Blog Posts and Comments Extraction and Impact on Retrieval Effectiveness

Javier Parapar, Jorge López-Castro, and Álvaro Barreiro

IRLab, Computer Science Department
University of A Coruña, Spain
{javierparapar,irlab,barreiro}@udc.es

Abstract. This paper is focused on the extraction of certain parts of a blog: the post and the comments, presenting a technique based on the blog structure and its elements attributes, exploiting similarities and conventions among different blog providers or Content Management Systems (CMS). The impact of the extraction process over retrieval tasks is also explored. Separate evaluation is performed for both goals: extraction is evaluated through human inspection of the results of the extraction technique over a sampling of blogs, while retrieval performance is automatically evaluated through standard TREC methodology and the resources provided by the Blog Track. The results show important and significant improvements over a baseline which does not incorporate the extraction approach.

1 Introduction and Motivation

Blog Search has become an important area in Information Retrieval as blogs became a significant portion of the Internet. In a blog, besides objective information, users may express their opinion, intention or many other data which may be valuable to exploit.

The Text Retrieval Conference (TREC) [1] is one of the main reference conferences promoting now Information Retrieval tasks. The TREC is composed of a set of tracks, to cover different areas of IR. In response to the need of exploring blog search behaviour, arises a new TREC track: the Blog Track. The TREC Blog Track includes an opinion finding task and an ad hoc retrieval task. The opinion finding task aims to gather opinion information expressed through a set of blogs on a given topic set. On the other hand, the ad hoc retrieval task consists of measuring retrieval performance on a blog set for a given query set.

In order to perform the Blog Track's tasks, different blog collections are provided to be used as a sample of the blog population present on the Internet. Currently, Blogs06 and Blogs08 collections are available for their use in the research of Information Retrieval Blog Search. These collections store the full web page of a blog post, plus some metadata about the page itself or the recollection process. In a blog post page, the most significant information will be on the post body itself and on the comments sent by the readers. In order to work with this

information, it is be necessary to extract it from the whole page body. This task must be performed in the same way, no matter whether the blog page comes directly from the Internet or from a previously gathered collection: the HTML source code for the page must be parsed and processed in order to extract the text from the post and the comments.

The main problem about extracting information from blog pages is caused by the lack of any standard way for representing the blog post and the comments. This makes necessary to perform a detection process in order to locate the desired information and, once located, extract it. The extraction process becomes difficult because of the heterogeneity present among blogs. This heterogeneity may be mainly of two kinds: format heterogeneity and content heterogeneity. Different blogs differ in the way they represent post or comments, and the HTML markup used to do it, because any page layout can be achieved through many different markup constructions. Besides, the content itself varies a lot not only through different blogs, but also through different posts in the same blog: a post may be composed by a long text or a single sentence or word, may be just a picture, a video or any combination of them. Although a human observer can trivially spot which region of the page belongs to the post and which belongs to the comments, it gets difficult to define a pattern which can be used in an automated process implemented with a computer.

Our approach for the extraction of posts and comments makes use of the similarities on posts and comments layout across blogs from the same service provider and identifies post and comment regions based on the structure of the page and certain attributes present on its nodes.

The rest of this paper is organized as presented next. In section 2 background in blog extraction and blog search will be introduced. Section 3 presents our purposed approach. Sections 4 and 5 present the evaluation methodology used to judge the extraction accuracy and its impact on retrieval, respectively, along with the the results obtained in each evaluation. Finally in section 6 conclusions and future work are presented.

2 Background

Post extraction and comment extraction from a blog page are two variants of the same general process: extracting text regions from a full HTML page. This task, easy for human eye, is quite hard to perform in an automatic way. This is due to the fact that a human observer will rely on the page's layout to spot post or comments region, but this layout is not fully defined in the HTML source code, because it is modified by external resources like the presence of images an the application of external style layout definitions. Two main complementary approaches are suitable for this purpose:

– Choosing text based on its constituent features. The approach here is trying to pick up text matching a set of conditions defined over its content. For example, a blog post usually is the bigger piece of text of a page or is composed by a collection of consecutive paragraphs, while the comments are

often represented as a list of items, or surrounded by metadata like author or posting date.
– Extract text based on its position on the global structure. This approach takes advantage of the HTML code hierarchical structure and tries to locate text based on its place on the node tree representing the web page: the DOM tree.

Both approaches are affected by heterogeneity, because every CMS may publish posts or comments following a radically different format from any other. In many of them, the author is even able of changing this format as desired.

In [2] the authors make an exhaustive analysis of blog comments for a broad sample of blogs and need to deal with the automatic detection/extraction problem. The authors choose a combined approach: in first place, they try to locate the region containing comments based on their position, usually located between the post and some sort of footer. Next, they look for a list of dates in that region and built the comments from the text present around those dates.

In [3] a few guiding lines are suggested for post and comment extraction. Here, the markup differences between different CMS are pointed. A per CMS classification for a sample of 551,763 blogs is provided as well, remarking CMS use proportion and suggesting to give priority to heuristic rule construction for most representative CMS, i.e. Blogger or WordPress, with a 51.29% and an 18.37% share of the total number of blog posts analysed. Table 1 shows the blog distribution given by the authors.

Table 1. CMS distribution for the blog sample used in [3]

CMS	Number of Posts	Percentage
Blogger	282,982	51.29
WordPress	101,355	18.37
LiveJournal	99,100	17.96
MovableType	9,267	1.68
Technorati	3,562	0.65
UserLand	1,869	0.34
FeedCreator	626	0.11
Unknown	53,002	9.60
Total	551,763	100.00

From this table, we can see that three CMS generate 87.62% of the total blog posts taken into account, while the rest have a marginal presence. This supports the approach based on independent analysis of each CMS.

3 DOM and Attribute-Based Blog Parts Extraction

The approach for extracting blog parts explored in this paper relies on DOM structure of the page and the value of attributes present on nodes to identify desired blog parts.

3.1 DOM and Problem Domain

Every web page can be represented as a tree, known as the DOM (Document Object Model). Each content fragment in the page is represented by a node or a set of nodes. Each node may have several attributes which describe properties for the node. With our approach, we assume an extraction rule can be defined based on the position of the nodes within the DOM and the values of their attributes so, once applied over the DOM, it prunes the rest of nodes, leaving only those nodes belonging to the desired content, this is, the post or the comments.

CSS selectors are suitable for defining our extraction rules. CSS (Cascading Style Sheets) is a style sheet language for defining presentation of documents written in a markup language, such as HTML or XML. A CSS selector is a declarative expression which specifies over which DOM nodes is applied a given style, based on their position on the DOM itself and the value of their attributes. So, CSS can be used for the definition of extraction rules.

With CSS selectors as a powerful tool for extraction rule definitions, criteria for guiding de extraction process should be defined. To deal with differences between different blog generators we propose a CMS-based detection process. Each CMS uses a common markup layout or a limited set of variants to represent the blog content, allowing definition of extraction rules for every blog belonging to any given CMS. Besides, a small number of CMS are used as providers for a high percent of the Internet blog population, as seen in section 2.

Focusing in popular CMS, as suggested in [3], shows two advantages: firstly, every improvement made to one of these CMS boosts the number of documents correctly extracted; in second place, minority CMS often mimic popular ones structure, so covering popular CMS may result as well in a better coverage of marginal CMS.

3.2 Detection of CMS and Extraction Rules Definition

In order to take advantage of the knowledge about the CMS distribution of blogs, it becomes necessary to detect the generating CMS for a given blog post. To achieve this, two approaches are proposed: generator tag based detection and URL based detection.

Among the HTML standard tags, there is a tag used to declare attributes defined by key-value pairs. This is the `meta` tag. The general syntax of this tag is `<meta name="someName" content="string defining content" />`. There is a convention for specifying the generator of a page through a `meta` tag, where the `name` key takes the value `generator` and the name of the generator CMS is stored in `content`. The negative side of this approach is that the specification of CMS is not mandatory, so there is no guarantee about this tag being present in a given page. Nevertheless, whenever this information is present, it provides the more accurate information about the CMS used.

Whenever the generator is not specified in the markup as explained before, URL-based detection can be performed. The approach here exploits the presence of certain patterns in blog URLs. For example, Blogger hosted blogs usually

follow the host name pattern `*.blogspot.com`, or MovableType based blog posts are filed under `/movabletype/*` or `/MTarchives/*` paths. All this information is in fact present in the page URL, so parsing the URL may point to the used CMS. This approach is handicapped for the ability of the user to configure these options: i.e. if a blog is hosted in Blogger, a custom domain name may be used whenever the user has the domain registered, which may lead to miss the CMS detection.

These two approaches can be combined, so tag-based detection is attempted in first place, since it provides more accurate results. If no CMS can be determined this way, URL-based detection is performed then.

At this point, extraction rules for known CMS need to be defined, along with some sort of default extraction rule for the case in which the CMS can not be detected. Intuitively, the blog post will be in a container HTML element with some defined style or attributes, while de comment area will be a list of container elements with same style or layout among them. For a given template, the DOM tree path for any of these elements can be defined, but too many elements may match the path defined. So, markup attributes used in the container will be used in conjunction with the tree path to achieve a more accurate detection. Often `class` and `id` attributes hold the proper information for this purpose, with typical values like `post`, `entry` or `content` for the post and `comment` or `cmt` for each comment. Many variations of these values may appear, such as hyphenations (i.e `post-*` or `comment-*`) or translations of these terms to other languages.

With this information, a filter set is created for posts, and another one for comments, where each filter consists of a set of matching CMS patterns and a extraction rule set applicable for a matching page.

3.3 General Extraction Algorithm

The general extraction algorithm is analogous in both cases: post extraction and comment extraction. It consists of two sequential steps: CMS detection and node extraction

In the first step, CMS detection is attempted. Current page is matched against the CMS pattern set, sorted in descendent order by priority criteria. The priority criteria used are:

- Generator matches have higher priority than URL matches, because the generator tag is an explicit CMS or author declaration, while the URL may be more loose pointing to the CMS.
- More restrictive patterns have higher priority. For example, a match of the type `*.blogger.com` will be considered before `*movabletype*`, because a blog which URL is `http://movabletype.blogspot.com` would be hosted in, and generated by, Blogger, even although it matches the MovableType pattern as well.

For a given page, each pattern is checked in priority order, until a match is found. If no match is found, CMS is considered undetected. The extraction strat-

egy for this default case will be sequentially trying every extraction rule defined, ordered by priority, until a node or set of nodes match one of the extraction rules[1].

Once the CMS is detected, the extraction rules defined for that CMS are applied to extract the information. This step varies, depending on whether comment or post extraction is being performed.

In post extraction, this step is divided in three phases:

1. The selected filter is applied, yielding a list of candidate nodes from the DOM for the post.
2. If the candidate node list is not empty, the first of the candidate nodes is selected, discarding the rest. This is done because in case of multiple node matches, usually the post is located on the first node. If no matching candidate nodes are found, the default extraction rule will be tried, because the common case is the blog post being present within the page.
3. The plain text content of the selected node is the detected post.

```
filter bloggerPostFilter
  matches:
    host=*.blogspot.com
    | generator=Blogger*
  extracts:
    div.blogPost
    | div.post>div.post-body
    | div.posts
    | div.blogposts>div
    | div#blog.blog
    | div.text AND NOT div.blogComments
    | div.bpost
    | div.entries>div
    | span.main AND NOT (span.main>span.main OR font>span.main)
end filter
```

Fig. 1. Filter for extracting posts from blogs generated by Blogger. The *matches* section describes the cases in which the filter is applicable while the *extracts* section contains the extraction rule set for the DOM nodes. The `|` operator preceding a rule means it will be tried after its precedent in case of fail. Matching patterns use `*` as wilcard. Extraction rules follow CSS notation where `.` stands for `class`, `#` for `id` and `>` denotes parent-child relationship. `AND`, `OR` and `NOT` operators are used to combine CSS selectors for an extraction rule.

Figure 1 shows an example post extraction filter. This filter is suitable for URLs with hostname ending with `.blogspot.com` or pages whose generator tag holds a value starting with `Blogger`. Whenever the filter is applicable, it will try to extract text from nodes defined like `<div class= "blogPost">` (first extraction rule), then, if no candidates are found, from nodes like `<div class="post-body">` and children of a node `<div class="post">` (second extraction rule) and so on.

[1] To facilitate the reproducibility the rules will be available at http://www.irlab.org

For the comment extraction, the phases are:

1. The selected filter is applied, yielding a list of candidate nodes for the comments.
2. If the candidate node list is not empty, all nodes obtained are selected, since it is expected that each node represents a comment. If no matching nodes are found, no default action is performed, since among the blogosphere is quite usual that a blog post has no comments at all.
3. For each selected node, a comment will be built with the plain text obtained from that node.

Since some CMS use a similar markup for blog post and blog comments, comments could appear included in the post text extracted or paragraphs part of the post could be detected as comments. To avoid this, extraction rules are modified before applying them to the page: nodes matching comment extraction rules are explicitly excluded from the post, while post nodes are excluded from comment results. This allows us to minimize the cases in which comments appear to be part of the post or where the comments are wrongly built from post paragraphs.

4 Extraction Evaluation

The detection and extraction process needs to be evaluated to measure its accuracy and impact over retrieval effectiveness. For the purposes of the TREC Blog Track, collections of blogs were created to provide a realistic sample of the Internet's blog population. The work for this paper was developed working with one of them, the Blogs06 Collection, whose creation process is described in detail in [4]. The table 2 is a summary of the Blogs06 collection and main features.

Table 2. Blogs06 collection composition summary

Measure	Value
Number of unique blogs	100,649
RSS	62%
Atom	38%
First crawl date	06/12/2005
Last crawl date	21/02/2006
Number of feeds obtained	753,681
Number of permalinks	3,215,171
Number of homepages	324,880
Feed size (uncompressed)	38.6GB
Permalink size (uncompressed)	88.8GB
Homepage size (uncompressed)	20.8GB
Total size (compressed)	25GB
Total size (uncompressed)	148GB

The Blogs06 collection is suitable for its use in Blog Search research because it is designed to mimic the whole blogosphere features, such as spam inclusion,

the proportion between highly visited blogs and anonymous ones, as well as professional or business blogs opposite to personal blogs. Besides, as a TREC collection, relevance judgements are available for their use in retrieval evaluation, as is shown in section 5.

4.1 Settings and Methodology

A manual approach has been chosen to evaluate the results of the extraction process. Because of the size of the collection, it is not easy neither feasible manually checking every document in it, so it was decided to choose a random sample of a reasonable size and manually evaluate the process on every document in this sample. To achieve this, a helper application was created.

The evaluation application shows to the evaluator the blog web page rendered on the left panel and the text extracts detected for the post and the comments on this page on the right panel. The evaluator may choose between three options, correct, incorrect and no answer (N/A) if it can't be decided, separately for each extraction target (post and comments), as shown in figure 2.

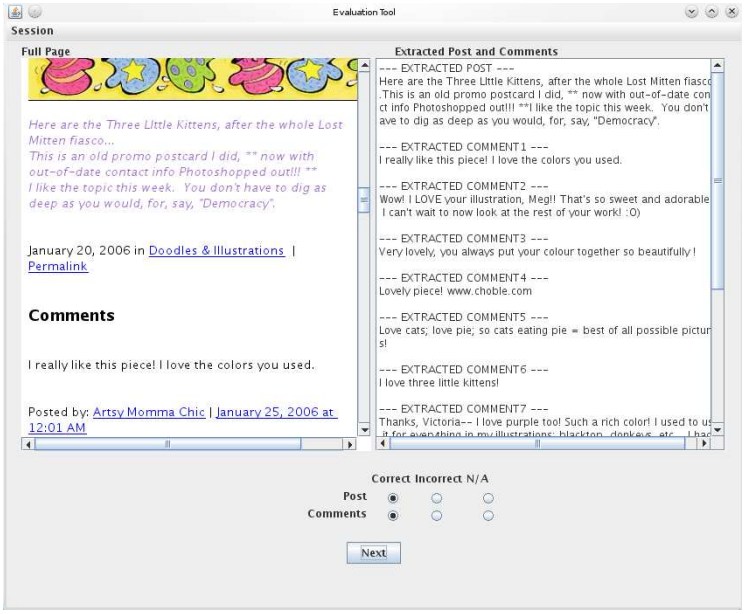

Fig. 2. Screen capture of the evaluation application provided to aid with extraction evaluation

4.2 Evaluation criteria

The human evaluator was instructed to follow the evaluation criteria listed below.

Post extraction is correct whenever one of these conditions is satisfied:

1. The extracted text matches exactly the observed post.
2. The extracted text contains the observed post plus some metadata, such as the posting date, title or the author.
3. The extracted text contains the full observed post plus some marginal text from the page which represents no significant part of the total.
4. The post is neither extracted, nor observed by the evaluator.

Post extraction is incorrect if one of these conditions is satisfied:

1. None of the rightness conditions specified are satisfied.
2. No text is extracted but it is indeed observed by the evaluator.
3. Text is extracted, but the evaluator does not identify a post in the observed page.
4. Marginal text included in the extraction is bigger than the observed post text itself.
5. Comments are included in the extracted post text.

Comment extraction for a document is correct if it satisfies one of these conditions:

1. Extracted text for each comment matches exactly the observed comment.
2. Extracted text for each comment contains the full comment plus metadata, such as posting date, author or title.
3. Extracted comment list contains all observed comments plus some other text conforming fake comments, but these fake comments are less than real comments. An example for this is detecting the comment count as a comment itself, whenever there are one or more observed comments.
4. Every observed comments is detected, although wrongly segmented, i.e. observed comments appear concatenated together as a unique extracted comment.
5. Extracted comment list is empty and the page shows no comments at all.

Comment extraction for a document is wrong if it satisfies one of these conditions:

1. None of the rightness conditions are satisfied.
2. There are extracted comments, but not observed comments.
3. There are not extracted comments, but the comments can be observed on the page.
4. The number of fake comments extracted exceeds the number of proper ones.

Whenever the evaluator is not able to decide whether an extraction satisfies rightness or wrongness conditions, N/A option is assumed as evaluation result.

4.3 Results

The evaluation of the extraction performance was made over a sample of 1,000 documents randomly chosen from the 3,215,171 compounding the Blogs06 collection. The results of the evaluation, obtained following the methodology previously explained, are shown in Table 3.

Table 3. Results of the extraction process for a sample of 1,000 documents. The same sample was used to judge both post and comment extraction.

	Posts	Comments
Correct	821	887
Incorrect	178	112
N/A	1	1
Total	1,000 pages	

As shown in table 3, from the 1,000 random documents, the post was rightly detected in an 82% of them, while the comments were correctly identified on almost an 89%. One of the documents was marked as N/A because it couldn't be evaluated since its markup made the HTML rendering engine used in the application to crash. These figures suggest that the heuristic-based approach chosen is suitable for this problem.

One remarkable fact here is that comment extraction outperformed post extraction. This fact may be due to the fact that commented posts are barely a 31% of the total (997,411 detected, correctly or not, out of 3,215,171) while the extraction reported a post detected for 87% of the documents (2,811,656 of 3,215,171). So, extracting no comments is a good default action since in fact it would be correct around a 70% of the times extraction is attempted, while almost every document in the collection has a blog post. Cases where the extraction process is not able to extract anything usually benefit comment extraction, while penalizing post extraction.

Besides, the extraction process was simultaneously correct for both post and comments in the same document in a 76% of the documents. This reflects a behaviour of some CMS which use a well-defined markup for post but not for comments, or vice versa.

5 Evaluation of the Impact on Retrieval

In order to evaluate how much the extraction process affects depending tasks, we chose to evaluate retrieval over the extracted post and comments and compare it with the retrieval over the page plain text.

5.1 Settings and Methodology

To perform the evaluation, the standard TREC methodology was used, because it is suitable for evaluating large data sets without intervention of a human evaluator. For the TREC Blog Track, a new topic set is provided every year, each of them containing 50 topics. We used the Blogs06 collection as data set, and tested the retrieval with two of these topic sets: topics 851-900 and topics 901-950, topics for year 2006 and year 2007, respectively. Among the fields present for each topic, only the title was used for retrieval, which represents a user query, without any further information added.

For the evaluation task, two indices were created: one holding the full text for each document, obtained extracting the text with a HTML parser from the original pages, and the other containing extracted post concatenated with the extracted comments, obtained through the technique shown in section 3. The index creation was aided by the Lucene [5] library. Our evaluation compares retrieval over these two indices. Higher retrieval effectiveness using the extracted text for post plus comments should point that relevant documents are better found searching over the extracted text and so, meaningful information is condensed within the post and the comments.

We decided to use a high performance state-of-the-art retrieval model: BM25 [6]. The similarity function was computed as follows:

$$R(q,d) = \sum_{t \in q} \frac{tf(t,d)}{k_1((1-b) + b\frac{length(d)}{avg_{dl}}) + tf(t,d))} \times idf(t)$$

Where $tf(t,d)$ is the term t's frequency in document d, $length(d)$ the document length for d, avg_{dl} the average document length across all documents, k_1 is a free parameter, $b \in [0,1]$ as a parameter denoting the degree of length normalization performed and $idf(t)$ is the inverse document frequency of t calculated as $idf(t) = \log \frac{N - df(t) + 0.5}{df(t) + 0.5}$ where N is the number of documents in the colection and $df(t)$ is the document frequency of t.

Using the two topic sets provided, a cross-validation methodology was used for the evaluation. Thus the b parameter for this model was trained using the 851-900 topic set for MAP optimization, yielding the value $b = 0.15$, while the k_1 parameters was left with its recommended value, 2 in this case. Topics 901-950 were used as test set.

To perform a fair evaluation, retrieval on both fields is performed only on those documents for which both post and comments have been extracted. Therefore from 3,215,171 post pages in the Blogs06 collection only 964,019 were selected. This leads to lower effectiveness values when compared to retrieval over the full collection, because part of the relevant documents are not considered.

The measurements used in the evaluation process are those used in the TREC Blog Track for the retrieval task, as seen in [7]: MAP, R-prec and bPref.

5.2 Results

The evaluation results over the Blogs06 collection are summarised in Table 4.

Table 4. Comparison of performance results on retrieval using the full page of the page and the text extracted for the post and comments. Statistical significant improvements according Wilcoxon Signed Rank test ($p\text{-}value < 0.05$) are starred, best values bolded

Topics	*Measure*	*Full text*	*Post + Comments*
	MAP	0.1239	**0.1349*** (+8.88%)
851-900	*R - prec*	0.2093	**0.2193*** (+4.78%)
(training)	*bPref*	0.1974	**0.2186*** (+10.74%)
	MAP	0.1500	**0.1681*** (+12.07%)
901-950	*R - prec*	0.2236	**0.2368*** (+5.90%)
(test)	*bPref*	0.2049	**0.2294*** (+11.96%)

It is shown that retrieval effectiveness over extracted post and comments achieves significant improvements for every measure over the full text of the page, of about a 5% for R-prec and a 10% for MAP and bPref. This experiment demonstrates that using only post and comments as indexing units not only improves the effectiveness of the retrieval but also keeps the indices smaller than when indexing the full text.

6 Conclusions and Future Work

The main objective of this paper was studying an approach to post and comment extraction on blog pages, and how a right extraction may affect depending tasks, such as retrieval.

The obtained results suggest that the followed approach was successful, as it can correctly extract 82% of the blog posts and 89% of its comments, as seen in section 4.

With the extracted post and comments, the retrieval results outperformed the retrieval over the full page text, suggesting that many superfluous text was eliminated and meaningful one was retained.

An interesting future work line on the extraction task should be investigating on CMS-independent text extraction, or systems trained by examples which can learn to identify text regions in a similar way as human eye does. We also plan to do a more exhaustive crowdsourcing [8] evaluation of the extraction process using Amazon Mechanical Turk [9] and our evaluation approach.

Acknowledgments: This work was funded by FEDER, *Ministerio de Ciencia e Innovación* and *Xunta de Galicia* under projects TIN2008-06566-C04-04 and 07SIN005206PR.

References

1. Voorhees, E.M., Harman, D.K.: TREC Experiment and Evaluation in Information Retrieval. MIT Press (2005)
2. Mishne, G., Glance, N.: Leave a reply: An analysis of weblog comments. In: Third annual workshop on the Weblogging ecosystem, Edinburgh, Scotland (2006)
3. Attardi, G., Simi, M.: Blog mining through opinionated words. In Voorhees, E.M., Buckland, L.P., eds.: TREC. Volume Special Publication 500-272., National Institute of Standards and Technology (NIST) (2006)
4. Macdonald, C., Ounis, I.: The TREC Blogs06 collection: Creating and analysing a blog test collection. DCS Technical Report, University of Glasgow (2006)
5. Apache Software Fundation: Lucene, http://lucene.apache.org/. (2010)
6. Jones, K.S., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments. Inf. Process. Manage. **36**(6) (2000) 779–808
7. Ounis, I., de Rijke, M., Macdonald, C., Mishne, G.A., Soboroff, I.: Overview of the TREC-2006 blog track. In: TREC 2006 Working Notes. (2006) 1527
8. Alonso, O., Rose, D.E., Stewart, B.: Crowdsourcing for relevance evaluation. SIGIR Forum **42**(2) (2008) 9–15
9. Amazon: Mechanical Turk, http://www.mturk.com/. (2010)