SlideSeer: A digital library of aligned document and presentation pairs

Min-Yen Kan
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
kanmy@comp.nus.edu.sg

ABSTRACT

Research findings are often transmitted both as written documents and narrated slide presentations. As these two forms of media contain both unique and replicated information, it is useful to combine and align these two views to create a single synchronized medium. We introduce SlideSeer, a digital library that discovers, aligns and presents such presentation and document pairs. We discuss the three major system components of the SlideSeer DL: 1) the resource discovery, 2) the fine-grained alignment and 3) the user interface. For resource discovery, we have bootstrapped collection building using metadata from DBLP and CiteSeer. For alignment, we modify maximum similarity alignment to favor monotonic alignments and incorporate a classifier to handle slides which should not be aligned. For the user interface, we allow the user to seamlessly switch between four carefully motivated views of the resulting synchronized media pairs.

Categories and Subject Descriptors: H.3.3 **Information Systems** – Information Search and Retrieval

General Terms: Algorithms, Design, Human Factors

Keywords: SlideSeer, presentations (slides), synchronized media, digital library, fine-grained alignment

1. INTRODUCTION

Scholarly digital libraries (DLs) today play a major role in information dissemination and learning. Such digital libraries help in making published work more accessible and allow researchers to search for work of interest. Many fielded DLs function as institutional repositories, which serve both to archive the institution's achievements while disseminating these published articles to the public.

In most cases, these DLs house only published documents such as post-print journal articles and conference papers. However, papers only constitute the most visible aspect of the research work – presentation foils, emails, author and project home pages, datasets and tools – also contribute to understanding research results [3].

In particular, slide presentations are important because they are summarized, narrated forms of the published work. They constitute a dual view of the published work, often quite different from the paper, and can help a first time reader of the paper grasp the work at a high level. Perhaps for this reason, authors occasionally place such slide presentations of their work on their websites. Our study shows that the rate of growth of this type of media is increasing and becoming commonplace. Despite this, to our knowledge, no DLs have yet to systematically archive such data.

Since presentations constitute a dual view, further utility can be gained if the two media are synchronized. In such a DL, fine-grained alignment between slides and document passages are constructed, allowing a user to view both media simultaneously. Such a DL can go beyond standard document-level retrieval, and implement for passage-level (e.g., slide or document section) retrieval.

We present our work on SlideSeer, a digital library of presentation slides and documents that addresses these needs. SlideSeer performs the fine-grained alignment of slides to documents that allows detailed viewing of document sections and individual slides simultaneously. SlideSeer's web based user interface employs scripting to add a level of dynamicity to the web page.

This paper describes the system aspects of SlideSeer. We first describe the system's overall architecture, and proceed to discuss the three components. We start describing the resource discovery component which locates presentation and document pairs from the Web and performs the subsequent conversion processing. We then describe the alignment component, which employs basic text similarity within an alignment algorithm to compute the fine-grained alignments. Finally, we describe the user interface component and how its construction is motivated by use case scenarios. We conclude with discussion on ongoing work.

2. SYSTEM OVERVIEW

SlideSeer is a customized digital library which comprises of an offline discovery, alignment, and indexing system and an online web user interface. Figure 1 shows the general architecture of the system. Offline, SlideSeer uses a resource discovery component which locates and downloads freely-available presentation and document pairs. Once a suitable pair is downloaded, both presentation slides and documents are pre-processed into plain text files, and each slide is exported into a set of image files. The alignment component takes both inputs in as plain text and outputs a separate metadata file which maps each slide to a set of consecutive paragraph(s) in the document.

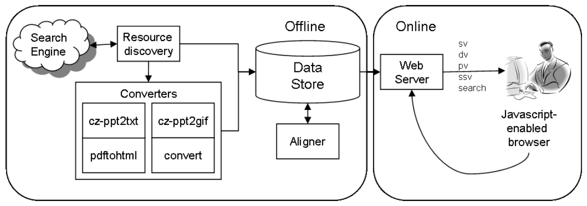


Figure 1: SlideSeer system architecture.

Online, SlideSeer is accessible through a standard web user interface, employing a freely-available cross-browser compatible Javascript library[1] to achieve interactivity. When an HTTP request for a particular presentation/document pair is made, the default slide view is presented. The interface component enables users to switch between several different views of the same presentation/document pair:

1. a Slide View (sv) that aligns consecutive document paragraphs to each slide, in which only one slide at a time is shown in the interface;
2. a Document View (dv) that aligns consecutive slides to each document section, in which only one document section at a time is shown;
3. a Print View (pv) that concatenates all of the document views together for printing a hardcopy of the coordinated media;
4. a Slide Show View (ssv) that maximizes the slide image in the browser window.

Currently, only presentations in Microsoft PowerPoint and documents in Portable Document Format (PDF) are handled. From an informal survey of such presentation/document pairs, this fits the majority of existing pairs, although presentation foils in PDF (prepared in SliTeX or exported from other office productivity software suites) are also common. Also, due to our current resource discovery policy described next, most pairs indexed by SlideSeer deal only with the discipline of computer science.

3. RESOURCE DISCOVERY

Any physical library is useless without materials to disseminate and circulate. The same is true of digital libraries such as SlideSeer. SlideSeer requires known document and presentation pairs to create its value- added synchronized views.

Such pairs could be obtained through user uploading or focused crawling. Instead, we have focused on an alternative bootstrapping approach that utilizes existing DL metadata. In particular, our group has access to a copy of the CiteSeer DL (containing over 750K noisy metadata records of computer science articles) and a snapshot of the DBLP metadata from August 2006 (containing over 1.2M metadata records).

[1]from `www.cross-browser.com`

Each of these records gives the title, author(s) and year of publication of a document, among other things. We can utilize these known document titles to locate any freely-available copy hosted on the web using the title as a a search engine query. We can use advanced file type operators with the search engine to filter out irrelevant results, and present only PDF and PPT documents of interest to us.

This scheme is an instance of the search engine-based focused crawling algorithm, in which queries to a search engine are used to locate suitable materials for collection building. Many digital libraries frameworks use similar schemes to populate their collections. In our domain of scholarly DLs, related work includes PaperFinder [15], MOPS [6] and PaSE [14], all which find either citation data or online copies of a document given document titles.

```
Algorithm ResourceDiscovery
1.  $C$ ←the set of CiteSeer metadata records
2.  $D$ ←the set of DBLP metadata records
3.  $E$ ←non-repetitive DBLP records, initially nil
4.  $P$ ←pairs of ppt/pdf, initially nil
5.  index $D$ into inverted list
6.  for $c \in C$
7.      do $hits_c \leftarrow \mathrm{retrieve}_n(c, \lambda_1)$
8.         for $hit \in hits_c$
9.             do $\alpha \leftarrow \mathrm{sim}_1(hit, c)$
10.               if $\alpha \geq \lambda_2$
11.                  then add($d, E$); add($d, C$)
12.                  else  discard($d$)
13. for $c \in C$
14.     do $pptList$ ←query(titleOf($c$),“PPT”)
15.        if $c \in E$
16.           then $pdfList$ ←query(titleOf($c$),“PDF”)
17.           else  $pdfList$ ←citeseerURL($c$)
18. for $d \in pdfList$
19.     do $t_d$ ←convertToText(fetch($d$))
20.        for $p \in pptList$
21.            do $t_p$ ←convertToText(fetch($p$))
22.               $\beta \leftarrow \mathrm{sim}_2(t_d, t_p)$
23.               if $\beta \geq \lambda_3$
24.                  then add($d, p, P$);
25.                       break
26. return $P$
```

We now outline our resource discovery algorithm in detail; a concise algorithmic description is given above. In SlideSeer, the resource discovery process consists of three steps. The first step is to determine the target set of metadata records to locate resources for. In our bootstrapping process, we merge the DBLP metadata with the CiteSeer metadata (lines 5-12) in a noisy, approximate text join [7]. Each CiteSeer and DBLP record consists of fielded data, of which only title, author and year are used to determine record linkage.

Normally, record linkage requires all $O(nm)$ pairs of records to be examined, as each DBLP record could match any potential CiteSeer record. This is unacceptable given the sheer size of the datasets, necessitating algorithms with lower complexity. Typically, record linkage tasks complexity can be greatly reduced using blocking [13] or canopy [9] methods, which use an inexpensive technique (such as hashing) to remove improbable pairs from consideration. Inspired by information retrieval (IR) techniques, we employ inverted indexing to reduce the complexity to linear time. We use the relevant title, author and year fields from DBLP and index them using Lucene[2], an open source IR library. Once the index is built, each CiteSeer metadata record is treated as a query to retrieve the top n candidates.

All candidates above an initial score threshold of λ_1 are then pairwise examined with the query CiteSeer record for a possible match. The Jaccard measure (i.e., $\frac{|A \cap B|}{|A \cup B|}$) is computed between the two records. If the measure exceeds the λ_2 threshold (set to .7 from observation), they are considered duplicate records. The result of this process is an automatically merged metadata file (the resulting list C in the algorithm) between the DBLP snapshot and the CiteSeer repository, amounting to some 1.5M records.

The second step (Lines 13-17) queries a search engine for appropriate documents and presentations. In our current development setup, we use the Google API to systematically retrieve the top 10 results for each record using the title (unquoted, as separate keywords) and an appropriate filetype restriction (*filetype:pdf* for documents, *filetype:ppt* for presentations). Note that the CiteSeer mirror that we host already contains the cached PDF files downloaded by the original CiteSeer crawler. As such, when dealing with records present in CiteSeer, we save a potential query. The results and date of execution of each API query is archived and indexed locally in SlideSeer, to avoid query duplication and to cater for future re-analysis of the query results.

The final step (Lines 18-26) compute the initial document-level alignment pairs. For each metadata record, we consider each possible document/presentation pair: for records from CiteSeer, we make at most 10 comparisons (1 known document, 10 possible presentations), and for records only present in DBLP, we make at most 100 comparisons. We tokenize the returned snippet and calculate the Jaccard similarity with the record's title and author fields to decide whether a resource is the corresponding presentation or document. If the similarity exceeds λ_3 (set empirically to 0.7) it is linked to the record. This second check uses author information, which was not employed in the query phase. We do this simply because the title + author fields together often exceed the 32 word limit for queries imposed by Google's API limit.

We attempt to download each linked resource and if successful, its file size is checked to filter out obvious mismatches (when files are too short, or the download is unsuccessful). Successful downloads are noted as well as unsuccessful ones (for example, if a site's robots.txt file disallows downloads, we do not attempt to download from this site again). The output of this phase are documents (only in PDF) and/or presentations (only in PPT) that correspond to the metadata record, cached into the SlideSeer data store.

Downloaded resources must be preprocessed into textual form for the alignment module, and images for each slide must be produced. We employ batch utilities for conversion, using `pdf2html` to generate rich, format preserving HTML output for the PDF files and an in-house Visual C++ utility that uses Microsoft's API to access PowerPoint's Document Object Model in batch mode to output the text. We post-process this HTML format to extract sequential, plain text paragraphs. A pipeline consisting of a commercial utility (`cz-ppt2gif`) paired with the open-source graphic converter (`convert`) is used to export interlaced PNG images for each slide at our maximal resolution of 1280×1024. This format is chosen for its comparatively good compression rate, lossless property and two-dimensional interlacing scheme, allowing images to "fade in" as needed.

3.1 Performance

All of the above steps must work correctly to yield a usable pair of resources for analysis. There are many points of potential failure in our system, each with some percentage of loss. When cascaded together, a significant percentage of potential presentation/document pairs are filtered out, to ensure a high level of precision. Only in cases where the system is able to extract the text from both types of resources can SlideSeer proceed to do alignment as discussed in the next section.

In the first stage, errors in the linking process can merge records that should have been kept distinct. We conducted a random sample of the results, which suggest the precision of our duplicate linkage is around 95%. While these results are not considered state-of-the-art, they are more than sufficient to create a starting point for the resource collection.

The second querying stage may introduce both errors of commission or omission. Incorrect slide/document pairs can pass through the filters if the two media are similar enough. Most often, these cases occur with slide presentations that are classroom lecture materials that discuss the target paper for a portion of the lecture, or when slides (instead of the paper) are found in the top ranking PDF results. Post-query author-based filtering (i.e., sim $\geq \lambda_3$) is particularly helpful in removing false pairs, reducing potential alignment errors by around 30% in our study of a random sample of 100 pairs. Errors of omission require sampling the documents where the system reports not finding a pair. These cases are hard to track as most documents (especially journal papers) do not have publicly available slide sets.

The final stage can face conversion challenges. PDF files can be set to disallow text extraction or may encode a raster image of the page (typically when scanned) rather than the raw text. Font encodings can also garble the text extraction process. Our post-processing filters out these types of errors, ensuring that the final results are processable. Even when text is extracted, our processor sometimes splits actual paragraphs into separate detected ones, or lumps multiple paragraphs together as one detected paragraph. Our informal survey shows that these series of checks filters out an additional 10% of potential errors.

Overall, we estimate the precision of this pipeline to be about 85%. Recall is difficult to measure without a known dataset of pairs, so we do not give an exact figure here. We believe the largest fallout occurs with presentations that use other presentation software. Estimates from polling colleagues and surveying research group websites indicates that SlideSeer may miss up to 20% of other pairs due to our current policy of looking only for PowerPoint (PPT) files.

In total, SlideSeer currently has over 11K filtered resource pairs downloaded, although we currently have processed only about 200K records of the full 1.5M merged list.

[2]`lucene.apache.org`

4. ALIGNMENT

Given a known corresponding presentation and document pair, we need to find a suitable fine-grained alignment. Creating such a unified medium requires aligning slides to a corresponding paragraph span in the document. We formalize this problem as *document-to- presentation alignment*:

> Given a slide presentation S consisting of slides s_1 to s_n and a document D consisting of text paragraphs d_1 to d_m, an *alignment* is a function $f(s) = (x, y)$, mapping each slide to a contiguous set of document paragraphs, starting at d_x and ending at d_y where $x \leq y$, or to *nil*.

Nil alignments are necessary to represent that a slide is "extra"; its contents are extraneous or not represented in the content of the document. We choose the granularity of the document as paragraphs (rather than sections or pages) as we want to capture finer-grain alignments. Note also that this alignment is directional; an optimal, reversed presentation-to-document alignment may not match, as the granularity of the spans may differ.

We approach this problem from an information retrieval perspective. In this sense, a slide should align to its most similar span of document paragraphs. However, we also have to account for a global preference to prefer monotonic alignments (where slide and document paragraph numbers strictly increase) over ones where consecutive alignments cross many times. Finally, we introduce a post-alignment classifier to determine whether the alignment for a slide should be kept or dropped in favor of a *nil* alignment.

We view alignment as series of nested processes, of which an instance is shown in the algorithm below. We first touch on related work, then describe how our system judges similarity between slide and presentation text first. The similarity measure is used to compute a full similarity matrix between slides and document paragraphs (Lines 5-7 below), which in turn is used to calculate alignment (Lines 8-15). The *nil* classifier is applied to transform poor alignments to *nil*s, based on a set of features (Lines 14-15).

```
Algorithm AlignmentWithNilClassifier
1.  $S_{1-n}$ ←text of $n$ slides from the presentation
2.  $D_{1-m}$ ←text of $m$ paragraphs from the paper
3.  $M$ ←similarity matrix, initially nil
4.  $P_{1-n}$ ←alignments, initially nil
5.  for $s \in S$
6.        for $d \in D$
7.              $M_{s,d} \leftarrow sim_1(s, d)$
8.  for $i \in |S|$
9.      do $P_i$ ←align($s_i, M$)
10. for $i \in |S|$
11.     do $P_i$ ←correct($s_i, M, P$)
12.        $P_i$ ←extend($s_i, M, P$)
13.        $\alpha$ ←nilClassify($P_i, M$)
14.        if $\alpha \leq \lambda_1$
15.           then $P_i \leftarrow nil$
16. return $P$
```

4.1 Related Work

Alignment is a pervasive and well-studied problem in many domains, including media synchronization. We discuss alignment strategies from three related areas.

Media. Multimodal interaction has been a specialized interest area of multimedia research in the last five years, resulting in series of workshops (e.g., [1]) as well as commercial products[3]. This research is closely related in terms of application to the work presented here, featuring algorithms to align multimodal events, including presentations where the two media are the spoken dialogue stream and the presentation slides.

However, such works [18, 11, 10] typically deal with naturally synchronous media. In the previous example, the dialogue stream and the slides are, by design, already in the correct order. The task is simplified to finding the correct alignment points for which one media should be advanced to the next unit (e.g., forward one slide in the presentation). In contrast, our problem needs to handle alignments in which the presentation and the document are not necessary monotonic. An example of this is when the "related work" section at the beginning of a document is relegated to the end of the presentation.

Multilingual text alignment (MTA). MTA aligns text from two (or more) textual sources in different languages which are translations of each other. The granularity of the textual units are typically sentences or words. Such alignments are useful in machine translation and bilingual lexicography [4, 2] and also to the study of evolution of languages or etymology. Our problem can be viewed as a instance of MTA since alignments are not necessarily monotonic and *nil* alignments can occur.

MTA approaches can be divided into two main classes: lexical and statistical [19]. Lexical approaches rely on bilingual lexicons to match sentences [12], while statistical approaches rely on statistic features [4, 2]. Statistical approaches generally choose the most probable alignment, in terms of ML or MAP. Several methods are possible, however the most common approach uses dynamic programming [4, 19], as its exhaustive search can be done in polynomial time $O(n^3)$ requiring only $O(n^2)$ memory. The most widely used dynamic programming approaches are based on variations of the Viterbi algorithm [16]. Surprisingly, using a simple heuristic such as the length of sentences gives quite a good accuracy of close to 90% [4]. Although statistical methods rely on little domain knowledge, they generally perform better than more sophisticated lexical approaches.

Summary Alignment. Hayama *et al.* [5] have studied our specific problem before, in aligning Japanese presentation and technical papers. Their work uses an variation of the Hidden Markov Model (HMM) [8] used to decompose sentences in summaries to corresponding sentences in the full document. Hayama *et al.* use this approach to find the most likely source location in the document for each slide's text. They encode knowledge about slide titles and likely position gaps to enhance their method. In SlideSeer, our extraction process does not (yet) yield formatting information (i.e., slide titles) and our results are not directly

[3]e.g., StreamSage's synchronization software

comparable. In addition, their problem formulation does not deal with *nil* slide alignments nor with span alignments, both of which we have found to be quite common.

4.2 Similarity Measures

At the core of the alignment process, spans of texts from the slides and the document must be compared. SlideSeer utilizes the full, nm slide-to-paragraph similarity matrix for subsequent alignment. For each pair s_i, d_j, several different similarity metrics can be applied (sim_1 in the algorithm). We computed such matrices for cosine and Jaccard similarity. For cosine similarity, corpus frequencies are required for the inverse document frequency (IDF) term. To get reliable estimates of IDF, we have used the counts compiled from the large WebBase corpus[4] rather than rely on our limited training data.

A problem with the above metrics is that they treat the texts as a bag of words, losing ordering information. To add in order information, we also tested using bigrams (sequences of two consecutive tokens) with the Jaccard similarity. We tested three different Jaccard schemes: only single tokens (unigram, as in the above paragraph), only bigrams, as well as combining both unigram and bigram similarity.

4.3 Alignment methods

We implemented and tested several different models for alignment: A simple maximum similarity model which ignores global constraints, an edit distance method that imposed strict monotonic alignment, and our reimplementation of the HMM model discussed by [8]. Note that all of these alignment methods align a slide to a single paragraph; we extend our approach to span alignment later.

1. **Maximum paragraph similarity.** This method simply aligns a target slide to the paragraph with the maximum similarity. This model is a greedy model that can make many jumps during the alignment process.

2. **Edit distance.** A dynamic programming approach is used to calculate an optimal, monotonic path through the similarity matrix. At each cell in the matrix, we compute the optimal path by deciding whether to skip a slide, skip a paragraph text or match the current slide to the current paragraph, as given by Equation 1. Here, the first case aligns a current slide s to the current paragraph p; the second skips over slide s (meaning that we align s to *nil*; the final case skips over the current paragraph.

$$score(s,p) = max \begin{cases} sim(s,p) + score(s+1, p+1) \\ score(s+1, p) \\ score(s, p+1) \end{cases} \tag{1}$$

   The weakness of this model is that it cannot align slides to previously skipped paragraphs; the alignment must be nondecreasing.

3. **Local jump model.** In this model, we alter the rigorous monotonic restriction of the edit distance model. We relax this restriction and allow the alignment to a paragraphs in the near past (within 5% of the total number of paragraphs). This relaxation adds additional overhead to our search space in dynamic programming from; otherwise nothing else changes.

4. **Hidden Markov Model (HMM).** This approach tries to attribute the text of the slides to specific paragraphs in the document. Each word in a slide sequence is assumed to be generated from a word in somewhere in the paragraph sequence. Simplifying for clarity, given a word in a slide and its known origin in the document, it assumes that the next slide word is most likely to be the next word in the document (i.e., contiguous), less likely to be some other word from the same sentence, even less likely to be a word from a sentence later in the document, and most unlikely to be a word taken from anywhere else in the document. These transition probabilities can be trained if given sufficient data or can be set by the implementer. We follow Jing's manually set transition probabilities.

4.4 Span Extension and Alignment Correction

In our experiments, discussed later, we found that most simple baseline model using maximum similarity was most competitive; the more complex methods that directly account for global alignment constraints (i.e., favoring or requiring monotonic alignment) failed to increase alignment accuracy. Furthermore, with the exception of Jing's HMM method, none of the models can be easily adapted to align to paragraph spans.

For these reasons, we decided to use the baseline maximum similarity as a first pass to get initial alignment points. We then post-process the results to expand the alignments from single paragraphs to spans when possible and to impose our global alignment constraints to favor monotonic alignments (Lines 10-12 in the algorithm). This is done by a second pass over the initial alignments.

For each slide in the presentation, we retrieve the top n most similar paragraphs (n set to 10). We then examine all possible pairs of these n paragraphs to construct candidate spans; choosing one paragraph to be the start of the span, another to be the end. Note that when the same paragraph is picked for both start and end, the span consists of just the single paragraph. We then calculate the average similarity value over all paragraph included in the span. Intuitively, a multi-paragraph span may be a better alignment than a single paragraph when average similarity is comparable. To effect this, we multiply the average similarity value by the logarithm of the span length.

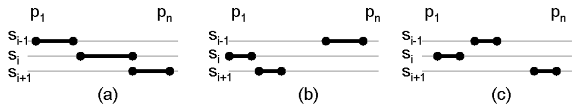


Figure 2: Alignment scenarios.

The second criteria for judging a candidate span is to see whether monotonicity is broken. We look at both neighboring slides' (i.e., s_{i-1} and s_{i+1}) alignments to judge how well the candidate span fits in the neighboring alignment. If the sequence of the three slide alignments are normal and all monotonically increasing, no penalty factor is applied,

[4]from elib.cs.berkeley.edu/docfreq/

as shown in Figure 2(a). If the candidate span causes the alignment to revisit paragraphs that are already past the current alignment point, a penalty is invoked. This may be acceptable when the alignment continues monotonically from the candidate span, as in (b). However, this penalty is set more severely when the two neighboring slides are in monotonically increasing order, as this suggests misalignment, as shown in (c).

These two factors are combined to determine the fitness of a span, as shown in Equation 2. The candidate span with the highest fitness score is used as the final alignment.

$$fitness(span) = (avg\_sim \times log(size + 1) \times (1 - penalty)) \tag{2}$$

4.5 Nil classifier

A final problem with the alignment described is that it does not handle slides that are not supposed to be aligned. Slides such as an ending "Any Questions?" slide or outline slides are examples of such cases. This is in contrast to the edit distance model which naturally models *nil* alignments.

We structured this challenge as a supervised machine learning problem. This *nil classifier* takes the alignment results and distills features from the alignment data and source document and slide text data. The resulting features are sent to a machine learner to learn a model of whether the alignment for a slide should be kept or whether the slide should be unaligned.

The nil classifier creates a feature vector for each slide and decides between the two classes of $\{nil, align\}$. A standard support vector machine (SVM) is employed with a linear kernel to implement the classifier. The following features are used:

1. **Similarity score.** A higher alignment score indicates a strong match. If a slide is strongly similar to its aligned paragraph, it is less likely to be a candidate for dropping.

2. **Number of words on slide.** More words on the slide can indicate content that should be aligned, however weak the alignment score is. Conversely, many slides with lots of embedded pictures and figures are likely to be examples and less strong candidates for alignment (especially if the examples are outside of the ones represented in the paper).

3. **All words in the slide.** The content of the slide can be indicative of whether a slide is supposed to be aligned or dropped. As stated, outline slides and conclusion slides that ask for comments from the audience often use formulaic words or phrases. We add all the words (canonicalized to lowercase) on the slide as individual features.

4. **Difference in alignment with previous or next slide(s).** This set of features look for singular changes in the alignment path, similar to our alignment fixing penalty score. If an alignment path is mostly on the diagonal but for a single slide jumps out and the returns to the path, it is likely that the alignment is noise, and that the alignment should be dropped. We have noticed that this occurs often with outline slides.

4.6 Evaluation

We compiled a small evaluation corpus of 20 presentation-document pairs, sampled from the resource discovery component. An associated researcher hand-annotated these pairs for gold standard answers, in the form of sets of $d_x - d_y$ indices that are acceptable alignments. As the purpose was to determine a gold standard, when multiple, non-overlapping spans were acceptable, the researcher was asked to annotate all spans (e.g., $1 - 3 : 7 - 10$). When no suitable alignment was found, the researcher recorded a *nil* alignment. Table 1 summarizes salient characteristics of our dataset.

Avg. # of slides	37.6
Avg. # of paragraphs	277.3
Avg. # of *nil* alignments	6.6 (17.4%)
Avg. # of span alignments	8.8 (23.4%)
Avg. # of point alignments	22.2 (59.2%)
Total	37.6 (100%)

Table 1: Dataset characteristics

As the system may create alignments that partially overlap with the gold standard, partial credit should be assigned. Our evaluation assigns a fractional credit using the Jaccard metric as an accuracy surrogate. This results in a [0...1] score range which penalizes both missing and added paragraphs. However, we believe recall is more important than precision for this task. For example, when a span is slightly longer than the exact alignment, the error may actually help the user by providing additional context around the correct alignment. For this reason we penalize false positives (extra alignment points) less; incurring 1/5th the penalty compared to false negatives (missed alignment points).

Method	Weighted Jaccard Acc.
1. Greedy Max (cosine)	33.4%
2. Edit Distance (cosine)	28.8%
3. Local Jump (cosine)	25.1%
4. Jing HMM	28.8%
5. 1. + correction & spanning (bigram)	39.9%
6. 5. + nil classifier (bigram)	41.2%

Table 2: Alignment accuracy.

Table 2 shows the results of representative alignment methods. The first four configurations describe the alignment methods discussed in Section 4.3. While these results may seem low in comparison to related work, it is important to note that most other works have compared using *soft accuracy* or variants, where a system's answer is considered fully correct if it returns any correct alignment point; we believe that our weighted Jaccard serves as a better approximation of the true difficulty of the problem.

As mentioned earlier, the simple maximum similarity method works best by a significant margin. When alignment correction and spanning post-processing is added using a bigram-based Jaccard similarity, we observe that some errors in local alignment are corrected, and spans can be captured. Adding support for spans does not actually contribute much in terms of our accuracy metric; many correct point (single slide) alignments are changed to spans, negating much of the advantage of correctly capturing spans. Finally, the *nil* classifier contributes a small percentage gain of 3% on top of the post-processed system. As *nil* alignments only consist of 17% of the alignments in total, such a gain is a significant improvement. A closer inspection of the results of the classifier shows that it is able to introduce correct *nil* classifications without inadvertently damaging any correct alignments.

5. USE CASES AND INTERFACE

Thus far we have discussed the technical aspects of creating aligned document pairs. What are the potential uses of such synchronized media by users? We identified several discrete use cases of such media. As SlideSeer currently seems to be only digital library that indexes presentations, we also examine use cases of this individual medium.

Learning and Comprehension. Synchronized media help users learn and comprehend a work. Small figures in a paper are often found as full slides in an accompanying presentation. Also, textual bullets in a presentation can serve as a summary of the work; synchronization means that a deeper understanding of the points may be traced directly into the full description in the document. Whereas the abstract of a document can be seen as assisting relevance judgment, we argue that the presentation assists comprehension. Even in the original setting, presentations are used to impart knowledge of the work to the audience. We believe an integral requirement is the deep linking of paired media, as described in the previous section. Simply having links to both forms of media certaintly helps, but does not make switching between the two media transparent and effortless.

Summarizing. Conference papers – the primary source of documents for our coordinated pairs – usually have length limits; which often finds authors squeezing figures and tables into cramped spaces that require careful analysis. In contrast, slide presentations are often meant for quick comprehension. Typically, a presentation covers the whole work within 15 to 30 minutes, which often means that presentation slides offer a summary of the work that fits nicely between the abstract and the full document. Such a resource could assist users needing to prepare critical summaries.

Searching for slides. While a picture may be worth a thousand words, it is difficult to locate relevant figures when good figures may lack words for an indexer to pick up. Using the full "query expanded" document to index presentations can help users find relevant figures, charts and diagrams.

Offline viewing. Despite recent advances in e-books and display technology, most users still prefer to read on physical paper. A suitable printed view must be an alternative to any online view of the synchronized medium.

Presenting. Presentations may need to be shown at any time; on both desktops and increasingly on mobile devices. While traditional web browsers can display images, most cannot natively handle either PDF or PPT formats. Current practice forces individual users to either export the slides into image formats (e.g., "Save As Web Page...") or activate presentation software to view the images. A digital library of presentations can centralize this effort, making all slides available as individual images, where a suitable addressing scheme allows each individual slide to be accessed by a proper URL. Ideally, the web browser can be repurposed as the vehicle for live presentations, as even mobile devices have modern web browsing capabilities.

Preparing own materials. Teaching professionals often source for materials to use in lesson and course planning. Such users often borrow presentation slides from others. Thus, an interface to facilitate this reuse is desirable. A centralized digital library of many presentations has an economy of scale that enables users to find suitable slide templates, figures and materials for pedagogy as well.

Comparing. The large scale of a digital library also has other intrinsic benefits. Users who do not understand concepts from one paper/presentation pair can locate related pairs, and browse them for comparison or to reinforce concepts introduced in another pair. Teaching professionals can easily search for a slide or figure that meets their exact requirements in a centralized system, rather than have to search and access slides through a large number of sites.

From the above, we conclude that there are two major potential advantages of a digital library such as SlideSeer: benefits reaped from a fine-grained synchronized medium and from the economy of scale in centralized collection and processing of such media.

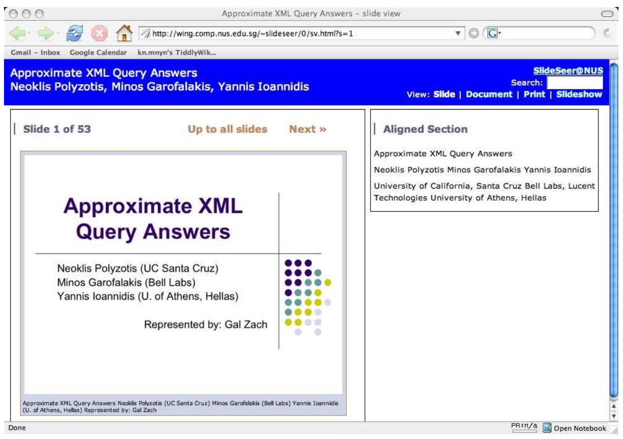
Our goal in creating SlideSeer is to be able to support these use cases implicitly with the user interface creating a minimal distraction from the DL content itself. For accessibility, a web browser interface is an optimal choice for a client, however, standard HTML is static and creates hassles for advanced users wanting keyboard shortcuts. Deploying a full-blown Java applet, Flash or SVG solution also creates problems for thin clients without proper plug-ins and analysis barriers for indexers. We have settled for a compromise by using dynamic HTML and JavaScript, for which browsers are deployed for virtually every platform, including mobile devices. We first discuss the detailed view of the media and then wrap up with a discussion of the collection-level considerations in SlideSeer.

5.1 Coordinated and Single Media Views

Given the desiderata above, we designed SlideSeer to offer four different views: three versions of the synchronized media and a slide show view specific to the presentation. Figures 3,4,6 and 7 show the slide, document, print and slide show views, respectively. The three synchronized views all feature a consistent header that displays the record metadata for the media pair on the left and navigation on the right. The navigation allows the user to switch between the different views given in the system. The pair shown in the screenshots is Polyzotis *et al.*'s *Approximate XML Query Answers* published in SIGMOD 2004.

The slide and document views are duals of each other, showing the medium in focus, with the other medium as context. Both views show a slice of the medium in focus (either an individual slide or document section), with hyperlink navigation to the previous or next slice.

For the slide view, a single slide of the presentation is shown along with the aligned paragraphs, as per the output of alignment module in Section 4. We believe this will particularly help users interpret figures on slides, as their context



Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis

Approximate XML Query Answers

Neoklis Polyzotis (UC Santa Cruz)
Minos Garofalakis (Bell Labs)
Yannis Ioannidis (U. of Athens, Hellas)

Represented by: Gal Zach

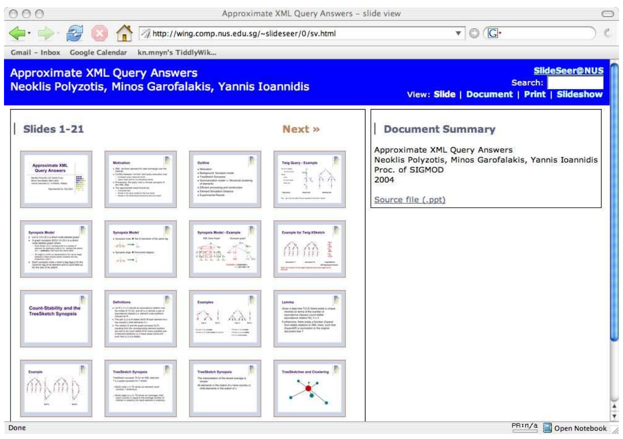
Approximate XML Query Answers Neoklis Polyzotis (UC Santa Cruz) Minos Garofalakis (Bell Labs) Yannis Ioannidis (U. of Athens, Hellas) Represented by: Gal Zach

Approximate XML Query Answers

Neoklis Polyzotis Minos Garofalakis Yannis Ioannidis

University of California, Santa Cruz Bell Labs, Lucent Technologies University of Athens, Hellas

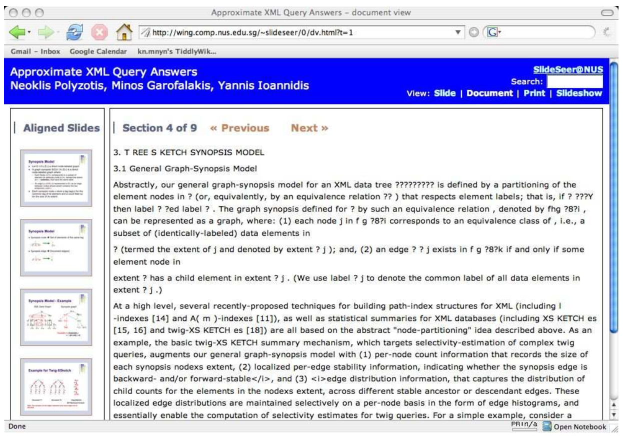
Figure 3: Slide View (sv). One slide is shown per screen, and any aligned paragraphs from the document.



Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis

Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis
Proc. of SIGMOD
2004

Figure 5: Gallery mode of Slide View (sv).



Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis

3. T REE S KETCH SYNOPSIS MODEL

3.1 General Graph-Synopsis Model

Abstractly, our general graph-synopsis model for an XML data tree ????????? is defined by a partitioning of the element nodes in ? (or, equivalently, by an equivalence relation ??) that respects element labels; that is, if ? ???Y then label ? ?ed label ? . The graph synopsis defined for ? by such an equivalence relation , denoted by fhg ?8?i , can be represented as a graph, where: (1) each node j in f g ?8?i corresponds to an equivalence class of , i.e., a subset of (identically-labeled) data elements in

? (termed the extent of j and denoted by extent ? j); and, (2) an edge ? ? j exists in f g ?8?k if and only if some element node in

extent ? has a child element in extent ? j . (We use label ? j to denote the common label of all data elements in extent ? j .)

At a high level, several recently-proposed techniques for building path-index structures for XML (including I -indexes [14] and A(m)-indexes [11]), as well as statistical summaries for XML databases (including XS KETCH es [15, 16] and twig-XS KETCH es [18]) are all based on the abstract "node-partitioning" idea described above. As an example, the basic twig-XS KETCH summary mechanism, which targets selectivity-estimation of complex twig queries, augments our general graph-synopsis model with (1) per-node count information that records the size of each synopsis nodexs extent, (2) localized per-edge stability information, indicating whether the synopsis edge is backward- and/or forward-stable</i>, and (3) <i>edge distribution information, that captures the distribution of child counts for the elements in the nodexs extent, across different stable ancestor or descendant edges. These localized edge distributions are maintained selectively on a per-node basis in the form of edge histograms, and essentially enable the computation of selectivity estimates for twig queries. For a simple example, consider a

Figure 4: Document View (dv). One document section is shown per screen. All slides aligned to any paragraphs in this section are shown as thumbnails.

are likely to be explained in the aligned paragraphs. For textual slides (e.g., bullets), the accompanying paragraphs can flesh out the text on the slide, such as adding omitted references or exceptions that were left out of the presentation due to time and space constraints. The extracted text from the slide is provided is also repeated as a caption so that users may copy and paste the text for their own use. To support random access and navigation, the slide view also has a gallery feature similar to PowerPoint's slide sorter view (shown in Figure 5) that allows the thumbnails of 20 slides to be shown at once.

For the document view, the document is divided into sections, each representing top-level headers (e.g., introduction, related work, etc.). This is done automatically by analysis of the text paragraphs of the document: short lines, leading sequentially numbered digits, upper case and typical keywords are used to identify the headers. To determine which slide thumbnails should appear with the section, SlideSeer examines whether any paragraph aligned to a slide appears in the displayed section. If so, the slide is displayed as a thumbnail. Note that this policy is symmetrical to the slide view; a slide may appear in two or more adjacent sections, just as paragraphs may be aligned to multiple slides.

Switching between these two views is also synchronized at the fine grained level. Activating the document view while in slide view accesses the document section that encloses the first aligned paragraph; if a slide has no aligned text, the document view is set to the beginning of the document. Switching to slide view from document view is similar – the first slide shown as a thumbnail for the section is accessed; if a section has no aligned slide, the slide view is set to the beginning of the presentation (slide 1). Clicking on a slide thumbnail in document view likewise shows the corresponding slide in slide view.

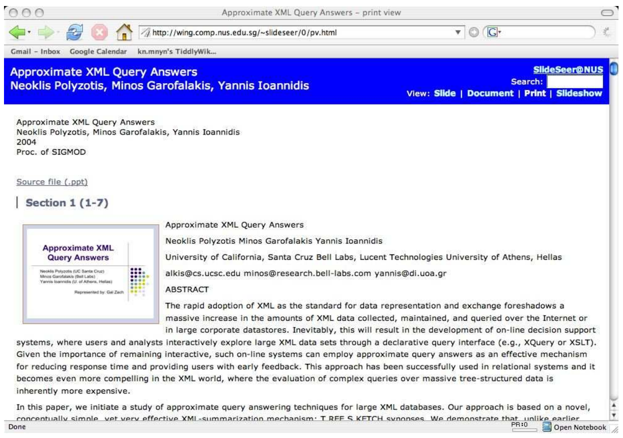
This fine-grained level of synchronization weaves both media into a single medium where the effort to switch focus and context is minimal. A user can view the details in the paper, or zoom out to the presentation to review the work's "big picture". Figures and text naturally complement each other, so media switching allows the user to enlarge the information for inspection, while maintaining the context of the other medium.

The current alternative to SlideSeer is to coordinate such media manually. In some cases, such media exists. The inspiration for SlideSeer are some of the course notes from MIT's OpenCourseWare, which offers notes as PDFs of the course slides accompanied with an aligned narrative notes[5].

In many cases, learning and comprehension is easier on paper, when annotation and markup is easy to do. To support this, a properly designed printed view is necessary. Neither the slide nor document views satisfy this requirement as they only present a slice of the medium in question, and multiple print commands would be necessary. Our solution is to offer a printed view, shown in Figure 6.

Any view that will be printed needs to cater to possible uses of the hardcopy. At the top of the printed view, the system approximates the number of pages needed to print the document using JavaScript and web page introspection. This needs to be calculate at run time due to possible differences in browser font sizes, although the printed view is a static page that is generated offline. In addition, the generated date and version number of the alignment (in case the alignment software is changed or manually corrected) is shown to help to reconcile any potential differences between

[5]http://ocw.mit.edu/


Approximate XML Query Answers – print view
http://wing.comp.nus.edu.sg/~slideseer/0/pv.html
Gmail – Inbox
Google Calendar
kn.mnyn's TiddlyWik...
SlideSeer@NUS
Search:
View: Slide | Document | Print | Slideshow
Done
Open Notebook

Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis

Approximate XML Query Answers
Neoklis Polyzotis, Minos Garofalakis, Yannis Ioannidis
2004
Proc. of SIGMOD

Source file (.ppt)

Section 1 (1-7)

Approximate XML Query Answers

Neoklis Polyzotis Minos Garofalakis Yannis Ioannidis

University of California, Santa Cruz Bell Labs, Lucent Technologies University of Athens, Hellas

alkis@cs.ucsc.edu minos@research.bell-labs.com yannis@di.uoa.gr

ABSTRACT

The rapid adoption of XML as the standard for data representation and exchange foreshadows a massive increase in the amounts of XML data collected, maintained, and queried over the Internet or in large corporate datastores. Inevitably, this will result in the development of on-line decision support systems, where users and analysts interactively explore large XML data sets through a declarative query interface (e.g., XQuery or XSLT). Given the importance of remaining interactive, such on-line systems can employ approximate query answers as an effective mechanism for reducing response time and providing users with early feedback. This approach has been successfully used in relational systems and it becomes even more compelling in the XML world, where the evaluation of complex queries over massive tree-structured data is inherently more expensive.

In this paper, we initiate a study of approximate query answering techniques for large XML databases. Our approach is based on a novel,

Figure 6: Excerpt of Print View (pv).

the online version and past hardcopies. Slide thumbnails are also made larger than in the document and gallery views, with the goal of the normal smallest font size used in presentations (18 points[6]) being legible on the print view (at least 6 point size). This difference is important as in the online environment, users have the option of switching to slide view to view the slide in detail; in a hardcopy no such option is possible.

While the learner is supported in the above views, other views are necessary to support teachers. A slide show view allows a presenter to easily navigate the slides within a presentation without the assistance of other software outside of the client web browser. The slide show view maximizes the size of the slide to the dimensions of the browser, while preserving the aspect ratio of the original slide. In slide show view, the navigation controls are simplified to two hyperlinks "<<" and ">>" to minimize interference. This slide show interface is also viable on small screen mobile devices. Figure 7 shows the slide show interface on the simulator of Opera Mini, a popular browser for small screen devices.

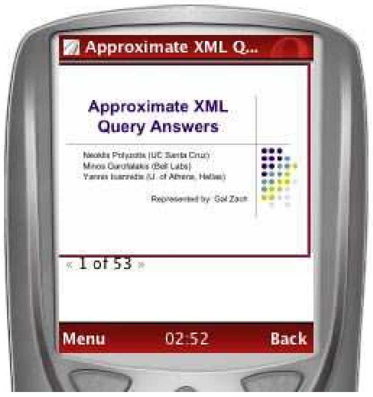


Figure 7: Slide Show View in Opera Mini.

5.2 Collection Interface Details

[6]Many references recommend the smallest font size on slides should be 20 or 24 points; but in practice this is often abused, thus we have set our limit to 18 points.

In our work on SlideSeer, we have concentrated on designing the coordinated media views. Although SlideSeer is not yet public, we believe most users wil be driven to the site by search engines searching by record metadata: document titles and author names. This claim is supported by usage patterns of our mirror of CiteSeer. If users have more than a casual interest, they may stay in SlideSeer to search and browse additional indexed pairs. We end by touching upon how we handle these collection-wide considerations, although these aspects are current undergoing significant development and change.

Searching. SlideSeer uses the open source Lucene indexing engine to drive its site search. We index only the text of the static print view, adding the title slide of each hit to aid visual memory. This interface is shown in Figure 8. Our current development focuses on replacing our document-level index with a finer grained, per-section and per-slide level retrieval. However, post processing is necessary to limit the number of hits per pair to ensure a single pair does not swamp all the hit ranks.

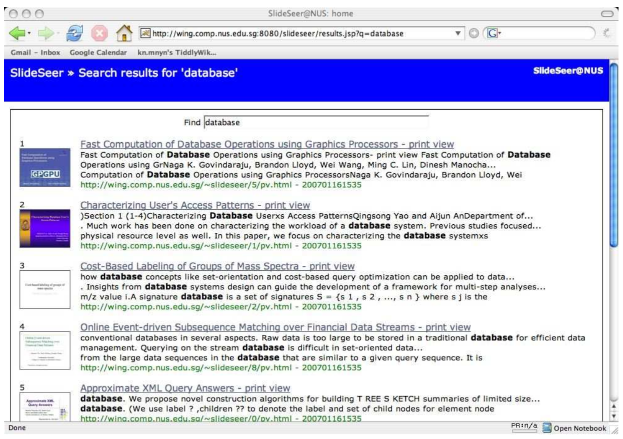
Spidering. As we expect search engines to be the primary access point for SlideSeer, spider accessibility to the system is critically important. Although exporting metadata in OAI or Google Sitemaps format will satisfy most indexers, we need to cater to the lowest common denominator. As Javascript is used dynamically to load and populate the views with the appropriate text and images, it is difficult for spiders appropriately load and analyze the coordinated media views. However, the print view is generated entirely offline as static HTML with this in mind.

URLs. Properly constructed URLs also serve as an access point as well as a relevance indicator. URLs need to be indicative of their content yet short enough to be displayed. The relative path component of SlideSeer URLs take the form of *subject/surname/year/title/ viewtype?offset*. Our path construction is justified by Vo's study on interdisciplinary citation patterns [17], which finds author, year, title metadata the most frequently used fields. More specifically, "subject" is a two-part subject code (e.g., *CS/DL*), followed the (truncated) surname of the first author, a two digit year code, "title" gives a similar systematic truncation of the title, and "offset" allows for random access to particular slides or sections (e.g., *d=4* for document section 4).

Keystroke shortcuts. When users become familiar with the interface, keyboard shortcuts become useful to enhance access. SlideSeer traps keyboard actions to navigate between and within views. Immediate help is accessible by moving the mouse over the "SlideSeer" anchor, revealing a tooltip that shows all available keyboard and mouse shortcuts.

6. CONCLUSION

We have discussed our current system work on SlideSeer, comprising of three components. Together, these components enable a fine-grained, synchronized viewing of scholarly contributions in the form of presentation and document pairs. In this work, we discuss the discovery, alignment and



SlideSeer » Search results for 'database'

Find database

1 [Fast Computation of Database Operations using Graphics Processors - print view]
Fast Computation of **Database** Operations using Graphics Processors- print view Fast Computation of **Database** Operations using GrNaga K. Govindaraju, Brandon Lloyd, Wei Wang, Ming C. Lin, Dinesh Manocha...
Computation of **Database** Operations using Graphics ProcessorsNaga K. Govindaraju, Brandon Lloyd, Wei
http://wing.comp.nus.edu.sg/~slideseer/5/pv.html - 200701161535

2 [Characterizing User's Access Patterns - print view]
)Section 1 (1-4)Characterizing **Database** Userxs Access PatternsQingsong Yao and Aijun AnDepartment of...
. Much work has been done on characterizing the workload of a **database** system. Previous studies focused...
physical resource level as well. In this paper, we focus on characterizing the **database** systemxs
http://wing.comp.nus.edu.sg/~slideseer/1/pv.html - 200701161535

3 [Cost-Based Labeling of Groups of Mass Spectra - print view]
how **database** concepts like set-orientation and cost-based query optimization can be applied to data...
. Insights from **database** systems design can guide the development of a framework for multi-step analyses...
m/z value i.A signature **database** is a set of signatures S = {s 1 , s 2 , ..., s n } where s j is the
http://wing.comp.nus.edu.sg/~slideseer/2/pv.html - 200701161535

4 [Online Event-driven Subsequence Matching over Financial Data Streams - print view]
conventional databases in several aspects. Raw data is too large to be stored in a traditional **database** for efficient data management. Querying on the stream **database** is difficult in set-oriented data...
from the large data sequences in the **database** that are similar to a given query sequence. It is
http://wing.comp.nus.edu.sg/~slideseer/8/pv.html - 200701161535

5 [Approximate XML Query Answers - print view]
database. We propose novel construction algorithms for building T REE S KETCH summaries of limited size...
database. (We use label ? ,children ?? to denote the label and set of child nodes for element node
http://wing.comp.nus.edu.sg/~slideseer/0/pv.html - 200701161535

Figure 8: Lucene Search for "database" in SlideSeer (search).

presentation of these media pairs which have largely been untouched by the DL community.

As such, SlideSeer's fine grained alignment offers affordances that previously was only possible through manual work. We believe SlideSeer users will find reviewing, learning and using the indexed scholarly work more accessible. Consideration was made to cater for users with different roles and needs in the user interface design. We hope that by collecting media pairs in a centralized DL, economies of scale in searching and browsing such pairs will be realized.

Our current usability testing aims to validate these claims. Aside from this, research and development continues on the DL. Research goals include improving both alignment and resource discovery as well as integrating other subdocument analysis modules such as keyphrase finding and citation processing. For development, our immediate goals are to increase coverage of pairs – by processing more DBLP+CiteSeer metadata records as well as including presentations in PDF format. Adding linkages with existing DL resources via DOI and OpenURL are also being considered.

7. ACKNOWLEDGMENTS

We would like to thank Eugene Ezekiele for help in creating the gold standard data for alignment and the PPT text extraction utility; Jin Zhao for his help in merging the CiteSeer and DBLP records. Special thanks to Hidetsugu Nanba and Tessai Hayama for sharing their gold standard collection and text extraction utility with us. Finally, thanks C Lee Giles and Isaac Councill for their generosity in sharing the CiteSeer data and programs, and assisting in the setup of our mirror site.

8. REFERENCES

[1] *2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, Edinburgh, UK, 2005.

[2] P. Brown, J. C. Lai, and R. L. Mercer. Aligning sentences in parallel corpora. *Proc. of Association for Computational Linguistics (ACL)*, 15(5):745–770, 1991.

[3] M. Davis. *Scientific Papers and Presentations.* Academic Press, 2005.

[4] W. A. Gale and K. W. Church. A program for aligning sentences in bilingual corpora. *Proc. of Association for Computational Linguistics (ACL)*, 1991.

[5] T. Hayama, H. Nanba, and S. Kunifuji. Alignment between a technical paper and presentation sheets using a hidden markov model. In *Active Media Technology*, pages 102–106. IEEE Press, 2005.

[6] G. Hoff and M. Mundhenk. Finding scientific papers with HPSearch and MOPS. In *SIGOC '01*, 2001.

[7] Y. Huang and G. Madey. Web data integration using approximate string join. In *Proc. of Int;l World Wide Web Conf. Alternate track papers & posters*, pages 364–365, New York, NY, USA, 2004. ACM Press.

[8] H. Jing. Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics*, 28(4):527–543, 2002.

[9] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM Int'l Conf. on Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[10] D. Mekhaldi, D. Lalanne, and R. Ingold. Using bi-modal alignment and clustering techniques for documents and speech thematic segmentations. In *Proc. of the Int'l. Conf. on Info. and Knowledge Management (CIKM)*, pages 69–77, New York, NY, USA, 2004. ACM Press.

[11] D. Mekhaldi, D. Lalanne, and R. Ingold. From searching to browsing through multimodal documents linking. In *Int'l Conf. on Document Analysis and Recognition (ICDAR '05)*, pages 924–929, Washington, DC, USA, 2005. IEEE Computer Society.

[12] A. Meyers, M. Kosaka, and R. Grishman. A multilingual procedure for dictionary-based sentence alignment. *Proc. of AMTA'98: Machine Translation and the Information Soup*, pages 187 – 198, 1998.

[13] M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *National Conference on Artificial Intelligence (AAAI)*, 2006.

[14] B.-W. On and D. Lee. PaSE: Locating online copy of scientific documents effectively. In *ICADL 04*, pages 408–418, 2004.

[15] A. E. Papathanasiou, E. P. Markatos, and S. A. Papadakis. PaperFinder: A tool for scalable search of digital libraries. In *WebNet '98*, 1998. Poster Paper.

[16] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pages 260–267, April 1967.

[17] T. A. Vo. Support for annotation of scientific papers. Technical report, National University of Singapore, 2005.

[18] Y. Wang, M.-Y. Kan, T. L. Nwe, A. Shenoy, and J. Yin. LyricAlly: automatic synchronization of acoustic musical signals and textual lyrics. In *ACM Int'l Conf. on Multimedia*, pages 212–219, New York, NY, USA, 2004. ACM Press.

[19] D. Wu. Aligning a parallel english-chinese corpus statistically with lexical criteria. *Proc. of Association for Computational Linguistics (ACL)*, pages 80 – 97, 1994.