reSearch: Enhancing Information Retrieval with Images

Martin Hugh Goodfellow, Ela Hunt, Daniel McCafferty
University of Strathclyde
26 Richmond Street
Glasgow, Scotland
{martin.goodfellow, ela.hunt, dmccaffe}@cis.strath.ac.uk

ABSTRACT

Combining image and text search is an open research question. The main issues are what technologies to base this solution on, and what measures of relevance to employ. Our reSearch prototype mashes up papers indexed using information retrieval techniques (Terrier) with Google image search for faces and Google book search. The user can interactively employ query expansion with additional terms suggested by Terrier, and use those terms to expand both the text and image search. We test this solution with a selection of recent publications and queries concerning people engaged in research. We report on the effectiveness of this solution. It seems that the combination works to a large extent, as testified by our observations.

1. INTRODUCTION

Mashups combine data from various internet sources, or from local data repositories and the internet. The current state of Internet mashups can be seen at *ProgrammableWeb.com*. We briefly discuss some of the mashups available in different areas. In this introduction we also give a brief overview of DBLife [5] and CompleteSearch [4] as this is the most relevant work in our area of interest. Beside those, we discuss the work of Alonso and Baeza-Yates [1] who demonstrate the integration of data from different search engines, and some underlying database management techniques which support mashups [13]. Example mashups in other areas include biology, see BioXMash [7] which allows data from an XML file repository to be integrated with a genome map. Our work and [1] are general frameworks, whereas DBLife and CompleteSearch combine only very specific information for a specific purpose. To make our work easier to evaluate, we limited our analysis to data of relevance from our community, i.e. a combination of an earlier version of DBLife and recent DBLP CompleteSearch.

There is an abundance of work on integration of information retrieval and database technologies to perform data integration. Vancea et. al. [13] propose a generic proxy mechanism to perform integration, inside the database, of internal and external data. Information management systems are generally split into one of three categories: text retrieval systems, relational/object database systems, or semistructured/XML database systems. However, generally most systems contain a mixture of data from all three categories. A brief but informative survey of integration and inter-operating of these different types of information management systems can be found in [12]. Work by Norrie and Signer [10] introduces web-based server technologies to provide integration of printed and digital information. Norrie [9] has also performed work in integration approaches for Computer Integrated Manufacturing (CIM). This involves coordinating engineering tasks and exchanging data between the specialised tools.

DBLife [5] was conceived as 'A Community Information Management Platform for the Database Research Community'. It has been live on the Web for almost 4 years but does not seem to have been updated since the summer of 2008. DBLife monitored a few hundered data sources, and crawled many web pages daily. It provided a variety of services that exploit the generated entity-relationship graph, including a daily community newsletter, entity superhomepages (pages that aggregate all detected and inferred information about an entity), and community event tracking. DBLife attempted to provide an image of the author as well as links to relevant authors and topics. It was based on automatic methods, such as PageRank and TF-IDF to extract relevant information from it's sources and form links between entities. DBLife also harnessed the critical power of its users, in allowing them to vote out irrelevant images. Although the success of DBLife's image retrieval is not clear, it should be noted that the image retrieval feature has been removed, possibly due to inconsistent results. DBLife also formed links between entities, which allowed it to offer a facet-like interface, including related people, topics, service, publications, organisations and co-authors. Discussions with researchers included in the database indicate that the relevance of these links was sometimes poor. Limited personal experimentation has confirmed that 'superhomepage' links - especially those that link to related topics appear to be irrelevant.

CompleteSearch [4] develops the DBLife idea further, based on DBLP data. The CompleteSearch engine differs from traditional engines. It uses prefix search - all query words are always interpreted as prefixes, rather than full words. It has no search button - a search is initiated automatically after a short delay, whenever the user stops typing. It produces

submitted to USETIM (Using Search Engine Technology for Information Management) workshop at VLDB
ACM 000-0-00000-000-0/00/00.

two result lists - every search produces a list of matching documents, sorted by relevance, and a list of suggestions to narrow down the search. CompleteSearch is more proactive than traditional search engines, as it actively tries to suggest to the user ways of further refining their search. This proactive feature may also be a cause of irritation and confusion. Discussions with researchers show that they were surprised that when clicking on an author name in the list of documents, it delivered the author page, as normal, while clicking on the same author's name in the list of suggestions for narrowing down, produces just the subset of those, as filtered by the name on the left hand side. Although the interface clearly says what it will deliver, and the search semantics appear to be obvious, it is still confusing even to a computer scientist. To deliver the new semantics, CompleteSearch uses a faceted interface [14].

Our aim was to develop a generic desktop application similar to DBLife but applicable to any topic and including images. reSearch searches both local and external information. Local information consists of text documents which the user specifies and indexes using Terrier [11]. As external information we tested Google image search and Google book search. Terrier creates a local index of user specified document directories. The external information is collected by issuing web queries to the Google search API. We were interested in the quality of the results which we measured by assessing the relevance of the results.

Our CONTRIBUTIONS are as follows. We implemented reSearch which mashes up local and web search results. This allows users to collect information about a researcher or research topic and enhance it with image data. We evaluated the solution to determine the quality of the results and discovered that they were all of high relevance. We propose future research directions which will further enhance the quality of the results and increase usability.

The remainder of the paper is structured as follows. Section 2 describes reSearch. Section 3 presents our quality results and Section 4 presents an example of our system in operation. Finally, in Section 5 we conclude and present future directions.

2. RESEARCH

reSearch is a desktop search application which generates a mashup of local text query results and web search results. We first discuss the application architecture. We then move on to how it handles web queries. Then, we discuss the Terrier search engine and explain how the quality of results is improved by means of query expansion.

2.1 Application Architecture

Figure 1 shows the architecture of reSearch. reSearch uses Terrier to index a document collection or uses an existing Terrier index. Terrier can index, among other types, XML, .doc or pdf files. Terrier normally performs query expansion on the user's original query. The expanded query, gathered from Terrier, is then used to query Google for relevant images. The returned information is then collated and presented to the user. reSearch allows the user to refine the search by adding Terrier's suggested query expansion term(s), and the option to remove them, should the new search return no results.

2.2 Web Queries

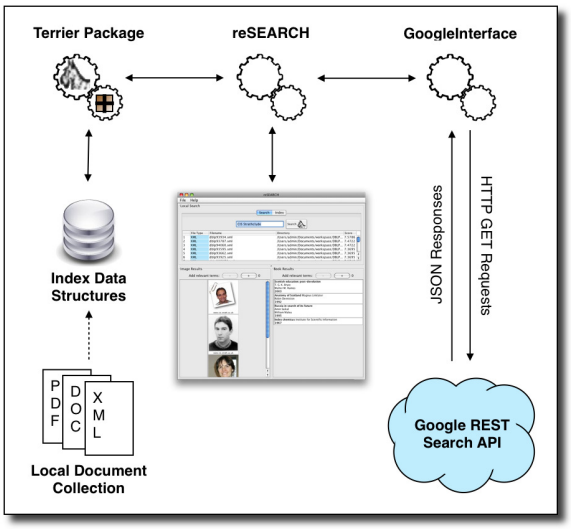

Figure 1: reSearch System Architecture

Google's search API was chosen for the image search due to it offering face detection, unlike other services considered. The search API can be used to query Google's many search services. The main API is intended for use in a Javascript environment. It also provides an interface for 'Flash and other Non-Javascript Environments', which was required for reSearch as it is a Java desktop application. The interface is based on a request-response system, where the request is a HTTP GET request containing a specific URL and the response is in a JavaScript Object Notation (JSON) format [1]. JSON is a ligthweight data-interchange format. In simple terms, the Google API is a web service provided through the HTTP protocol. A standard request is shown in Figure 2. Two arguments can be seen in the URL in this figure, '**?v=1.0**' which specifies the API version and '**&q=Google**' which is the query term itself. There are many other arguments that can be used in relation to an image search. For example, to indicate that only images of faces are required, we would append '**&imgtype=face**'. A full listing of all possible arguments can be found at the Google Code website. [2]

To further illustrate how the API is used, an example is provided in Figure 3. We show an HTTP GET request specifying two keywords; **CIS** and **Strathclyde**. The request also contains the argument '**&imgtype=faces**' so the results are restricted to images that are identified by Google as containing a face. The response that is shown is a single result from the four that are returned for this query. The response is in JSON format. Google returns a wide range of information for each result. reSearch only uses some of this information. The following attributes are used:

- **tbWidth & tbHeight** - the dimensions of Google's thumbnail of the image. reSearch uses this to determine if the image needs to be scaled to fit in the GUI.
- **tbUrl** - the URL of Google's thumbnail of the image. This is used to load the images into the GUI.

[1] http://www.json.org/
[2] http://code.google.com/

Standard Request

http://ajax.googleapis.com/ajax/services/search/images?v=1.0&q=Google

Figure 2: Google API Standard Request

Request

http://ajax.googleapis.com/ajax/services/search/images?v=1.0....&q=CIS%20Strathclyde&imgtype=faces

Response

{“responseData”: {“results”:[{“GsearchResultClass”:“GimageSearch”, “width”:“136”, “height”:“142”, “imageId”:“ZUyDyPjDSGRXIM”, “tbWidth”:“90”, “tbHeight”:“94”, “unescapedUrl”:“http://local.cis.strath.ac.uk/images/people/mv.png”, “url”:“http://local.cis.strath.ac.uk/images/people/ mv.png”, “visibleUrl”:“www.cis.strath.ac.uk”, “title”:“mv.png”, “titleNoFormatting”:“mv.png”, “originalContextUrl”:“http://www.cis.strath.ac.uk/cis/staff/index.php?uid\u003dmv”, “content”:“University of \u003cb\u003eStrathclyde \u003c/b\u003e Edit”, “contentNoFormatting”:“University of Strathclyde Edit”, ‘tbUrl”:“http://images.google.com/images?q\u003dtbn:ZUyDyPjDSGRXIM:local.cis.strath.ac.uk/images/people/mv.png”},....[snip]....“estimatedResultCount”:“293”...[snip]}

Figure 3: Example Google Search API Request and Response

Figure 4: Google Image API results for ’q=CIS Strathclyde’ and ’imgtype=face’

Figure 5: Search Results for "Gregory Grefenstette"

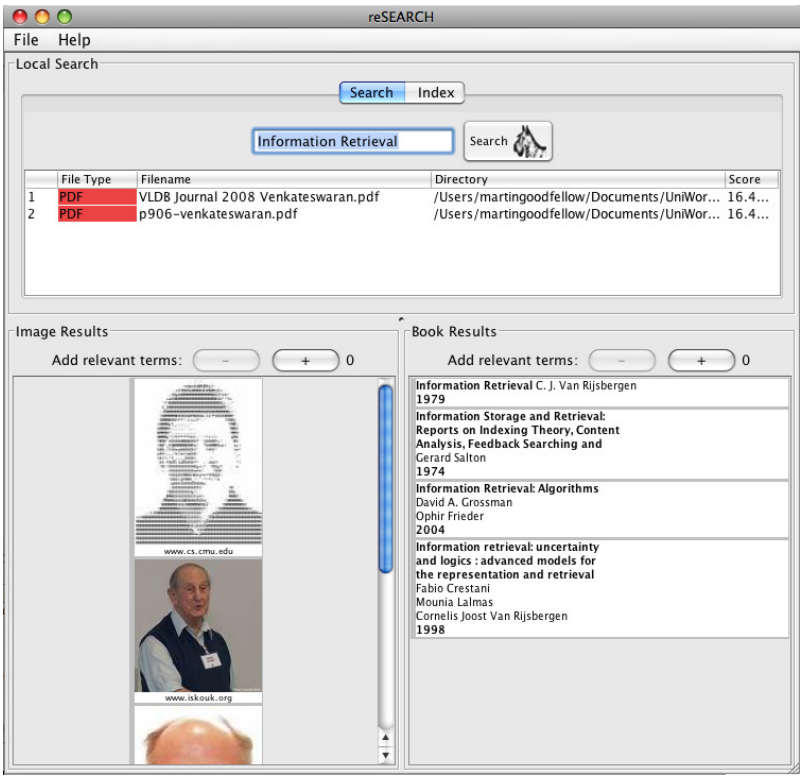

Figure 6: Search Results for "Information Retrieval"

- **originalContextUrl** - the URL of the image.
- **visibleUrl** - the URL of the root domain of the image.

2.3 Terrier

Terrier [11], Terabyte Retriever, is a scalable, high performance search engine that supports fast development of large-scale retrieval applications. It provides a test-bed for experimentation and research in Information Retrieval (IR).

Terrier is written in Java and offers a variety of features: a range of weighting models, alteration of document scores and a query language. It also supports various ways of word stemming and query expansion. It includes numerous weighting models, including BM25, TF-IDF, eight models from the Divergence from Randomness (DFR) framework and Ponte-Croft's language model [6]. It also allows the score of individual terms or retrieved documents to be modified. Its query language allows additional operations to be specified by the user on top of the normal probabilistic queries. Finally, Terrier includes automatic pseudo-relevance feedback, by providing query expansion (QE). QE used in reSearch was Bo1 [2, 8].

Terrier provides several document parsers. It can index HTML, plain text, Microsoft Word, Excel and Powerpoint, and Adobe PDF. It also allows users to add support for other document types by including a Document plugin which extracts the terms from the document. Before indexing, Terrier passes terms through a 'Term Pipeline', which performs term transformations. As with document parsers, Terrier provides various Term Pipeline plugins, including stopword removal, two variants of Porter's stemming algorithm and the snowball stemming algorithm. Users can include their own Term Pipeline plugins, similar to the Document plugins. The index used is a Block Index.

The index consists of four main data structures:

- a lexicon containing information for each term;
- an inverted index containing the postings lists of each term;
- a document index containing information for each document;
- a direct index containing the terms that appear in each document.

2.4 Query Expansion

An integral feature of reSearch is the refinement of external results. This refinement is achieved through reformulation of the user's original query, based on Terrier QE. When a user performs a search, Terrier determines additional terms that are relevant to the local document results, as described in Section 2.1. The user can then add these terms in order of relevance (determined by Terrier) to the original query, to obtain new refined results. In most searches it is possible to add too many terms to the search, in which case no results will be returned. To allow greater control over the results, users can add terms as well as remove them. Although this interface supports a reasonable level of control and feedback, there remain opportunities for improvement, as discussed in the future work section.

3. RESULTS

reSearch was user tested and evaluated using a questionnaire and then directly by the authors. As subjects we used four students from the Computer and Information Sciences department. Each user was asked to perform an author and research area search task. The author to be searched for was one of six options from the department, for whom documents were part of the local index. The research area to be searched for was one of the selected individuals' research areas. The users were asked to assess the quality of the results returned both before and after query expansion. The quality was recorded on a five-point Likert scale where 1 was 'completely irrelevant' and 5 was 'completely relevant' and for improved results after the query expansion was 1 for 'no improvement' and 5 for 'large improvement'.

Judging by the results from our questionnaire, we claim reSearch was successful. All outcomes were at least a 4 on the Likert scale before query expansion and at least 3 on the Likert scale after query expansion. The reason for this could be that the initial results were somewhat saturated with relevant results, leaving very little to be improved upon. Although query expansion has not appeared to result in huge improvements, it was not detrimental to relevance.

The authors tested the system using this workshop's (USETIM) program committee chair names along with some of their research areas. The queries and results can be seen in Table 1. Table 2 shows the results after 1 and 2 query expansion terms have been added to the search terms. We have defined relevance for text documents as any document that contains the person as an author or discusses the research area being searched for. For images relevance is defined as an image of the person or relating to the research area being searched for. However, pictures from a page mentioning a person or research area are also deemed relevant, e.g. colleagues or co-authors of the person being searched for. As can be seen from the results, our solution was a success. This is due to all results having a high degree of relevance.

David Simmen returns two images of who we are looking for but also two other David Simmens. In our results we haven't defined these as being relevant, however, with respect to the query terms they are in fact relevant.

Query	Text		Images	
	Relevant	Irrelevant	Relevant	Irrelevant
Gregory Grefenstette	6/6	0/6	4/4	0/4
Wolfgang Nejdl	23/23	0/23	4/4	0/4
David Simmen	5/5	0/5	2/4	2/4
Mashup	6/6	0/6	4/4	0/4
Image Retrieval	6/6	0/6	4/4	0/4

Table 1: Query Result Quality

4. EXAMPLE

The following shows an example of the application in operation. The first example to be shown is a search for 'Gregory Grefenstette' and the second shows a search for 'Information Retrieval'.

The search results for 'Gregory Grefenstette' can be seen in Figure 5. In the diagram, the files listed at the top of the screen are local files which match the query. The bottom

Query	Text		Images	
	+1	+2	+1	+2
Gregory Grefenstette	6/6	6/6	4/4	4/4
Wolfgang Nejdl	23/23	23/23	4/4	4/4
David Simmen	5/5	5/5	4/4	4/4
Mashup	6/6	3/3	4/4	4/4
Image Retrieval	6/6	6/6	4/4	4/4

Table 2: Query Expansion Results Quality

left column shows results from Google image search and the bottom right column shows results from Google books. The results from the second example can be seen in Figure 6.

5. DISCUSSION, CONCLUSIONS AND FUTURE WORK

reSearch aimed to repair the image search deficiencies observed in DBLife. We believe that our solution improves on the image relevance. This is due to two factors. The main improvement comes from the use of query expansion in Terrier. Papers indexed locally provide relevant query expansion terms and those terms are used to query the images indexed by Google. The second source of improvement is the fact that the Google image search API allows us to query for faces specifically. It needs to be noted though, that although we attempted to query for book covers in Google books, we found some technical problems which did not allow us to add book covers to our interface, although this should have been possible.

reSearch is a desktop search application. It allows users to search for a specific researcher or research area and collates information from various sources to produce more informative results than from a single source. It incorporates stemming, weighting models and query expansion into its design. Initial testing results have been positive and have introduced directions for further research. The solution is generic, and given an appropriate image query API, could be used for any other mashup, based on text and images.

Future work will attempt to improve on the query expansion and incorporate more data sources in the results. For QE we have identified two avenues. The first method to be considered is the use of synonyms. Using a database such as WordNet [3] synonyms could be added to searches to broaden the results. A second strategy for the improvement of query expansion is the use of an ontology (taxonomy) as in [3]. Using an ontology exploits each word's 'parents' and/or 'children' (e.g. cat → 'mammal' (parent) 'American Shorthair' (child)). This method of query expansion would require access to a large, up-to-date ontology to be successful. One such ontology is Wikipedia which organises all of it's articles into hierarchical categories. Access to Wikipedia could be provided through an ongoing Wikipedia Labs project [4] which aims to provide access to the ontology, and already provides a Web Services Description Language (WSDL) document that describes the service. Any combination of these two approaches and the current approach to query expansion could be used to improve relevance in local and web-based results.

Currently, when query expansion terms are added, the most relevant ones are added first as determined by Terrier. To give the user more control, we could allow the users to select which query expansion terms to add to the query, using an improved interface showing those terms interactively.

[3]http://wordnet.princeton.edu/

[4]http://wikipedia-lab.org/en/index.php/Wikipedia_API

6. ACKNOWLEDGMENTS

We would like to thank the Terrier team at the University of Glasgow, particularly Craig McDonald.

7. REFERENCES

[1] O. Alonso and R. A. Baeza-Yates. Integration of Search Engines with User Interfaces. In *WWW Posters*, 2001.

[2] G. Amati. *Probabilistic Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Department of Computer Science, University of Glasgow, 2003.

[3] N. Anwar and E. Hunt. Improved Data Retrieval from TreeBASE via Taxonomic and Linguistic Data Enrichment. *BMC Evolutionary Biology*, 9:93, 2009. doi:10.1186/1471-2148-9-93.

[4] H. Bast and I. Weber. The CompleteSearch Engine: Interactive, Efficient, and Towards IR & DB Integration. In *CIDR*, pages 88–95. www.crdrdb.org, 2007.

[5] P. DeRose et al. DBLife: A Community Information Management Platform for the Database Research Community (Demo). In *CIDR*, pages 169–172. www.crdrdb.org, 2007.

[6] B. He and I. Ounis. Term Frequency Normalisation Tuning for BM25 and DFR Models. In *ECIR*, pages 200–214, 2005.

[7] E. Hunt, J. Jakubowska, C. Bösinger, and M. C. Norrie. Defining Mapping Mashups with BioXMash. *J. Integrative Bioinformatics*, 4(3), 2007.

[8] C. Macdonald, B. He, V. Plachouras, and I. Ounis. University of Glasgow at TREC 2005: Experiments in Terabyte and Enterprise Tracks with Terrier. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, November 2005.

[9] M. C. Norrie. Integration Approaches for CIM. In *SIGMOD Conference*, page 470, 1995.

[10] M. C. Norrie and B. Signer. Web-based Integration of Printed and Digital Information. In *DIWeb*, pages 71–85, 2002.

[11] I. Ounis et al. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, August 2006.

[12] S. Raghavan and H. Garcia-Molina. Integrating Diverse Information Management Systems: A Brief Survey. *IEEE Data Eng. Bull.*, 24(4):44–52, 2001.

[13] A. Vancea, M. Grossniklaus, and M. C. Norrie. Generic Proxies - Supporting Data Integration Inside the Database. In *OTM Workshops (1)*, pages 5–6, 2007.

[14] R. Villa, N. Gildea, and J. M. Jose. FacetBrowser: A User Interface for Complex Search Tasks. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 489–498, New York, NY, USA, 2008. ACM.