GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia

Shlomo Geva

Faculty of IT, Queensland University of Technology
Brisbane, Australia
s.geva@qut.edu.au

Abstract. The INEX 2007 evaluation was based on the Wikipedia collection. In this paper we describe some modifications to the GPX search engine and the approach taken in the Ad-hoc and the Link-the-Wiki tracks. In earlier version of GPX scores were recursively propagated from text containing nodes, through ancestors, all the way to the document root of the XML tree. In this paper we describe a simplification whereby the score of each node is computed directly, doing away with the score propagation mechanism. Results indicate slightly improved performance. The GPX search engine was used in the Link-the-Wiki track to identify prospective incoming links to new Wikipedia pages. We also describe a simple and efficient approach to the identification of prospective outgoing links in new Wikipedia pages. We present and discuss evaluation results.

Keywords: GPX, INEX, XML, Information Retrieval, Link Discovery.

1 Introduction

In this paper we describe the submission of QUT and the GPX search engine in the Ad Hoc and Link the Wiki tracks of INEX 2007. Both the Ad Hoc track and the Link the Wiki track are described in some detail elsewhere in these proceedings and so we restrict our brief introduction to some background that is related to our specific approach. Having reported on GPX over several years now, we refer the reader to previous proceedings of INEX [2,3] for a comprehensive general overview of the Ad Hoc track and focus our attention more on the Link the Wiki task.

The Wikipedia is a free online document repository written collaboratively by wiki contributors around the world. The INEX collection is composed of about 660,000 articles and it offers many attractive features as a corpus for information retrieval tasks. The INEX Wikipedia collection has been converted from its original wiki-markup text into XML [1]. That collection is composed of a set of XML files where each file corresponds to an online article in Wikipedia. Search as well as retrieval could benefit from rich semantic information in the XML Wikipedia collection, where it exists. However, the XML semantics is not rich and relates mostly to structure. This is arguably a deficiency that hinders taking advantage of the XML technology which offers semantic annotation capacity.

N. Fuhr et al. (Eds.): INEX 2007, LNCS 4862, pp. 404–416, 2008.

The semi-structured format provided by the XML-based collection offers a useful property for the evaluation of various semi-structured retrieval techniques. In 2007 we have made a modification to GPX [3] ranking algorithm and this is discussed in the following sections. We omit further literature review of link discovery since this can be found in the Link-the-Wiki track overview paper, elsewhere in theses proceedings, but references [5-12] herein provide good coverage.

There are essentially two immediate approaches that come to mind when setting out to link the Wikipedia. One is to perform Link Analysis on the existing connectivity graph in the wikipedia, and the other is to look for semantic connections between documents. Link analysis relies on the already existing semantic links (previously generated by users) while as context analysis does not assume semantic links, but attempts to discover such from scratch. As it turns out, these approaches are very effective and both were in fact represented in the Link the Wiki track in 2007.

The remainder of this paper is organized as follows. In sections 2, 3 and 4 we report and analyse the performance of GPX in the various tasks of the Ad Hoc track. The Link the Wiki task is discussed in sections 5 and 6. We conclude in section 7.

2 The GPX Search Engine

In this section we provide a very brief description of GPX. The reader is referred to earlier papers on GPX in INEX previous proceedings [3] for a more complete description.

2.1 GPX Inverted List Representation

The GPX search engine is based on XPath inverted lists. For each term in the collection we maintain an inverted list of XPath specifications. This includes the file name, the absolute XPath identifying a specific XML element, and the term position within the element. The actual data structure is designed for efficient storage and retrieval of the inverted lists which are considerably less concise by comparison with basic text retrieval inverted lists.

The GPX search engine is using a relational database implementation (Apache Derby) to implement an inverted list data structure. It is a compromise solution which provides the convenience of a DBMS at the cost of somewhat reduced performance compared to what might otherwise be possible with an optimized file structure.

Consider the XPath:

/article[1]/bdy[1]/sec[5]/p[3]

This could be represented by two expressions, a Tag-set and an Index-set:

Tag-set: **article/bdy/sec/p**

Index-Set: **1/1/5/3**

The original XPath can be reconstructed from the tag-set and the index-set. There are over 48,000 unique tag-sets, and about 500,000 unique index-sets in the collection. We assign to each tag set and each index-set a hash code and create auxiliary

database tables mapping the hash-codes to the corresponding tag-set and index-set entries. These hash tables are small enough to be held in memory and so decoding is efficient.

The GPX database tables are then:

Term-Context = { **Term-ID**, File-ID, XPath-Tag-ID, XPath-IDX-ID, Position }
Terms = { **Term**, Term-ID }
Files = { **File-Name**, File-ID }
TagSet = { **XPath-Tag-ID**, Tag-Set }
IndexSet = { **XPath-IDX-ID**, Index-Set }
XPathSize = { **XPath-ID**, Node-Size }

Given a search term the database can be efficiently accessed to obtain an inverted list containing the context of all instances where the term is used (identified by File Name, full XPath, and term position). Having retrieved a set of inverted lists, one for each term in the query, the lists are merged so as to keep count of query terms in each node and also keeping the term positions. Stop words are actually indexed, but too frequent terms are ignored by applying a run-time stop-word frequency threshold of 300,000. We also used plural/singular expansion of query terms. We have found that - on average - the use of a Porter stemmer is not adding to system performance and so it was not used.

Having collected all the nodes that contain at least one query term, the system proceeds to compute node scores. Calculation of node relevance score from its content is based on a variation of TF-IDF. We used the inverse collection frequency of terms rather than the inverse document frequency (TF-ICF). The score is then moderated by a step function of the number of unique terms contained within the node. The more unique terms the higher the score. The score is further moderated by the proximity within which the terms are found. Additionally, the scores of all article nodes that contained query terms in the *name* node were further increased. All this can be calculated with the information in the inverted lists.

2.2 Calculation of Text Nodes Score

GPX 2007 deviates significantly from earlier versions with respect to the way that ancestor node scores are calculated. For clarity we shall refer to GPX-2007 to denote the current system and GPX to denote the older system. In the earlier version GPX computed node scores on the basis of direct text content (having a text node in the DOM model) and then the scores were propagated upwards in the XML tree. GPX accumulated all children node scores for a parent and reduced the score by a decay factor (typically about 0.7) to account for reduced specificity as one moved upwards in the XML tree. In GPX 2007 the scores are computed directly from the node text content, direct, or indirect. That means that any node is scored by the text it contains regardless of whether it has a direct text node in the DOM representation – all the text in the node and its descendents is used.

Naturally, nodes closer to the root could receive a higher score on account of more query terms in descendent nodes. A common variation to TF-IDF is to normalise the score by taking into account the document size. The motivation is to account for the increased probability of finding query terms in larger documents and hence biasing

the selection towards larger documents. The motivation here is similar with a slight twist. Node normalisation in the XML score calculation is motivated by the need to compensate for the reduced specificity of larger nodes. We are aiming for focused retrieval and look for nodes of "just the right size" (whatever that may be.) Node normalisation introduces a penalty in a parent node that contains large amounts of irrelevant text in descendent nodes and which do not contribute towards an increased score. However, when two nodes have a similar size but contain different amount of relevant text then the more relevant node will score higher.

But there is another twist here. We also know that nodes that are too small are unlikely to satisfy a user information need (except perhaps in factoid type QA). At least with the Wikipedia we know that the most common element selected by assessors is a paragraph (or passage). Very small passages are not common in the qrels of past experiments. Therefore, we do not want to normalise the scores of too small nodes thereby unduly increasing their score relative to otherwise similarly scoring nodes which are somewhat larger. Node scores are normalised by dividing the raw score by the node size (measured as the number terms), but all nodes with size of below 75 terms are normalised by 75.

Equation 1: Calculation of S, node size for normalisation

$$S = \begin{cases} NodeSize & if\ (NodeSize > 75) \\ 75 & if\ (NodeSize \leq 75) \end{cases} \quad (1)$$

The value of S, the node size for the purpose of normalization, is thus equal to 75 for nodes smaller than 75 terms, but taken as the actual node size for nodes with more terms.

This heuristic is convenient in the XML case because when breaking ties in node selection (focused retrieval) we prefer the ancestor to the descendant when the scores are equal. This means that we prefer parent nodes as long as the parent is larger than the descendant and below 75 terms in size. For example, this means that a very deep XML branch with no breadth will be collapsed to an ancestor of up to size 75 terms (if such exists). So in summary, node size normalisation is biasing the selection towards passages of 75 terms, both from above and from below. We experimented with other values for node size from 50 to 150 with little difference in results.

Since GPX 2007 computes node scores over much larger text segments it is necessary to take account of term proximity. The intuition is that we should award higher scores to nodes in which search terms are found in closer proximity to each other. In earlier versions of GPX this was not critical since node scores were computed at text nodes and these were typically paragraphs, titles, captions, and other such relatively small nodes. A proximity function was defined and incorporated into the score calculation.

Equation 2: Calculation of P, node terms proximity score

$$\mathrm{Pr} = 10 \sum_{i=1}^{n} \exp\left(-\left(\frac{p_i - p_{i+1} + 1}{5} \right)^2 \right) \quad (2)$$

Here terms are processed in the order in which they appear in the text node. P_i is the position of term i in the text node. Note that for immediately successive terms Pr=10. This is a Gaussian function with a maximum value of 10 and decaying exponentially with increased term distance between successive terms. The function is depicted in Figure 1, for two terms separation. Note that in practice, a table lookup is more efficient than the numerical calculation.

So finally we have the following score calculation:

Equation 3: Calculation of element relevance score from its content

$$L = \frac{\mathrm{Pr}}{S} K^{\,\mathrm{n}-1} \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{t_{\mathrm{i}}}{f_{\mathrm{i}}} \tag{3}$$

Here n is the count of unique query terms contained within the element, and K is a small integer (we used K=5). The term K^{n-1} is a step function which scales up the score of elements having multiple distinct query terms. This heuristic of rewarding the appearance of multiple distinct terms can conversely be viewed as taking more strongly into account the absence of query terms in a document. Here it is done by rewarding elements that do contain more distinct query terms. The system is not sensitive to the value of K and a value of k=5 is adequate [3]. The summation is performed over all n terms that are found within the element where t_i is the frequency of the ith query term in the element and f_i is the frequency of the ith query term in the collection.

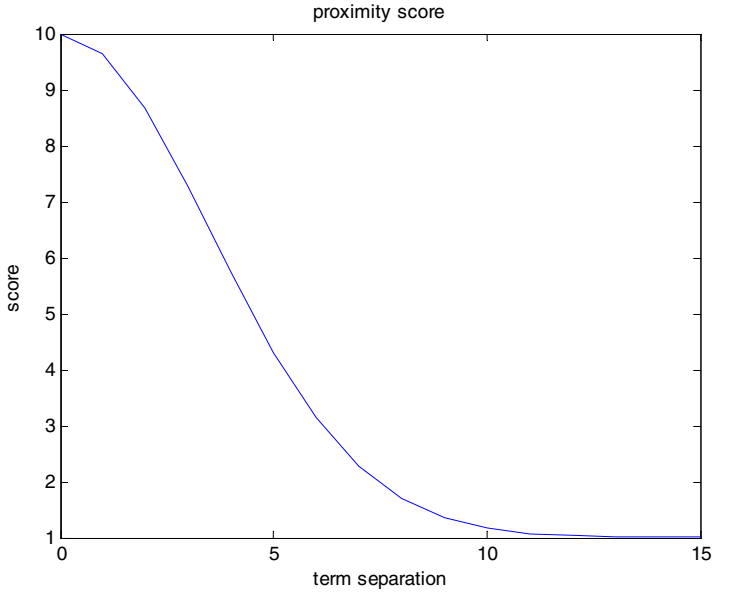

Fig. 1. Proximity score as a function of term separation

Finally, nodes that contain query terms that are preceded by a minus sign (undesirable) are eliminated.

At this point we have computed the score of all (overlapping) nodes in each article that contains query terms. The score of the <article> node itself is then added to all nodes in the article. This lifts the scores of all nodes that appear in a high scoring article. The intuition is that an article with many scoring nodes is more likely to be relevant and so all its scoring elements are ranked higher on account of more scoring nodes appearing in the same article. Without this modification, two similar nodes, one being an isolated instance of a relevant node in an article, and the other being one of many relevant nodes in an article, would receive a similar score.

As we will see below, results suggest an improved performance in GPX. The runs labeled RIC_04 and BIC_04 were produced with the 2006 GPX version (score propagation) while BIC_07 and RIC_07 were run with the GPX_07 version with direct score calculation. The GPX 07 version seems to perform better than the earlier GPX version. It does not require any magic numbers (decay constants), it treats each node as if it were a document, and it is therefore conceptually less arbitrary and more appealing.

2.3 GPX and Ad-Hoc Retrieval Tasks

The Ad-Hoc track at INEX 2007 consisted of 3 tasks – Focused, Relevant in Context, and Best in Context. These tasks are described elsewhere in this proceedings collection. We briefly describe the approach taken to each of the tasks in our best performing run.

2.3.1 Focused Retrieval

Focused Retrieval starts with the thorough results recall base. Within each article the highest scoring elements on a path are selected by keeping only elements that have a higher score than any of their descendents or ancestors. The submission consists of the remaining overlap free focused elements, sorted by descending score.

2.3.2 Relevant in Context Retrieval (RIC)

The objective of the task was to balance article retrieval and element retrieval. Whole articles are first ranked in descending order of relevance and within each article a set of non-overlapping most focused elements are grouped. We have used the focused results, which were overlap free already, but grouped the elements within articles and sorted the articles by article score.

2.3.3 Best in Context Retrieval (BIC)

We tested a straightforward approach here – we simply kept the highest scoring element in each document appearing in the focused recall base.

3 Ad Hoc Retrieval Results

The GPX system performed well and produced particularly good results in the Relevant in Context and Best in Context tasks of the Ad-hoc track.

Table 1. Comparative results for GPX vs. the best performing run at INEX 2007. RIC and BIC tasks are measured by MagP while Focused is measured by interpolated precision at 0.01 recall.

Task	Best run	Best GPX	Run Rank	System rank
RIC	0.1552	**0.1489**	6/66	2/17
BIC	0.1919	**0.1831**	2/71	2/19
Focused	0.5271	**0.4924**	15/79	8/25

Relatively good results were achieved in terms of precision at early recall levels on most of the tasks. We have submitted several variations of the GPX search engine in order to compare the performance. The GPX 2007 variation was compared with the GPX 2006 system, unchanged. The results were similar and are depicted in Table 2.

Table 2. Comparative results for GPX 2007 vs. GPX2006

Task	GPX 2006	GPX 2007	GPX 2007*
RIC	0.1298	**0.1489**	0.1369
BIC	0.1808	**0.1831**	0.1519
Focused	0.4924	**0.4828**	0.4235

Overall the performance of the GPX 2007 version (bold) is a little better than the 2006 version. The column titled GPX 2007* corresponds to GPX 2007 without the term proximity correction in equation 3. It is worth noting that the results with the correction for term proximity are considerably better than the results without this correction. The proximity correction is necessary because when computing node scores directly from content, in very large nodes (e.g. whole article) the terms may well be independent of each other. On the other hand, when dealing with short documents, the correction is not necessary. The GPX 2006 search engine computes node scores in relatively small passages – typically paragraphs. Term proximity is not particularly advantageous when documents (elements) are short. Our experiments with term proximity in previous years, with GPX 2006, provided insignificant variations in the scores when switched on or off.

4 Link the Wiki

The Link the Wiki task is described in detail elsewhere in this proceedings collection. The objective of this task was to identify a set of incoming links and a set of outgoing links for new Wikipedia pages. In practice, the topics were existing Wikipedia pages that were stripped of exiting links. The links were only at the article-to-article level.

4.1 Incoming Links

Incoming links were identified by using the GPX search engine to search for elements that were *about* the topic name element. For each topic the *name* element was used to construct a standard NEXI query: //article[about(.,*name*)]

We have used the SCAS task setting whereby the results were interpreted strictly. In this case it only means that articles nodes were returned. This was sufficient since only article-to-article links were needed. Results were ordered by article score with the more likely relevant articles returned earlier in the list. The process took an average of 8.5 seconds per topic on a standard mid-range PC. Considering that the task that is supported by the LTW process is the creation of new documents in the collection (this is the use case), 8 seconds for identifying recommended links is not significant. By comparison, the creation of a new Wikipedia topic may in some cases be measured in minutes, but it is more likely to take hours, even days. Hence the response time is quite adequate.

4.2 Outgoing Links

We have adopted a very simple approach to this task. All existing page names in the Wikipedia were loaded into an in-memory hash table. With 660,000 articles this is not an onerous task. The identification of potential links was based on a systematic search for anchor text that matches existing page names. In the first stage we have extracted the text of the topic (eliminating all markup information.) Prospective anchors for outgoing links were identified by running a window over the topic text and looking for matching page names in the collection. The window size varied from 12 words down to 1 word, and included stop words. Longer anchors were ranked higher than shorter ones, motivated by the trivial observation that the system was less likely to hit on a longer page name by accident. A naïve approach perhaps, but quite effective as it turns out. The process is purely computational and does not incur any I/O operations. The process took an average of 0.6 seconds per topic.

While it is straight forward to obtain candidate anchors by systematic comparison of substrings (of various lengths) against exiting page titles in the collection, numerous matches arise, and not all are useful. A pruning strategy is needed. We adopted the following process:

- Identify all candidate phrase-anchors of length 12 words down to 2, in that order.
- Append candidate year anchors
- Append all single term anchors

No ordering was performed other than the above. Phrases were ordered by length, followed by years, followed by single terms. Within these groups the ordering was in the sequence in which the anchors were encountered. The heuristic is simply that we are unlikely to encounter long phrases, which happen to be page names, by accident. On the other hand, single terms matches are more likely to be accidental in an encyclopedic collection and thus more risky in recommending at higher rank. Of course had we performed a deeper analysis of the anchor text context and the target document context we may have been able to resolve ambiguities, but that would have complicated the approach considerably.

5 Link the Wiki Results

The official results of the evaluation are depicted in figures 2 and 3. It should be noted that better results were subsequently obtained by the University of waterloo for

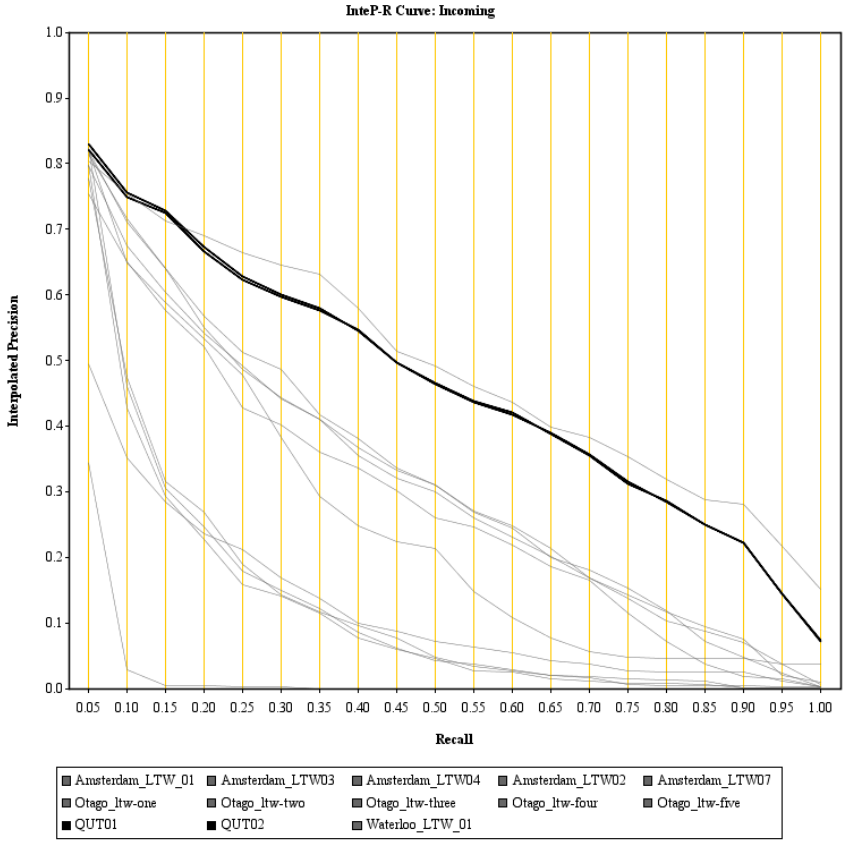

Fig. 2. Link-the-Wiki Incoming links, interpolated precision vs. Recall (official results)

Outgoing Links and we incorporate these unofficial improved results in Figure 4 for comparison.

It is evident from the official evaluation plots that the performance of our system was quite good – when viewed in terms of conventional IR. However, the Link the Wiki task is far more demanding than the conventional web search engine task. Here it is *not* sufficient to identify several good results. The user is interested in numerous anchors and links – almost exhaustive recall (within the limits of reason). All of the proposed links have to be checked by the user because it is highly undesirable to have inappropriate links. Although precision at early recall points may be at 0.8, it is even more desirable to achieve a high MAP value (i.e. high precision through a long tail of recall levels.)

Figure 4 presents the precision-recall curves for the two systems. “Anchors 90” is the Waterloo system and “Page Titles 90” is the QUT system. Both are shown for the 90 INEX topics. The link analysis approach (Waterloo) is better at almost all recall points, however, in a collection that is not already extensively linked it will not be

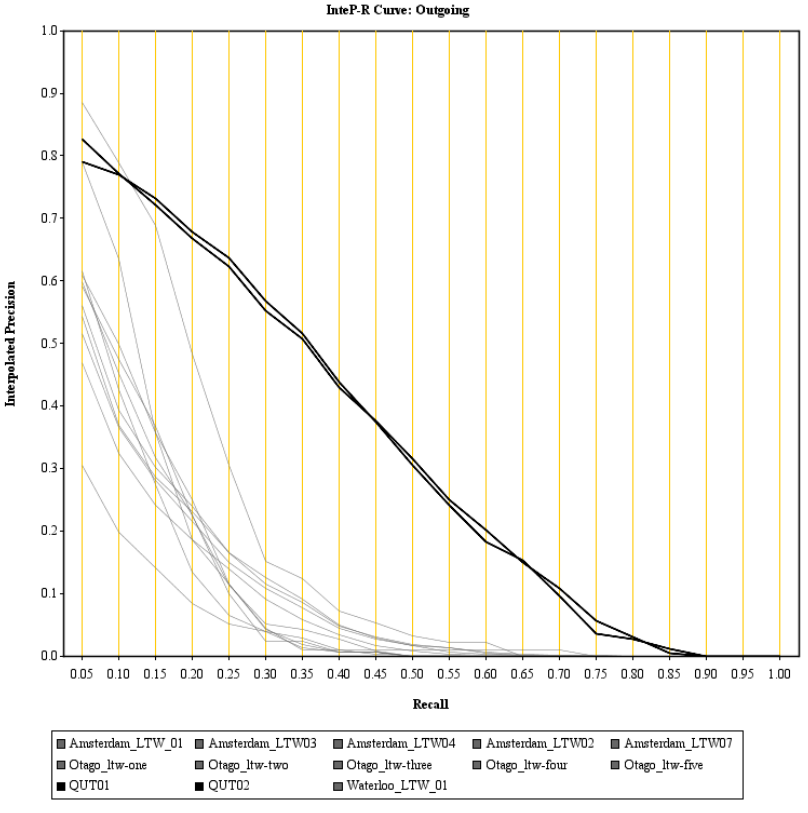

Fig. 3. Link-the-Wiki Outgoing links, interpolated precision vs. Recall (official results)

applicable. The QUT approach is independent of existing links. The Waterloo system performance in Figure 4 provides the current best performance and it is included here to provide a baseline for comparison.

To test the scalability of automated link discovery we additionally ran an extensive experiment on the collection. We randomly extracted 1% of the 660,000 documents and re-ran the experiment. The experiment was run on a PC with 2GB memory and 1.6GHz clock speed. It took 6 minutes to complete the process, processing in excess of 1,100 documents per minute. Figure 4 also presents the recall-precision curve for that run, labeled Page Titles 6600. It can be seen that performance over a very large number of topics selected at random is similar to the performance achieved over the hand picked INEX set, suggesting that 90 hand picked topics are sufficient to measure the performance of link discovery systems. Importantly in the context of the INEX evaluation, it is feasible to manually assess 90 topics whereas it would not be feasible to assess 6,600 (using the resources available to INEX). Manual assessment in future cycles of Link the Wiki would allow us to study more deeply the nature of link discovery, to identify those links returned by automatic systems that have not yet been

identified by Wikipedia authors, and those automatic link links that already exist in the Wikipedia and which are not useful (e.g. *year* links are common, yet often of no use).

In order to assess the contribution of each component (phrase/year/term), we created separate submissions for each component. Figure 5 presents the recall-precision curves. Most surprisingly, the contribution of the year links is small; they are ubiquitous throughout the Wikipedia and were expected to contribute considerably to performance. However, the precision of year links at low recall levels is relatively low. Single terms contribute more than years not only at low recall levels, but over all. This is because there are many more terms that could be linked. However, it is difficult to avoid irrelevant links using only single term anchors. The phrase links achieve higher precision and recall than terms and years. Phrases (and long phrases in particular) that match page names are less likely to occur in a document without also being related to the page of the same name. Years and single terms frequently match a page name, but not in a meaningful context. The combination of phrases, years, and terms is very effective as can be seen from the combined curve labeled Phrases Years Terms in Figure 5.

It is possible to improve the ranking of single terms by estimating the likelihood that a term will be a page name. We estimate this likelihood as the ratio of the number of times that the page is linked to over the number of times that the page name appears in the collection (using either the term collection frequency or the term document frequency). Indeed the ranking of single terms in this manner provides further improvement. The top 2 curves in in Figure 5 correspond to these variations. The improvement is only marginally greater when using the document frequency in place of collection frequency. The performance of the GPX system with this correction is significantly better (see figure 4).

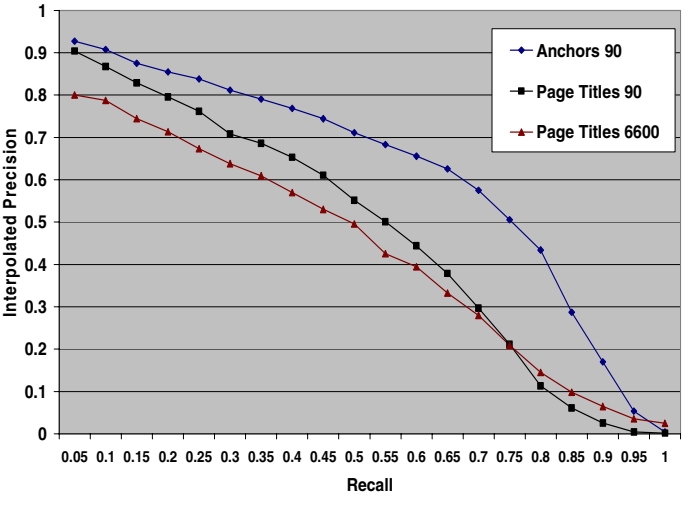

Fig. 4. Link-the-Wiki Outgoing links, interpolated precision vs. Recall

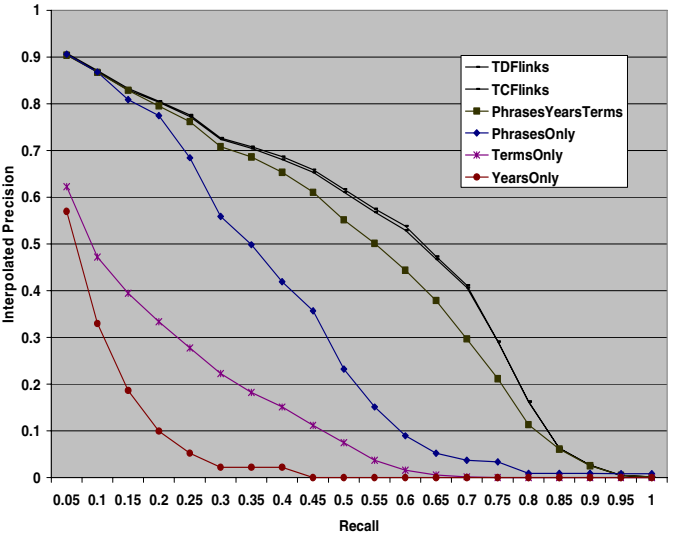

Fig. 5. Link-the-Wiki Outgoing links, interpolated precision vs. Recall

6 Conclusions

The GPX search engine was briefly described and modifications to the earlier version were described. Results indicate that performance is marginally better in terms of precision and recall, however, the approach is easier to implement. At any rate, both GPX 2006 and GPX 2007 produced highly competitive results in 2007.

Although relatively good results were produced for the Link the Wiki task, performance may be improved through analysis of the contexts surrounding the anchor texts and the corresponding target documents. This remains as a task for the next round of Link-the-Wiki evaluations in 2008.

References

1. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum. 40(1), 64–69 (2006)
2. Comparative Evaluation of XML information Retrieval Systems 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 17-20. LNCS. Springer, Heidelberg (2007) ISBN 978-3-540-73887-9
3. Geva, S.: GPX - Gardens Point XML IR at INEX 2006. In: Comparative Evaluation of XML information Retrieval Systems 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 17-20. LNCS, pp. 137–150. Springer, Heidelberg (2007)
4. Robertson, S.: Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation 60(5), 503–520 (2004)
5. Wilkinson, R., Smeaton, A.F.: Automatic Link Generation. ACM Computing Surveys 31(4) (December 1999)

6. Ellis, D., Furner-Hines, J., Willett, P.: On the Measurement of Inter-Linker Consistency and Retrieval Effectiveness in Hypertext Database. In: Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 51–60 (1994)
7. Green, S.J.: Building Hypertext Links By Computing Semantic Similarity. IEEE Transactions on Knowledge and Data Engineering 11(5), 713–730 (1999)
8. Allan, J.: Building Hypertext using Information Retrieval. Information Processing and Management 33(2), 145–159 (1997)
9. Green, S.J.: Automated Link Generation: Can We Do Better than Term Repetition? In: Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pp. 75–84 (1998)
10. Zeng, J., Bloniarz, O.A.: From Keywords to Links: an Automatic Approach. In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004), 5-7, pp. 283–286 (2004)
11. Adafre, S.F., de Rijke, M.: Discovering missing links in Wikipedia. In: Proceedings of the SIGIR 2005 Workshop on Link Discovery: Issues, Approaches and Applications, Chicago, IL, USA, pp. 21–24 (August 2005)
12. Jenkins, N.: Can We Link It (2007), `http://en.wikipedia.org/wiki/User:Nickj/Can_We_Link_It`