SubSift: a novel application of the vector space model to support the academic research process

Simon Price SIMON.PRICE@BRISTOL.AC.UK
Institute for Learning and Research Technology,
University of Bristol,
Bristol BS8 1HH, UK

Peter A. Flach PETER.FLACH@BRISTOL.AC.UK
Sebastian Spiegler SPIEGLER@CS.BRIS.AC.UK
Department of Computer Science,
University of Bristol,
Bristol BS8 1UB, UK

Editors: Tom Diethe, Nello Cristianini, John Shawe-Taylor

Abstract

SubSift matches submitted conference or journal papers to potential peer reviewers based on the similarity between the paper's abstract and the reviewer's publications as found in online bibliographic databases such as Google Scholar. Using concepts from information retrieval including a bag-of-words representation and cosine similarity, the SubSift tools were originally created to streamline the peer review process for the ACM SIGKDD'09 data mining conference. This paper describes how these tools were subsequently developed and deployed in the form of web services designed to support not only peer review but also personalised data discovery and mashups. SubSift has already been used by several major data mining conferences and interesting applications in other fields are now emerging.

Keywords: vector space model, cosine similarity, web services

1. Introduction

In this paper we describe how SubSift, a novel application of concepts from information retrieval to problems in conference paper peer review, has given rise to a more general, re-usable set of web services with interesting applications in other domains, for instance profiling research groups and organisations. We begin in this section with an overview of the development of the original software and its subsequent redevelopment and deployment.

1.1. Motivation and implementation of the original SubSift software

Peer review of written works is an essential pillar of the academic process, providing the central quality control and feedback mechanism for submissions to conferences, journals and funding bodies across a wide range of disciplines. However, from the perspective of a busy conference chair, journal editor or funding manager, identifying the most appropriate reviewer for a given submission is a non-trivial and time-consuming task. Effective assignment, first and foremost, requires a good match to be made between the subject of the submission and the corresponding expertise of reviewers drawn from a, sometimes large, pool of potential reviewers. In the case of conferences, a recent trend transfers much of this allocation work to the reviewers themselves, giving them access to the full range of submissions and asking them to bid on submissions they would like to review. Their bids are then compared to inform the allocation decisions of the programme committee chair. It was the challenge of just such a bidding process that motivated the second author of this paper, programme co-chair for *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2009* (SIGKDD'09), to develop the original SubSift tools – not as research *per se*, but as a practical implementation of theory (Flach et al., 2009).

SubSift, short for *submission sifting*, addressed three specific problems: (i) matching submissions to reviewers; (ii) ranking potential assignments; (iii) allocating papers to reviewers. In the first step, each reviewer's bids were initialised based on textual similarity between the paper's abstract and the reviewer's publication titles, as listed in the DBLP bibliographic database. In the second step, each of the 199 reviewers was sent an email containing a link to a personalised SubSift generated web page listing details of all 537 papers ordered by initial bid allocation and similarity to their own published works. Guided by this personalised perspective, plus the usual titles and abstracts, reviewers affirmed or revised their bids. In the final step, after all reviewer bids were submitted the programme co-chairs were able to consult, for any paper, a similarity ranked list of reviewers to assist them in allocating papers with either too few or too many bids[1].

To quantitatively evaluate SubSift's performance, we compared SubSift's initial bids against the revised final bids made by SIGKDD'09 reviewers. Disregarding the level of the bid, we calculated: *precision* as the proportion of non-zero actual bids among the non-zero initial bids; *recall* as the proportion of non-zero initial bids among the non-zero actual bids; and *F-measure* as the harmonic mean of precision and recall. Over all reviewers the median F-measure is 72.7%; median precision 88.2%; and median recall 80.0%[2]. These results, plus qualitative feedback from reviewers, were encouraging and demonstrated that there is considerable scope for automated support during the bidding process.

1.2. Subsequent redevelopment and deployment as web services

The application of the original SubSift tools for SIGKDD'09 prompted requests from programme committee (PC) chairs of other data mining conferences to re-use the software in other settings. This in turn prompted a successful bid to the UK Joint Information Services Council (JISC) for a software development project to repackage SubSift tools as *SubSift Services*, a collection of lightweight web services, usable individually or in mashups, to support: matching reviewers with submissions, ranking potential assignments, and allocating papers to reviewers.

One the key goals of the SubSift Services project is to share and enable re-use of the software in different settings, and in a sustainable way. The decision to repackage and host SubSift as web services was in direct pursuit of this goal. Apart from the obvious advantage of prospective users being able to use the software without having to install it,

1. Beyond the scope of this article, the KDD'09 tools also assisted in the constraint satisfaction problem of allocating papers given reviewer bids and conflicts of interest, and of calibrating reviewer's scoring.
2. We use the median because of the skew of the distribution.

the web services model has a number of other benefits, including the ability for users to: make their own application-specific customisations by integrating the services with mashup tools like Yahoo Pipes; select other data sources, such as Citeseer, Google Scholar, eprints, news and blogs; and, importantly, integrate the software with their own research outputs without having to modify code. However, in recognition that web services will not suit all applications, the SubSift Services project has released the software as Open Source on Google Code. Further details of SubSift are available on the project website[3].

To demonstrate SubSift Services the project produced an example workflow as a series of web forms that: (i) accepts a list of author names, looks up these names on the DBLP Computer Science Bibliography[4] and suggests author pages which, after disambiguation, are used to profile the reviewers based on the text of these pages; (ii) allows the user to upload a CSV file containing all the submitted paper abstracts, which form the basis of the abstract profiles; and (iii) compares reviewer profiles to paper abstract profiles, producing downloadable web pages with ranked lists of papers per reviewer, and ranked lists of reviewers per paper. Although only a demonstrator, this workflow has now been used to support several major data mining conferences.

The remainder of this paper begins with a review of relevant background topics, followed by a tour of the SubSift Services API, and a look at some interesting applications in other areas, before rounding up with future work and concluding remarks.

2. Background

The theoretical basis for the text matching component of SubSift, and of SubSift Services, is the well known *vector space model* from information retrieval (Salton et al., 1975). SubSift Services software also draws on the *representational state transfer* (REST) design pattern for web services (Fielding, 2000). In this section we give a brief overview of each of these topics. Readers already familiar with them can safely skip to the next section.

2.1. Vector Space Model

The canonical task in information retrieval is, given a query in the form of a list of words (terms), rank a set of text documents D in order of their similarity to the query. The vector space model is a common approach to solving this problem. Each document $d \in D$ is represented as the multiset of terms (bag-of-words) occurring in that document. The set of distinct terms in D, vocabulary V, defines a vector space with dimensionality $|V|$ and thus each document d is represented as a vector $\boldsymbol{d}$ in this space. The query q can also be represented as a vector $\boldsymbol{q}$ in this space, assuming it shares vocabulary V. The query and a document are considered similar if the angle θ between their vectors is small. The angle can be conveniently captured by its cosine, which is equal to the dot product of the vectors scaled to unit length, giving rise to the *cosine similarity*, s.

3. SubSift – `http://subsift.ilrt.bris.ac.uk`
4. DBLP Computer Science Bibliography – `http://dblp.uni-trier.de/`

$$s(\boldsymbol{q}, \boldsymbol{d}) = \cos(\theta) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}}{||\boldsymbol{q}|| \cdot ||\boldsymbol{d}||}$$

However, if raw term counts are used in vectors $\boldsymbol{q}$ and $\boldsymbol{d}$ then similarity will be biased in favour of long documents and will treat all terms as equally important. The *term frequency – inverse document frequency* (tf-idf) weighting scheme compensates for this by normalising term counts within a document by the total number of terms in that document, and by penalising terms which occur in many documents. More formally, *term frequency* tf_{ij} of term t_i in the document d_j, and *inverse document frequency* idf_i of term t_i are defined as

$$\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}, \qquad \mathrm{idf}_i = \log_2\left(\frac{|D|}{\mathrm{df}_i}\right), \qquad \text{tf-idf}_{ij} = \mathrm{tf}_{ij} \times \mathrm{idf}_j$$

where *term count* n_{ij} is the number of times term t_i occurs in the document d_j, *document frequency* df_i of term t_i is the number of documents in D in which term t_i occurs.

In SubSift, instead of comparing a single query against a set of documents, we pair-wise compare every document in one collection D_1 (e.g. abstracts) with every document in another collection D_2 (e.g. reviewer bibliographies) to produce a ranked list for each document. To capture the overall importance of each term across the combined collections, df_i, and hence tf-idf_i, values are calculated over the union of both collections, $D_1 \cup D_2$.

2.2. Representational State Transfer (REST)

REST is an easily understood design pattern for web services, based around the ubiquitous HTTP protocol and its familiar vocabulary of URIs, media types, requests and responses (Fielding, 2000; Fielding and Taylor, 2002). Recently, REST web services have become popular as the web API behind numerous Web 2.0 sites, including Twitter, Flickr and Facebook. Like conventional websites, *RESTful* websites offer a stateless, cacheable, layered and uniform client-server interface. However, unlike conventional sites, which are designed to render human-readable data as HTML pages to be viewed in a browser, RESTful sites serve data in formats such as XML, JSON and CSV, that may be readily consumed by arbitrary applications. Furthermore, in the same way that HTML forms on conventional web pages can be used to submit data from the client to the server for storage and processing, arbitrary RESTful applications can use exactly the same protocols to achieve the same end. Also, the usual HTTP authentication and authorisation mechanisms can be used to control access to specific services and resources.

The intuition behind REST is that URIs are used to represent resources and that HTTP request methods are used to specify fundamental operations on those resources. Additional operations are specified by adding verbs into the URIs. The most widely used HTTP request method, `GET`, is invoked every time a web browser requests a URI. This usually returns a web page in the form of an HTTP response which the browser displays. However, `GET` is only one of several HTTP request methods; the five that are most significant for REST web services are summarised in the table below.

HTTP Method	**Usage in REST**	**Changes Resource**
`GET`	*show* and *list* operations	no
`HEAD`	*exists* operations to check if a resource exists	no
`POST`	*create* and *compute* operations	yes
`PUT`	*update* and *recompute* operations	yes
`DELETE`	*destroy* operations	yes

Of these, the most familiar to HTML authors are `GET` and `POST` - as used in hyperlinks (e.g. `<a href="http://www.example.com">...</a>`) and HTML forms (e.g. `<form method="POST" action="...">...</form>`). As is the case with HTML forms, pairs of attribute-value parameters can be supplied with the HTTP request in the usual way.

3. SubSift REST API

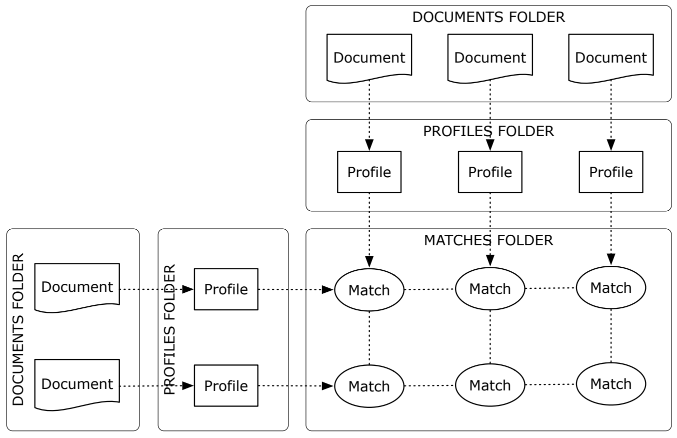
The SubSift REST API is organised around a series of *folders* into which data *items* are stored. This organisation is modelled on the familiar filing system concept of folders and files. The three main folder types are documents, profiles and matches. In SubSift, a document is a piece of text to be profiled and matched. A document will usually be the text from some external source such as the text of a web page or a conference paper abstract. A profile is a summary representation of the features of a single document, with respect to the other documents in the same documents folder. One usage of profiles in SubSift is to obtain a list of distinguishing terms, or keywords, for a document – for example, automatically extracting keywords from abstracts of papers submitted to a conference. Another usage is for two profile folders to be compared against each other to produce a matches folder. A matches folder is created by analysing every pairing of profile items drawn from the two profiles folders. Each match item records the tf-idf cosine similarity, and various related statistics, of a single profile from the first profiles folder against every profile from the second profiles folder. A typical usage of such a comparison is to match submitted conference abstracts with the bibliography pages of programme committee members in order to rank potential reviewers for each paper and visa versa. This usage is depicted in Figure 1.

A wide range of API methods make intermediate data and metadata available for every type of folder and item. For example, the relative contribution of each term towards a particular similarity calculation can be retrieved or the entire similarity matrix exported. For ease of integration, there is flexibility in both input and output. Document text may be added per item or in bulk or by supplying a list of URLs to be fetched asynchronously by SubSift's harvester robot. The API methods can return data in the following representational formats: CSV, JSON, XML, YAML, and Prolog terms.

4. Applications in Other Areas

In this section we outline an exploratory investigation of a potential application of SubSift to social network analysis within an organisation, and briefly mention other uses that have emerged. Figure 2 shows a fragment of a dendrogram produced in Matlab by clustering a similarity matrix calculated by SubSift, pairwise matching ILRT staff homepages. The numbers to the right of the labels represent the staff member's actual ILRT project group

Figure 1: A SubSift REST API controlled sequence of transformations from a pair of document folders (e.g. abstracts versus PC members) through to a folder of matching statistics. Useful data and metadata can be obtained at each step in the process via API methods.



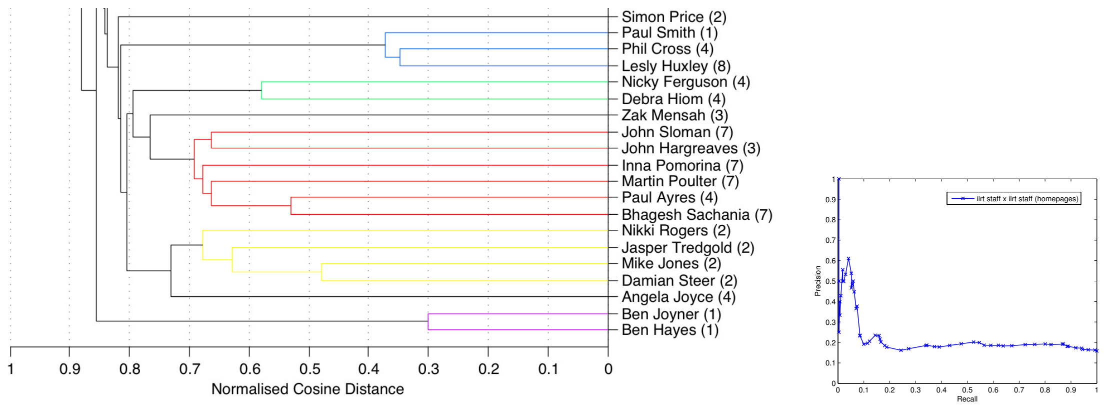


Figure 2: Using SubSift similarity data to cluster ILRT staff homepages.

membership, and the chart on the right shows pairwise precision and recall for thresholds at each node in the dendrogram. While the precision-recall plot suggests no single ideal threshold, some project groups are rediscovered by the clustering but, more interestingly, the clustering also reveals older project groupings and co-authorships prior to organisational restructuring at ILRT. Potential applications might include locating research bid partners across an organisation or identifying informal structures within an organisation.

Other applications include using SubSift to produce a “profiling” list of keywords associated with a person or a group of people, followed by transformation into Friend of a Friend (FOAF) semantic web metadata or, more visually, into tag cloud diagrams. Using the ranked lists from matches allows the creation of network diagrams visualising social networks and, along similar lines, following friends’ interests on Twitter. SubSift is also being used to produce research datasets for machine learning experiments.

5. Future Work

Although SubSift Services is neither pure research nor pure IT, there are potential developments in both these areas. On the research administration front, SubSift is a useful service for constructing, manipulating and publishing document-centric data sets, and the REST model itself is a promising way to persist and share algorithms within the research community. SubSift, by its very nature, has potential for integration into a wide range of applications ranging far beyond the original peer review domain.

We are now building on the SubSift tools to develop ExaMiner, an application for mining and mapping a university’s research landscape, beginning initially with the University of Bristol but with the ultimate aim of being able to map any research-centric institution. Specifically, the ExaMiner software will: (i) generate automatic profiles of researchers and research groups from university web pages; (ii) allow visual navigation of these profiles, as well as navigation by free-text query; (iii) provide social networking components, including the ability to store a personal profile and a network of interesting contacts, events and connections; (iv) notify a user of new resources in their sphere of interest; and (v) improve performance over time by responding to user feedback. Possible users include academics, research managers, current and prospective students, and the media. The ExaMiner project has been funded by the University of Bristol and it is hoped that a system based on this prototype will improve the discoverability of its research within the institution, the community, the media, prospective students, and the government.

6. Concluding Remarks

The implementation of SubSift Services as a repackaged version of the text matching functionality of the SIGKDD’09 software has created a more general purpose resource with potential applications in areas outside the specific domain of peer review. A similar approach may be successful in publishing the functionality of other research-produced applications.

Acknowledgments

We would like to thank conference PC chairs Bart Goethals (SDM'10), Qiang Yang (KDD'10), Mohammed Zaki (PAKDD'10) and Geoff Webb (ICDM'10) for trialing SubSift; the SubSift Services project team members Nicola Rogers and Chris Bailey from ILRT; and other developers of the original software, Bruno Golénia from Department of Computer Science, John Guiver, Ralf Herbrich and Thore Graepel from Microsoft Research in Cambridge, and Mohammed Zaki from Rensselaer Polytechnic Institute.

References

Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2:115–150, May 2002. ISSN 1533-5399. doi: http://doi.acm.org/10.1145/514183.514185.

Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. URL `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

Peter A. Flach, Sebastian Spiegler, Bruno Golénia, Simon Price, John Guiver Ralf, Herbrich Thore Graepel, and Mohammed J. Zaki. Novel tools to streamline the conference review process: Experiences from SIGKDD'09. *SIGKDD Explorations*, 11(2):63–67, December 2009. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1809400.1809413.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/361219.361220.