body block and "It was the best of Twitter It" from the second, are combined into one compound query, i.e., *"JAMES C BEST JR THE NEW YORK TIMES" "It was the best of Twitter It"*. When there remains no component query in one of the two body blocks, we select another body block according to the occurring order. Note that if only some component queries from one body block are left, we combine component queries from the single block

Overall, we adopt a hybrid approach where different query formulation strategies are used depending on the attributes of blocks. As shown in Fig. 2c, for body blocks, we use compound queries, whereas, for title and other blocks, we still use simple queries. The reason is that it is risky to combine a high quality block with a low quality block. A title block usually generates high quality queries, which are usually discriminative enough to identify the article, whereas most "others" blocks generate low quality queries. But we may lose useful queries if ignoring "others" blocks because some of them, such as a body block misclassified as others, are useful. Therefore, we apply the simple query formulation for title and others.

4.3 Result Aggregation

Once queries Q are formulated, we retrieve the top-K search results by issuing each query to a Web search engine. In our experiments, we use Bing and set K as 8. We observe that good queries often return overlapped results and the target URL can be ranked high. Bad queries return diverse search results. Thus it is promising to use Borda Count [14] to aggregate the lists and rank the target URL at the top:

$$\text{Evidence}\,(q, k) = \frac{1}{\sqrt{k}}, \tag{1}$$

where $k \in [1, K]$ is the rank of a retrieved result for a query q.

Furthermore, if attributes are available, queries composed from title or body blocks are more likely to be good queries than those from other blocks. For example, in Fig. 3b, only the line "Single ? Meet NYC's 22 Hottest Bachelors" is useful while others are irrelevant to our seeking article. Therefore, we integrate query weights into the Borda Count formula as follows:

Table 2 Comparison of methods on attribute prediction

Method	Title		Boby	
	Precision/recall	F_1	Precision/recall	F_1
Heuristic (micro)	0.199/0.784	0.317	0.830/0.726	0.775
CRF (micro)	**0.830/0.803**	**0.816**	**0.982/0.914**	**0.947**
Heuristic (macro)	0.340/0.912	0.327	0.754/0.780	0.702
CRF (macro)	**0.928/0.919**	**0.868**	**0.967/0.880**	**0.893**

The bold is used to highlight better results in terms of different measures. For example, in terms of micro-F1 on title field, CRF performs better than Heuristic. In terms of macro-F_1 on title field, CRF also outperforms Heuristic.

$$\text{Evidence}(q, k) = \frac{w(q)}{\sqrt{k}}, \tag{2}$$

where $w(q)$ is the weight for the attribute block from which the query q is formulated. To distinguish from the Borda Count aggregation, we call it weighted Borda Count aggregation. We use the maximum likelihood estimator for each attribute using a training dataset. The weights of title, body, and others are set as 0.852, 0.778, and 0.252 in our experiments.

5 Experiments

5.1 Experimental Setup

We create two datasets. One is for training our CRF model and tuning parameters and the other is for testing. We collect 100 screenshots for training in May 2014 and 200 screenshots for testing in August 2014. The screenshots are captured from different mobile apps (including news apps like NYTimes, aggregated news apps like Facebook Paper, and other popular apps like Etsy) and mobile Web browsers. As the Web is changing, the corresponding pages of some screenshots had expired when we were conducting experiments in November 2014. As a result of filtering out those screenshots, 98 screenshots remain in the training dataset and 189 screenshots in the testing dataset.

The screenshots in our testing dataset are taken in different conditions to evaluate how well our approach can deal with a wide range of real situations. 36.5 % of the screenshots are captured from the beginning of articles, 58.7 % captured in the middle, and 4.8 % captured from the end. Only 36.5 % of them contain a complete title, while 15.3 % have part of a title. 48.1 % of the screenshots contain at least one image, 6.9 % contains at least one video, and 25.4 % contain advertisements. Images, videos, or advertisements occupy space but provide less useful information in searching articles by text. Sometimes advertisements even provide noise text. It is more challenging but natural to include such cases in datasets.

We hire annotators to manually search on the Web and identify the best URLs. They assign one of the following five categories to each of their found pages: "perfect", "same content but different source", "same content but different title", "related topic", and "totally different". In our evaluation, we regard the pages belonging to one of the first three categories as correct answers. We use Google in this step to prevent their search/click behavior from influencing ranking algorithms of Bing, which is used in our automatic approach.

5.2 Experiments on Screenshot Representation

We conduct experiments to compare the prediction performance of our CRF-based method (denoted by CRF) with a heuristic method (denoted by Heuristic). The Heuristic method regards a block that has the largest font-size and contains more than one word (because the site name is often bigger than a title) as a title. The method takes into account whether a block contains punctuations and its text is long enough in predicting a body block.

We manually label attributes for each OCR line over both the training set and the test set. We train our CRF model and tune parameters of the Heuristic method using the training set. Then, we evaluate the two methods over the test set. We use precision, recall, and F_1 measure for each class. Both Macro and Micro measures are calculated. We show Macro measures in Table 2 and observe similar trends in terms of micro measures.

Overall, the CRF method is much better than heuristic rules in labeling blocks. Table 2 shows that the Heuristic method performs badly on title prediction in terms of precision (only 0.34). It is partially because about a half of screenshots do not contain a title, but the Heuristic method selects one anyway. Such a decision somehow hurts the recall of body blocks. Thus it is not surprising that the Heuristic method performs worse than the CRF method in terms of recall for body blocks. The Heuristic method is also worse than the CRF method by 21 points in terms of precision for bodies. This is strong evidence that the Heuristic method is not effective in distinguishing a body block from other blocks, whereas, the CRF method achieves precision as high as 0.967. The CRF method is also effective in identifying a

Table 3 Contribution of segmentation and attribute prediction in terms of final effectiveness

Method	Precision/recall	F_1	#Queries
Simple w/o block	0.849/0.831	0.840 (−2.6 %)	11.6
Simple	0.862/0.862	0.862	7.3
Simple w/ block	0.905/0.905	0.905 (+5.0 %)	7.3

title block while rejecting others as it achieves 0.928 precision for titles.

We also evaluate how block segmentation and attribute prediction contribute to our Simple method in terms of the effectiveness of discovering the best URLs. Results are shown in Table 3. The method of *Simple w/o Block* means that we regard each OCR line as a unit to generate simple queries. Compared to the Simple method that regards our segmented block as a unit, the *Simple w/o Block* method is worse by 2.6 % in terms of F_1 measure and has to submit 4 more queries on average. This indicates that merging lines into semantic blocks does help for both effectiveness and efficiency. When we weight queries by the predicted attributes, the *Simple w/ Block* method further improves the Simple method by 5.0 % in terms of F_1 measure without increasing the number of issued queries. This indicates that block attribute prediction is useful to enhance the effectiveness.

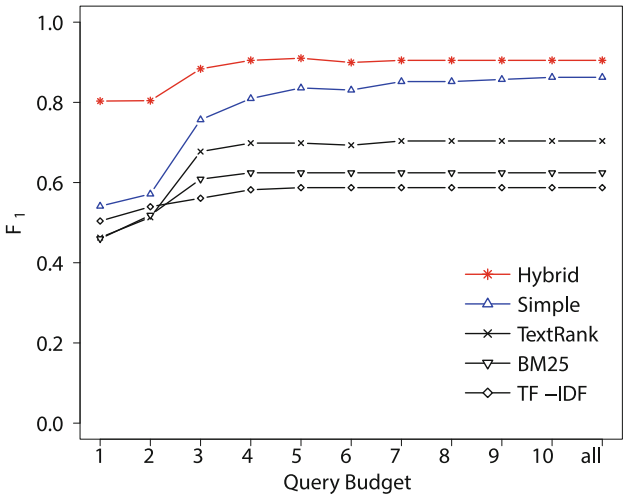

Fig. 4 Comparing of query formulation methods with different query budgets

5.3 Experiments on Query Formulation

We evaluate the retrieval performance of different query formulation methods. We compare our Simple and Hybrid methods with three baseline methods, i.e., TF-IDF [7], BM25 [32], and TextRank [30], all of which extract keywords from the whole text of a screenshot and use them to formulate queries, whose word length ranges in [4,14] too. We do not use quotation marks around the keywords in a query because they are not required to be adjacent. Quoting the extracted keywords may lead to zero search results. All methods use the Borda Count in the aggregation phrase.

As each method may submit multiple queries and aggregate the result lists, the bottleneck of efficiency is calling search engines. It would be better if fewer queries are used in search and aggregation while keeping good effectiveness. Therefore, we draw a curve containing 11 points for each method. The first ten points correspond to the condition when the method submits one through ten queries and aggregates different number of search result lists accordingly. The 11th point corresponds to the condition when there is no limitation of submitted queries and thus the method can submit all queries they generated and aggregate search result lists.

Figure 4 shows the curves in terms of F_1 measure on the testing set. It indicates that our proposed two methods outperform the TextRank method, which achieves the best performance among all baselines, for all conditions. When issuing one or two queries, the baseline methods are similar to the Simple method, but are much worse than the Hybrid method. This indicates that although keyword extraction is useful in selecting important words, the other words, such as stop words, are useful if considering their ordering in our task.

The Hybrid method, which utilizes different query formulation strategies for title, body and others, dramatically improves the Simple method. It improves the F_1 measure from 0.862 to 0.905 when all queries are used. More improvements are gained when the budget of queries is small. For example, if only one query is given as a budget, the Hybrid method improves the Simple method by 48.4 % (from 0.536 to 0.803) in F_1. If the budget is two, the Hybrid method improves the Simple method by 40.7 %. If the budget is three, the Hybrid method improves the Simple method by 16.7 %. This indicates that the Hybrid method is the best among all methods in both effectiveness and efficiency.

5.4 Experiments on Result Aggregation

We conduct experiments to compare two aggregation methods, i.e., the Borda Count and the weighted Borda Count. We evalaute four methods: the first two are the Simple method with or without the weighted aggregation (denoted as Simple and Simple+Weight), and the other two are the Hybrid

Table 4 Comparison of methods in the effectiveness and efficiency of discovering the best URLs

Rank	Method	Precision/recall	F_1	#Queries
1	Hybrid + weighted	0.905/0.905	0.905	6.9
1	Hybrid	0.905/0.905	0.905	6.9
3	Simple + weighted	0.905/0.905	0.905	7.3
4	Simple	0.862/0.862	0.862	7.3
5	Sight	0.838/0.794	0.815	Unknown
6	TextRank	0.725/0.725	0.725	3.7
7	BM25	0.619/0.619	0.613	4.7
8	TF-IDF	0.613/0.613	0.613	5.5

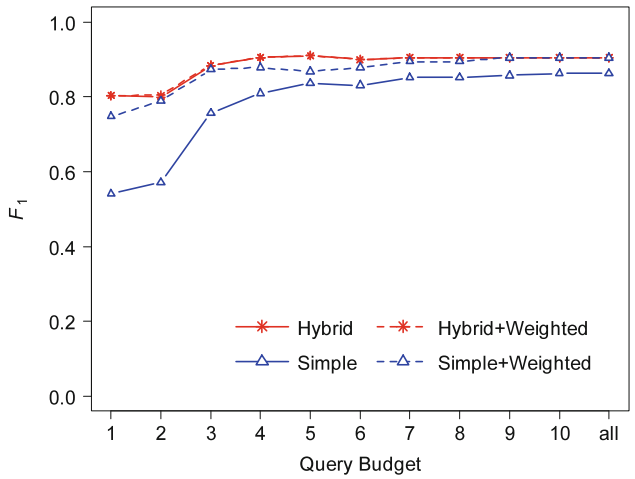

Fig. 5 Comparing of methods on query weighting with different query budgets

method with or without the weighted aggregation (denoted as Hybrid and Hybrid+Weight). Results are shown in Table 4, together with the performance of the competitor commercial app called Sight, without limitations of query budget.

The table indicates that the Simple+Weighted method dramatically improves the Simple method from 0.862 to 0.905. This indicates that the weighted Borda Count aggregation is effective to improve the Simple method. However, we could not observe any further improvement when applying the weighted aggregation to the Hybrid method. It can be explained by the Hybrid method well utilizing the attribute information in formulating queries with different strategies and in ordering those queries from title, body, and others. Thus, it has less room to improve compared to the Simple method.

When we plot the performance curves with different query budgets for the four methods (see Fig. 5), we find that there is almost no difference between the Hybrid and the Hybrid+Weighted curves, whereas, the Hybrid curve is still above the Simple+Weighted curve until they meet when all queries are used. This indicates that the Hybrid method is so far the best way to utilize our screenshot representation and query formulation methods. As Table 4 shows, it also outperforms the commercial app Sight by 8 % in precision and 14 % in recall.

6 Conclusions and Future Work

In this paper, we propose a UniClip service that allows users to clip an article just by taking its screenshot. Given an article's screenshot, UniClip tries to discover the exact same article from the Web and save the full article for users. Compared to existing solutions in industry, UniClip has advantages of taking notes effortlessly and independently from other apps. Experimental results showed that our proposed methods achieved 0.905 in terms of F1 measure, which outperforms the baseline methods.

In the future, we plan to expand from screenshot to photos. For example, a user takes a photo of a page from a magazine and tries to find the full article from the Web. We are also interested in designing new interaction ways to allow users to specify an area they are interested in within the photo. For example, a user can take a photo of some app icons and specify one with their finger. We will then try to search the app and the links to download it.

Acknowledgments We thank our colleagues Qiang Huo, Ivan Stojiljkovic, Magdalena Vukosavljevic and Momcilo Vasilijevic for their help in OCR engines.

References

1. Ageev M, Guo Q, Lagun D, Agichtein E (2011) Find it if you can: a game for modeling different types of web search success using interaction data. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval, SIGIR'11. ACM, New York, pp 345–354. doi:10.1145/2009916.2009965
2. Balasubramanian N, Kumaran G, Carvalho VR (2010) Exploring reductions for long web queries. In: Proceedings of the 33rd International ACM SIGIR conference on research and development in information retrieval, SIGIR '10. ACM, New York, pp 571–578. doi:10.1145/1835449.1835545
3. Balasubramanian N, Kumaran G, Carvalho VR (2010) Predicting query performance on the web. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval, SIGIR '10. ACM, New York, pp 785–786. doi:10.1145/1835449.1835615

4. Bendersky M, Croft WB (2008) Discovering key concepts in verbose queries. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '08. ACM, New York, pp 491–498. doi:10.1145/1390334.1390419
5. Brin S, Davis J, García-Molina H (1995) Copy detection mechanisms for digital documents. SIGMOD Rec 24(2):398–409. doi:10.1145/568271.223855
6. Broder AZ, Glassman SC, Manasse MS, Zweig G (1997) Syntactic clustering of the web. In: Selected papers from the sixth international conference on world wide web. Elsevier Science Publishers Ltd., Essex, pp 1157–1166. http://dl.acm.org/citation.cfm?id=283554.283370
7. Büttcher S, Clarke C, Cormack G (2010) Information retrieval: implementing and evaluating search engines. MIT Press, Cambridge
8. Charikar MS (2002) Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on theory of computing, STOC '02. ACM, New York, pp 380–388. doi:10.1145/509907.509965
9. Cho J, Shivakumar N, Garcia-Molina H (2000) Finding replicated web collections. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data, SIGMOD '00. ACM, New York, pp 355–366. doi:10.1145/342009.335429
10. Chowdhury A, Frieder O, Grossman D, McCabe MC (2002) Collection statistics for fast duplicate document detection. ACM Trans Inf Syst 20(2):171–191. doi:10.1145/506309.506311
11. Clough P (2003) Measuring text reuse. In: PhD thesis, University of Sheffield
12. Clough P, Gaizauskas R, Piao SSL, Wilks Y (2002) Meter: measuring text reuse. In: Proceedings of the 40th annual meeting on association for computational linguistics, ACL '02. Association for Computational Linguistics, Stroudsburg, pp 152–159. doi:10.3115/1073083.1073110
13. Cronen-Townsend S, Zhou Y, Croft WB (2002) Predicting query performance. In: Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '02. ACM, New York, pp 299–306. doi:10.1145/564376.564429
14. de Borda JC (1781) Mémoire sur les élections au scrution. Histoire de l'Académie Royal des Sciences, Paris, pp 657–665
15. Deng F, Rafiei D (2006) Approximately detecting duplicates for streaming data using stable bloom filters. In: Proceedings of the 2006 ACM SIGMOD international conference on management of data, SIGMOD '06. ACM, New York, pp 25–36. doi:10.1145/1142473.1142477
16. Faidhi JAW, Robinson SK (1987) An empirical approach for detecting program similarity and plagiarism within a university programming environment. Comput Educ 11(1):11–19. doi:10.1016/0360-1315(87)90042-X
17. Fan X, Xie X, Li Z, Li M, Ma WY (2005) Photo-to-search: using multimodal queries to search the web from mobile devices. In: Proceedings of the 7th ACM SIGMM international workshop on multimedia information retrieval, MIR '05. ACM, New York, pp 143–150. doi:10.1145/1101826.1101851
18. Frantzi KT (1997) Incorporating context information for the extraction of terms. In: Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics, ACL '98. Association for Computational Linguistics, Stroudsburg, pp 501–503. doi:10.3115/976909.979682
19. He B, Ounis I (2006) Query performance prediction. Inf Syst 31(7):585–594. doi:10.1016/j.is.2005.11.003
20. Henzinger M (2006) Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '06. ACM, New York, pp 284–291. doi:10.1145/1148170.1148222
21. Huo Q, Feng ZD (2003) Improving chinese/english ocr performance by using mce-based character-pair modeling and negative training. In: Proceedings of the seventh international conference on document analysis and recognition, 2003, vol 1, pp 364–368. doi:10.1109/ICDAR.2003.1227690
22. Joy M, Luck M (1999) Plagiarism in programming assignments, vol 22. IEEE Press, Piscataway, pp 129–133. doi:10.1109/13.762946
23. Kumaran G, Allan J (2007) A case for shorter queries, and helping users create them. In: In HLT-NAACL conference. HLT, pp 220–227
24. Kumaran G, Carvalho VR (2009) Reducing long queries using query quality predictors. In: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, SIGIR '09. ACM, New York, pp 564–571. doi:10.1145/1571941.1572038
25. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning, ICML '01. Morgan Kaufmann Publishers Inc., San Francisco, pp 282–289. http://dl.acm.org/citation.cfm?id=645530.655813
26. Lopresti D (1999) Models and algorithms for duplicate document detection. In: Proceedings of the fifth international conference on document analysis and recognition, 1999. ICDAR '99, pp 297–300. doi:10.1109/ICDAR.1999.791783
27. Lyon C, Malcolm J, Dickerson B (2001) Detecting short passages of similar text in large document collections. In: Proceedings of the 2001 conference on empirical methods in natural language processing, pp 118–125
28. Manning CD, Schütze H (1999) Foundations of statistical natural language processing. MIT Press, Cambridge
29. Medelyan O, Witten IH (2006) Thesaurus based automatic keyphrase indexing. In: Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries, JCDL '06. ACM, New York, pp 296–297. doi:10.1145/1141753.1141819
30. Mihalcea R, Tarau P (2004) Textrank: bringing order into texts. In: Lin D, Wu D (eds) Proceedings of the 2004 conference on empirical methods in natural language processing, EMNLP '04. Association for Computational Linguistics, Barcelona, pp 404–411
31. Parker A, Hamblen J (1989) Computer algorithms for plagiarism detection. IEEE Trans Educ 32(2):94–99. doi:10.1109/13.28038
32. Robertson S, Walker S, Jones S, Hancock-Beaulieu M, Gatford M (1994) Okapi at trec-3. In: The third text retrieval conference, pp 109–126. Gaithersburg
33. Tomokiyo T, Hurst, M (2003) A language model approach to keyphrase extraction. In: Proceedings of the ACL 2003 workshop on multiword expressions: analysis, acquisition and treatment—volume 18, MWE '03. Association for Computational Linguistics, Stroudsburg, pp 33–40. doi:10.3115/1119282.1119287
34. Turney PD (2000) Learning algorithms for keyphrase extraction. Inf Retr 2(4):303–336. doi:10.1023/A:1009976227802
35. Weng L, Li Z, Cai R, Zhang Y, Zhou Y, Yang LT, Zhang L (2011) Query by document via a decomposition-based two-level retrieval approach. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval, SIGIR '11. ACM, New York, pp 505–514. doi:10.1145/2009916.2009985
36. Wise M (1993) Running Karp–Rabin matching and greedy string tiling. Technical report. Basser Department of Computer Science, University of Sydney. http://books.google.com.hk/books?id=9OtpAAAACAAJ
37. Xie X, Lu L, Jia M, Li H, Seide F, Ma W (2008) Mobile search with multimodal queries. In: Proceedings of the IEEE, 2008, vol 96, pp 589–601. doi:10.1109/JPROC.2008.916351

38. Yeh T, Chang TH, Miller RC (2009) Sikuli: using GUI screenshots for search and automation. In: Proceedings of the 22nd annual ACM symposium on user interface software and technology, UIST '09. ACM, New York, pp 183–192. doi:10.1145/1622176.1622213
39. Yeh T, Tollmar K, Darrell T (2004) Searching the web with mobile images for location recognition. In: The 2004 IEEE Computer Society conference on computer vision and pattern recognition. IEEE Computer Society, Washington, DC
40. Zhang L, Rui Y (2013) Image search—from thousands to billions in 20 years. ACM Trans Multimed Comput Commun Appl 9(1s):36:1–36:20. doi:10.1145/2490823
41. Zheng S, Song R, Wen J (2007) Template-independent news extraction based on visual consistency. In: The 22nd national conference on artificial intelligence, vol 2. AAAI Press, pp 1507–1512