RC25026 (W1007-092) July 27, 2010
Computer Science

IBM Research Report

Outline Wizard: Presentation Composition and Search

Lawrence Bergman, Jie Lu, Ravi Konuru, Julie MacNaught, Danny Yeh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Outline Wizard: Presentation Composition and Search

Lawrence Bergman, Jie Lu, Ravi Konuru, Julie MacNaught, Danny Yeh
IBM T.J. Watson Research Center
Hawthorne, NY USA
{bergmanl, jielu, rkonuru, jmacna, dlyeh}@us.ibm.com

ABSTRACT
Assembling electronic presentations from existing presentation material is a commonly-performed task. Yet current tools provide inadequate support – search tools are unable to return individual slides, and the linear model employed by presentation creation tools lacks structure and context. We propose a novel method for presentation creation, implemented in a tool called Outline Wizard, which enables outline-based composition and search. An Outline Wizard user enters a hierarchically-structured outline of a presentation; using that structure, the tool extracts user requests to formulate contextual queries, matches them against presentations within a repository, taking into account both content and structures of the presentations, and presents the user with sets of slides that are appropriate for each outline topic. At the heart of Outline Wizard is an outline-based search technique, which conducts content search within the context derived from the hierarchical structures of both user requests and presentations. We present a heuristic outline-extraction technique, which is used to reverse engineer the structures of presentations, thereby making the structures available for our search engine. Evaluations show that the outline-extraction technique and outline-based search both perform well, and that users report a satisfying experience when using Outline Wizard to compose presentations from libraries of existing material.

Author Keywords
Presentation composition, presentation search, outline-based search, context-sensitive information retrieval

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces — User interaction styles, User-centered design; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval — Query formulation, Search process

1. INTRODUCTION

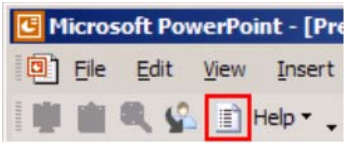
Presentations, created and presented using software tools such as Microsoft PowerPoint[1] and OpenOffice Impress[2] are widely used, with millions produced each day [9]. In a series of informal interviews with corporate executives and managers, we have discovered that creating new presentations by assembling slides from previously existing presentations is a common practice. For example, slides from presentations of individual products may be needed for inclusion in a marketing presentation; slides from presentations of various projects may be required for a management report.

Using today's tools, collating slides into new presentations is a painful process. The user must first *search* for the slides. Current search tools are unable to operate at the level of individual slides, which causes two problems. First, because the search is at the presentation level, any presentation containing all search terms anywhere within it will be returned, even though the objective may be to obtain a single slide containing all search terms. Second, the user must sift through entire presentations returned by the search to find and extract relevant slides, which is often a time-consuming and difficult task.

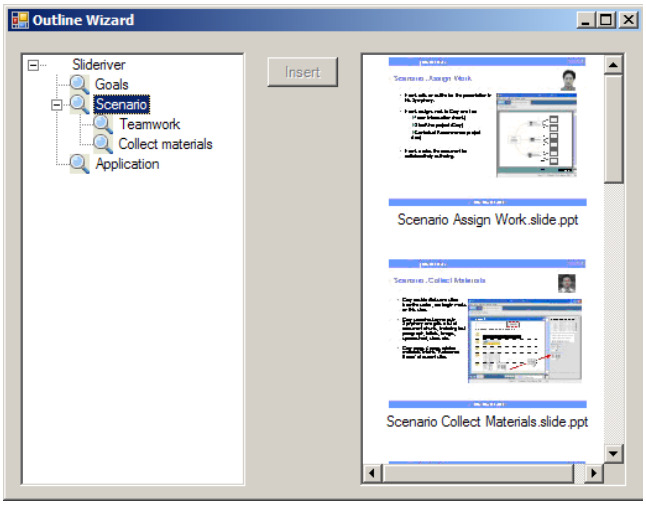
In an attempt to address these issues, our group has developed a slide-level presentation repository, known as *SlideRiver*. Presentations (in PowerPoint or OpenOffice formats) are uploaded into SlideRiver, which automatically indexes individual slides within presentations. These slides can then be browsed, shared, and searched.

Simply providing a slide-level search facility, however, is not a panacea. First, presentations often include slides whose content does not contain sufficient context for slide-level search. Consider searching for a slide that contains "*goals*" for the "*SlideRiver*" project. Presentations on SlideRiver may contain slides that describe goals (and contain the word "*goals*"), but do not have the word "*SlideRiver*" in their content. As a result, these slides may not be considered relevant when judged on their content alone without considering context, i.e., the presentations they come from. An additional complication is introduced when the desired material for a given topic spans multiple slides. For instance, a scenario or use case may consist of a

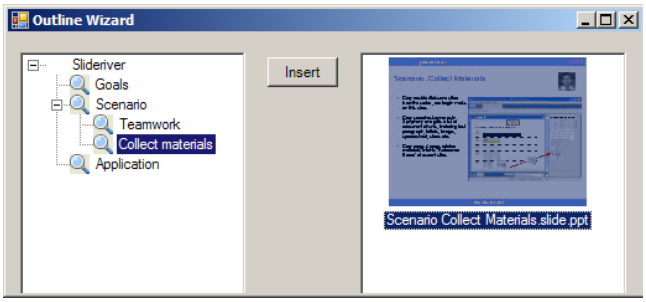
[1] http://office.microsoft.com/en-us/powerpoint/default.aspx
[2] http://www.openoffice.org/product/impress.html



(a)



(b)



(c)

Figure 1. Outline Wizard user interface

sequence of slides, but the search terms "*scenario*" or "*use case*" may not be present on all slides of the sequence. A slide-level search method that lacks knowledge of presentation structure will be incapable of identifying and returning relevant groups of slides under such circumstances.

Once slides are located, the new presentation must be *composed*. Composition consists of two portions – the structure of the presentation must be designed, and materials must be created/inserted into the structure. Current tools provide almost no support for designing presentation structure. Users often structure presentations hierarchically – this can be seen in the large number of presentations that begin with an agenda or outline slide. Yet, most of today's tools represent a presentation solely as a linear sequence of slides. Insertion of materials from multiple sources is typically accomplished by the laborious

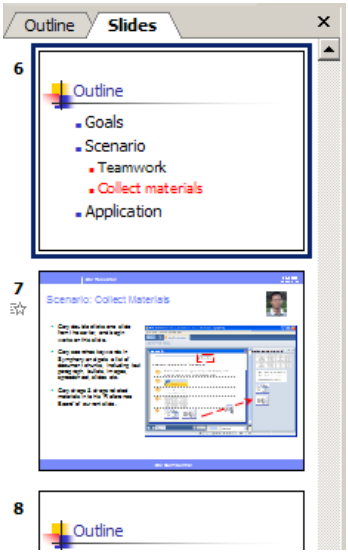


Figure 2. Presentation composition using Outline Wizard

process of opening each source presentation, then cutting-and-pasting between source and target using separate windows for each.

Our work attempts to provide support for search and composition through a new model of creating presentations from existing slides. Based on the common practice of structuring presentations via outlines, we present a methodology that unifies search and composition. In our system, entitled *Outline Wizard*, a user creates an outline, using text editing operations, which defines the structure of a presentation. As she does, a query is constructed at each level of the hierarchical outline, with nested context from the outline used to scope the query. To address the shortcomings of single-slide search discussed earlier, we employ a novel outline-based search technique. The technique matches scoped queries against sets of existing presentations to find candidate slides or groups of slides by considering both presentation content and structure. Since the structure of currently existing presentations is not typically available, we introduce an outline-extraction technique to reverse engineer presentation structures that can be used for search.

As an example of using Outline Wizard for presentation composition and search, we consider a user who wants to construct a presentation detailing the SlideRiver software system. She creates an outline (shown in Figure 1 (b) and (c)) that includes the title of the presentation – "*SlideRiver*," topics – "*Goals*," "*Scenario*," "*Application*," and subtopics (of "*Scenario*") – "*Teamwork*," "*Collect materials*." The hierarchically structured outline provides context used for search. In Figure 1(c), for example, the highlighted topic specifies that a "*Collect materials*" *scenario* is being sought for *SlideRiver*. Outline Wizard derives contextual queries based on the outline, conducts search over a repository of

presentations, associates sets of search results with these contextual queries, and supplies them in-context. By clicking on a topic within the outline, the user is able to preview the content of retrieved slides associated with that topic (Figure 1(b)). Outline Wizard automatically constructs a series of outline slides for the presentation based on the user-specified outline. The user can select particular slides from the search results for inclusion at the appropriate place in the presentation, as shown in Figure 2.

Compared to existing tools, Outline Wizard offers users three distinct benefits. First, it provides a mechanism for the user to design a presentation using a much more structured representation than that provided by traditional linear presentation tools. Second, Outline Wizard automatically formulates queries and conducts search based on the outline specified by the user, which frees her from manually crafting and issuing multiple queries to search for content. Third, Outline Wizard allows the user to easily inspect search results, and incorporate selected results into the presentation without cutting-and-pasting between multiple windows.

Below, we briefly discuss related work, followed by an overview of Outline Wizard. We then describe Outline Wizard's key components and our evaluation.

2. RELATED WORK

Our work is related to several systems that extend the typical linear presentation model to support hierarchical or graph-based representations of presentation structures [1, 6, 8]. We go beyond these systems by addressing not only how to organize presentation materials, but also how to locate them through search. [5] addresses the issue of composing presentations from multiple versions of a particular presentation; our focus is on assembling presentations from heterogeneous sources.

Chen [3, 4] presents a system that uses extracted presentation structures to search a PowerPoint database. It focuses on finding particular items (slides, diagrams, images, etc.) with queries specified via a generated ontology. In contrast, our system allows the user to specify the complete presentation structure via free-text, and automatically formulates queries from the structure.

XML information retrieval matches "content and structure" (CAS) queries, which specify requested or required content structure, against Extensible Markup Language (XML) documents. In outline-based search, contextual similarity plays a bigger role than structural similarity in determining the relevance of a slide to a user request. In spite of this difference between outline-based search and XML retrieval, our work is inspired by the body of research in XML retrieval, particularly the work on extending the vector space model for structured-based search [2, 7].

Our outline-based search technique creates and uses context vectors to conduct context-sensitive information retrieval, which is related to work on context-aware, adaptive information retrieval [10]. Context-aware retrieval uses context vectors, created from a graph of user actions which represents the user's investigative context, to augment explicit user queries. In comparison, outline-based search creates context vectors based on user-specified outlines and presentation structures, and these context vectors are used to inform context-sensitive search.

3. SYSTEM DESCRIPTION

Here we provide an overview of Outline Wizard, starting with its user interface, followed by its system architecture.

User Interface

Outline Wizard was developed as a PowerPoint plug-in. When the user clicks the Outline Wizard toolbar button (highlighted in Figure 1(a)), a new PowerPoint presentation is initiated, and the Outline Wizard user interface is displayed (Figure 1(b)). The user enters a presentation outline in the panel on the left. The top-most item is the presentation title ("*SlideRiver*" in Figure 1), with presentation topics and subtopics contained in a nested tree structure. The outline tree is editable; with tree items indented or dedented via keystrokes.

As the user completes entry of each topic, a search is initiated; when results are obtained, a spyglass icon is presented to the left of the topic. In Figure 1(b), search results returned for the highlighted topic "*Scenario*" are presented in the right-hand panel as thumbnails, each representing a single PowerPoint slide. The user can select any of the slides, as shown in Figure 1(c), and then insert that slide into the PowerPoint presentation, by clicking on the "Insert" button. The user can see a larger preview of a slide thumbnail by double-clicking on it; the preview panel also contains an "Insert" button. Figure 2 shows a portion of the PowerPoint user interface, containing the presentation being constructed. Outline slides are automatically inserted for each topic, showing the current topic ("*Collect materials*" in Figure 2) highlighted within the full outline context. Slides that have been inserted are displayed immediately following the topic they represent.

System Architecture

Figure 3 shows the architecture of the Outline Wizard system. It has two subsystems comprised of five main components: a front end containing an input processor and output processor, and a back end containing an outline processor, outline searcher, and outline extractor.

As the user types a presentation outline into the Outline Wizard user interface, the *input processor* processes user input and creates an XML-based representation of the outline to send to the outline processor. Given the outline, the *outline processor* first constructs and updates a

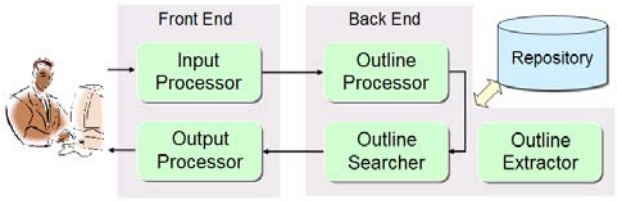


Figure 3. Outline Wizard architecture

hierarchical tree structure to represent the outline. Next it extracts content and context information from the hierarchy to formulate contextual queries. The *outline searcher* matches each query against context-sensitive representations of presentation content. Query results are passed to the *output processor*, which displays sets of results and supports user interaction with them.

The *outline extractor* is responsible for reverse engineering presentation outlines which are used for creating context-sensitive representations of presentation content. It reads and parses PowerPoint presentations stored in the repository, and infers an outline structure for each based on a variety of heuristic rules. Outline extraction is executed during an offline process.

4. OUTLINE EXTRACTION

Outline extraction is the process of reverse engineering an outline using content contained within an existing presentation. We began development of this component with an informal study of presentation structure. We examined about 100 presentations collected from colleagues. The presentations included project reviews and summaries, proposals, and technical materials. We noted a number of regularities in many of the presentations, including repeated structures (e.g., multiple slides in sequence with identical or similar titles indicating a slide group), and explicit organizational aids (e.g., introductory outline or agenda slides). Based on this examination, we initially focused on presentations containing agenda slides (slides typically titled "outline," "agenda," "table of contents," "roadmap," or "overview;" we will use the term *agenda* to refer to all of these).

We randomly selected 755 PowerPoint presentations from a large corporate information repository containing sales and marketing presentations. We developed an *automated agenda extractor* which identified 229 (30%) of these as having agenda slides. We randomly selected 100 of these agenda-containing presentations for use as a development set. A more detailed examination of these 100 yielded the results displayed in Table 1. This table displays both the results discussed here (in the column labeled "Development"), and the breakdown of a test set used for evaluation.

	Development	Test
Non-structural	**11**	**14**
Single agenda slide	**55**	**46**
Multiple agenda slides	**34**	**40**
Color/bold for current topic	(20)	(22)
Box/highlight for current topic	(9)	(11)
Other schemes for current topic	(5)	(7)
Total	**100**	**100**

Table 1. Types of agenda slides contained in presentation samples

11% of the presentations had agenda slides that on inspection clearly did not indicate the structure of the presentation, but instead presented workshop timetables, gave product highlights, etc. Although we were able to distinguish these non-structural agenda slides from those that reflect presentation structure, we felt it would be difficult for an algorithm to do so.

55% of the presentations contained a single agenda slide near the start of the presentation. Often the agenda topics matched the titles of particular slides within the presentation, either exactly or near-exactly. In other cases, the agenda might contain the topic (e.g., "*Why it sells*") with the associated slides containing titles that give specific reasons, but with no keyword matches.

34% of the presentations contained multiple agenda slides. Typically, these multiples were replicas of the agenda slide marking the start of each topic. Often, the current topic would be highlighted, usually by changing the color or by bolding the font (59% of multiple-agenda presentations), or by visually changing the background via a box or background highlight (26% of multiple-agenda presentations).

Based on these results, we developed an *outline inference module*. Figure 4 shows a display of output from the module. The leftmost panel displays slide titles, one line per slide. The middle panel shows the topics on the agenda slide. The rightmost panel shows the inferred outline. Topics from the agenda slide have become group titles (shown in blue), with each group containing zero or more slides (shown in red).

The inference module extracts topics from an agenda slide, then assigns individual slides to agenda topics using a segmentation-based algorithm, which assumes that slides appear in the same order as agenda topics (usually, but not universally true). The segmentation algorithm seeks to find a starting slide for each topic, and assumes that all slides that follow belong to the topic, until the slide that starts the next topic. Note that this approach allows hierarchically nested topics.

For a presentation with a single agenda, the correspondences between slides and agenda topics are determined by matching agenda topics with slide titles

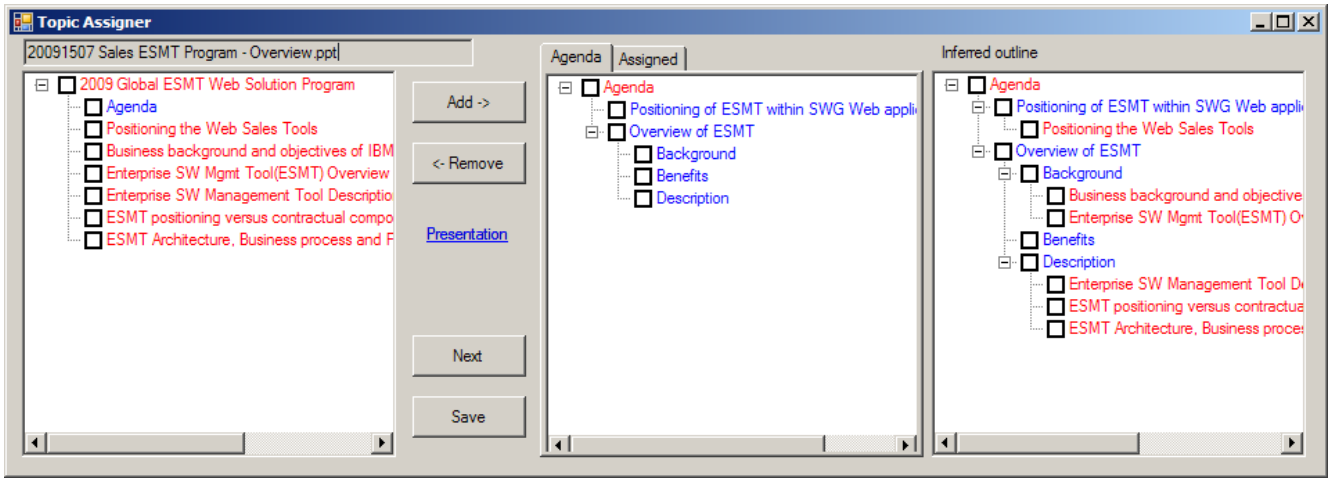


Figure 4. Outline extraction interface, showing slide titles, agenda topics, and extracted structure

based on the keywords extracted from each. Keywords are stopped with a stopword list derived from Lucene[3] and stemmed using the Snowball package[4]. Quoted strings are retained intact. A match score M between a slide title S and an agenda topic A is computed as the percentage of keywords from the slide title found in the topic:

$$M(S, A) = |K_s \cap K_a| / |K_s|$$

where K_s is the set of keywords in the slide title, and K_a is the set of keywords in the agenda topic. Any value of $M(S, A)$ that exceeds an empirically determined cutoff level is considered a match.

When there are multiple identical or near-identical agenda slides in a presentation, the inference module uses these slides to segment the presentation; each marks the start of a topic. The topic associated with each agenda slide is identified by recognizing color or bold highlighting. Details of this method are omitted due to space limitation.

If no color/bold highlighting is found, and the number of agenda slides is equal to the number of agenda topics, we assume a one-to-one correspondence between agenda slides and topics. Otherwise, the inference module ignores the multiple agenda slides, and segments the presentation via title matching as if it contains a single agenda slide, as described earlier.

5. OUTLINE-BASED SEARCH

In this section, we present our approach to outline-based search. First, we introduce the hierarchical tree-based representations used for modeling user-specified outlines and existing presentations. Second, we describe how we derive context-sensitive vectors from these representations to represent queries and presentation elements (e.g., slides or groups of slides). Third, we present the process that, given a user-specified outline, retrieves and ranks presentation elements based on their estimated relevance to the queries, and determines which slides to return.

Hierarchical Representations of Outlines and Presentations

Any outline, whether user-specified, or derived from an existing presentation, is represented by a hierarchical tree of *nodes*. For a user-specified outline, a node corresponds to one topic in the hierarchical outline, e.g., "*Scenario*," "*Teamwork*," etc. in Figure 1(b). We refer to these as *query nodes*, since they are used to automatically formulate searches. For an outline derived from an existing presentation in the repository, a node corresponds to a presentation element, which can be the entire presentation, a group of slides associated with a topic in the presentation outline, or a single slide. We refer to these as *repository nodes*.

The *content* of a node is determined by the *type* of the node. For a repository node representing a presentation, its content corresponds to the title of the presentation. For a repository node that represents a slide or a query node that represents an outline topic, its content is the text contained in the slide[5] or the topic. For a repository node that represents a group of slides, its content corresponds to the group title, which comes from the presentation outline topic with which these slides are associated.

The links between nodes in the hierarchical tree are determined by the parent-child relations as indicated by the outline structure. A top-level outline topic such as "*Overview of ESMT*" shown in Figure 4 is a child of the node that corresponds to the entire presentation. An outline data structure organizes all the nodes of a hierarchical outline, and provides methods for its navigation.

[3] http://lucene.apache.org/
[4] http://snowball.tartarus.org/

[5] We extracted text from all text areas within a slide, but not text embedded within graphics or images, nor did we use text within the notes areas.

The representation of a user-specified outline is created and updated dynamically as the user creates and edits the outline structure. The representations of presentation elements in the repository are created and indexed by an offline process and are loaded on demand at run time.

Context-Sensitive Vector Representations of Queries and Presentation Elements

We employ a vector space model to capture both content and context of query nodes and repository nodes. The *context* of a node is defined as the aggregate content of all of its ancestors and descendants in the hierarchical tree of nodes. A node's *context-sensitive vector* integrates the node's content with its context. It is created in two steps. First, the *content term vector* of the node is created, based on the content it encodes, without considering its context. Second, this content term vector is integrated with all content term vectors from the node's context to create its context-sensitive vector. Next we describe these two steps in detail.

We construct a node's content term vector by removing punctuation marks and stopwords, then stemming the set of words and quoted strings. For a query node, the weight of a term is determined by the term's frequency in the node's content. For a repository node, the weight of a term is computed based on its frequency in the node's content as well as its location and overall popularity in the presentation. Location refers to the hierarchical nesting level of a term, from inner to outer – slide content, slide title, outline topic, presentation title. Following the common practice of assigning location-based term weights in information retrieval, we give a higher weight to a term when it occurs at an outer level than when it occurs at an inner level in the hierarchy. Specifically, the location-based weight $w_{location}$ of a term t in a node n's content is set to 1.0 for the node's content that corresponds to presentation title, 0.8 for outline topic, 0.6 for slide title, and 0.4 for slide content. The simple linear weighting scheme is used due to the lack of training data for determining the relative importance of terms at different locations in a presentation.

A term's overall popularity is inversely related to its discriminative power, which is typically measured by inverse document frequency (*idf*) in traditional information retrieval. Because the basic result unit for outline-based search is a slide, we use inverse slide frequency *isf* to measure a term t's discriminative power within a presentation p:

$$isf(p, t) = log(N_p / N_{p,t})$$

where N_p is the total number of slides in the presentation p, and $N_{p,t}$ is the number of p's slides containing the term t.

The weight w of a term t in the content term vector $\mathbf{v_c}$ of the node n for a presentation element is therefore calculated as the product of the term's frequency f in the node's content, its location-based weight $w_{location}$, and its inverse slide frequency *isf* in the presentation to which the node belongs:

$$w(t) = f(t) \times w_{location}(t) \times isf(p, t)$$

To create a context-sensitive vector $\mathbf{v_s}$ for the node n, its content term vector $\mathbf{v_c}$ is integrated with all of the content term vectors from n's context as follows:

$$\mathbf{v_s}(n) = \mathbf{v_c}(n) + \Sigma_{n' \in context(n)}\, min(0, 1 - 0.2d(n, n')) \times \mathbf{v_c}(n')$$

where each content term vector $\mathbf{v_c}(n')$ from the context is discounted based on the distance (i.e., path length) $d(n, n')$ between its node n' and the targeted node n in the hierarchical tree, so that terms located closer to the targeted node are given higher weights. The discount factor of 0.2 is determined empirically.

As with the node representations, context-sensitive vectors that represent queries are created dynamically as the user edits the outline; vectors that represent presentation elements are created and indexed offline then dynamically loaded at run time.

Process of Outline-Based Search

As the user creates and edits an outline in the interface, the outline topics are dynamically sent to the outline processor, which updates the hierarchical representation created for the outline, extracts from it a set of nodes for topics that have new/changed content or context, and creates context-sensitive query vectors for these nodes. Each query vector is passed to the outline searcher, which conducts search in three steps. First, the query is sent to a Lucene text search engine, which uses the traditional *tf.idf*-based ranking

```
Input: the list of ranked presentation elements L_e
Output: the list of slides to return as the search result L_s

Procedure:
 foreach presentation element e in L_e
     if e.type == Slide
         L_s.add(e)
     else if e.type == SlideGroup || Presentation
         L = all the slides that belong to e
         foreach slide s in L
             if s.score < e.score
                 // boost the slide score
                 s.score = (s.score + e.score) / 2
             endif
             L_s.add(s)
         endfor
     endif
 endfor
 L_s = sort(L_s)
 foreach slide s in L_s
     // normalize the score to be between 0 and 1
     s.score = (s.score – min(L_s)) / (max(L_s) – min(L_s))
 endfor
 L_s = sub-list of L_s including all the unique slides with ranks
       higher than a cutoff c or scores greater than a threshold t
 return L_s
```

Figure 5. The method for determining which slides to return as the search result for a query

algorithm to rank its indexed presentations and returns a list of top-ranked presentations as candidates. Second, the outline searcher retrieves the context-sensitive vectors of the presentation elements contained in these candidate presentations, and estimates the relevance r of each presentation element e to the query q based on a combination of the standard cosine similarity Sim_{cos} between the query vector and the vector of the presentation element, the Boolean similarity Sim_{bool} between them, and the relevance score of the presentation p to which the presentation element belongs:

$$r(e, q) = Sim_{cos}(\mathbf{v_s}(e), \mathbf{v_s}(q)) \times Sim_{bool}(\mathbf{v_s}(e), \mathbf{v_s}(q)) \times r(p, q)$$

The Boolean similarity Sim_{bool} is calculated as the percentage of query terms that are matched. It is introduced to favor presentation elements that match all query terms. Third, the outline searcher ranks the presentation elements by their relevance scores, and generates a result list.

Currently the basic result unit for outline-based search is a slide. Figure 5 describes the method used by the outline searcher to determine which slides to return as the search result for the query. It uses the scores of the presentation elements at the level of presentation or slide group to boost the scores of the slides that belong to them, so that a slide is more likely to be returned when it belongs to a presentation or a slide group that is deemed relevant, even if this slide seems less relevant judged on its own. Slides which exceed a rank-based cutoff, c, or a score-based threshold, t, (both constants determined empirically) are included in a ranked list of return results.

6. EVALUATION

The evaluation consisted of three portions: 1) evaluation of the outline-extraction technique, 2) evaluation of the outline-based search, and 3) evaluation of the Outline Wizard user experience.

Outline Extraction Evaluation: Methodology

Outline extraction was initially evaluated on the development set of 100 agenda-containing presentations described in Section 4. Two of the authors (Bergman and Lu) independently assigned slides to agenda topics using a manual assignment process. The tool shown in Figure 4 allowed us to select a set of slides, select a topic (from the automatically identified agenda slide), and assign the slides to the topic. We ran the automated outline extractor on the same 100 presentations.

We devised an outline similarity metric S for comparing two outlines o_1 and o_2 representing a presentation p. It calculates the average degree of agreement between two outlines as follows:

$$S(o_1, o_2) = \sum_{s \in p} A(t_1(s), t_2(s)) / |p|$$

where for each slide s in p, $t_1(s)$ and $t_2(s)$ denote the agenda topics to which s is assigned in the two representations, A denotes the agreement between them, and $|p|$ denotes the number of slides in the presentation p.

A has a non-zero value if $t_1(s)$ and $t_2(s)$ are located on the same sub-tree in the topic hierarchy of p's agenda, with the degree of agreement discounted by a measure of their "distance" from each other. Specifically, it is computed as:

$$A(t_1, t_2) = 1 - min(1, 0.2 \times D(t_1, t_2))$$

where D is a measure of the "distance" between two agenda topics in the presentation agenda's topic hierarchy:

$$D(t_1, t_2) = max(d(t^*, t_1), d(t^*, t_2))$$

where t^* is the closest common topic to t_1 and t_2 among the set of agenda topics that includes t_1, t_2 and their ancestors in the topic hierarchy, and $d(\bullet, \bullet)$ is the distance (i.e., path length) between two topics. If t_1 and t_2 refer to the same topic, D is set to 0. The discount factor of 0.2 is determined empirically.

If a slide is assigned to a topic by the outline extractor but is left unassigned by manual assignment, A is set to 0.5. If a slide is assigned manually but is left unassigned automatically, A is set to 0.

For each presentation in the development set, we calculated similarity scores on three pairs – comparing the two manual assignments, and then comparing the automatic extract with each of the manual assignments.

Once we had completed development of the algorithm, we randomly selected an additional 100 agenda-containing presentations, and repeated this set of evaluation procedures.

Outline Extraction Evaluation: Results

Results comparing the assigned outlines are shown in Table 2. Note that the two human annotators (designated as Annotator 1 and Annotator 2) corresponded well (but not perfectly) in their assignments, both with the development set as well as the test set. The degree of agreement between the outline assignments of the automated outline extraction and both humans was satisfactory – about 70% for the development set, and about 60% for the test set, indicating the effectiveness of our outline-extraction technique.

The lower scores for the automated outline extraction can be attributed to several factors. First, there were errors in outline extraction for a small handful of presentations. This was primarily due to the algorithm looking for indented bullets, but being unable to recognize other form of indenting, such as tabs. Second, the algorithm did not handle all forms of structure marking within the presentation sets. In particular, some presentations with multiple agenda slides contained a different two-level structure on each agenda slide; our algorithm did not handle this case, and performed poorly. Finally, the keyword matching approach failed in some cases, particularly where the text in the agenda did not correspond well with the terms on the slide titles.

The lower scores on the test set (relative to the development

	Development set		Test set	
	Average	StDev	Average	StDev
Annotator 1 vs. Annotator 2	0.90	0.15	0.85	0.20
Annotator 1 vs. Automatic	0.71	0.25	0.60	0.28
Annotator 2 vs. Automatic	0.69	0.28	0.57	0.28

Table 2. Evaluation results of outline extraction

set) can be attributed to a lack of homogeneity between the two sets. First, the test set had a higher number of non-structural agendas (14 vs. 11). Second, the test set had a larger number of multiple-agenda presentations highlighted with outline boxes (11 vs. 9) which we are currently not handling. Finally, the test set had a number of multiple-agenda presentations containing agenda slides with more than two highlights per slide; these were quite a bit less common in the development set, and we did not handle these in our algorithm.

Outline-Based Search Evaluation: Methodology

We evaluated search on two separate sets of presentations, which came from two different organizations within our corporation – sales and development. The presentations within each set shared a common theme, e.g., providing a profile of a potential corporate customer, or documenting a development process. Each set of presentations was specified by a template, but individuals who created the presentations were free to either delete elements of the template, or add additional elements. The presentations for customer profiles contained explicit agenda slides. Although the development process presentations did not contain explicit agenda slides, we asked two colleagues who are somewhat familiar with this development process to manually construct agendas for eight of them.

We simulated two use cases of presentation composition and search, one for each data set. The first use case was composing a presentation to compile particulars (e.g., executive summary, industry analysis, company overview) of several potential corporate customers. The second use case was constructing a presentation to summarize common development aspects (e.g., functional description, design requirements) of several projects. We simulated a two-level user-specified outline for each use case, with the outer level specifying various particulars/aspects required for composing the presentation, and the inner level specifying names of different customers/projects. The outlines resulted in a total of 30 queries, 18 for the first use case and 12 for the second.

We added both sets of presentations to the presentation repository, and created context-sensitive vectors of presentation elements based on the presentation outlines generated by the outline extractor. Then we ran our outline searcher to obtain a set of top-ranked slides for each outline topic query. We manually compiled a list of relevant slides by inspecting the presentations about targeted customers and projects. For each query, we calculated accumulative precision and recall as well as F-measure (the harmonic mean of precision and recall) for the top 20 ranked slides. The average number of relevant slides per topic was 8 for the first use case and 2.75 for the second use case.

As a baseline, we used the slide-level search facility provided by the SlideRiver system, which employed a Lucene text search engine to index both presentations and individual slides from these presentations as documents. Given a query, the search engine retrieved documents ranked using Lucene's default *tf.idf*-based ranking algorithm. The Boolean *AND* query operator was applied to multiple query terms. We measured precision and recall for the baseline results using two different methods. The first method measured the performance of slide-level search and ignored presentation results. The second method expanded each presentation result to include all the slides from this presentation in the search result. Here we refer to the evaluation result using the first method as "*baseline-slide*" and that using the second method as "*baseline-all*".

We also compared *context-sensitive search* – outline-based search using context-sensitive vectors (created from presentation structures) with *content-only search* – standard slide-level search using only content term vectors of slides without context.

Outline-Based Search Evaluation: Results

Figure 6 shows evaluation results of outline-based search in both use case 1 (composing a presentation to profile potential corporate customers) and use case 2 (constructing a presentation to summarize projects under development). Figure 6(a) and Figure 6(c) plot precision against recall, averaged over all the queries in each use case. Figure 6(b) and Figure 6(d) depict the change in F-measure as the document (slide) rank increased from top 1 to top 20. Because on average the number of slide results returned by the baseline approach was very small (1.72 per query for use case 1 and 1.58 for use case 2) due to the Boolean query constraint, "baseline-slide" had fewer data points than the other methods.

The results indicate that outline-based *context-sensitive search* performed well in both use cases, no matter whether it was recall-oriented (use case 1, with an average of 8 relevant slides per query) or precision-oriented (use case 2, with an average of 2.75 relevant slides per query). By contrast, *baseline-slide* (which only considered slide results) yielded very low recall in recall-oriented use case 1, while *baseline-all* (which included individual slide results and all the slides from presentation results) had very low precision in precision-oriented use case 2, indicating that no single strategy worked well in both cases for the SlideRiver baseline system.

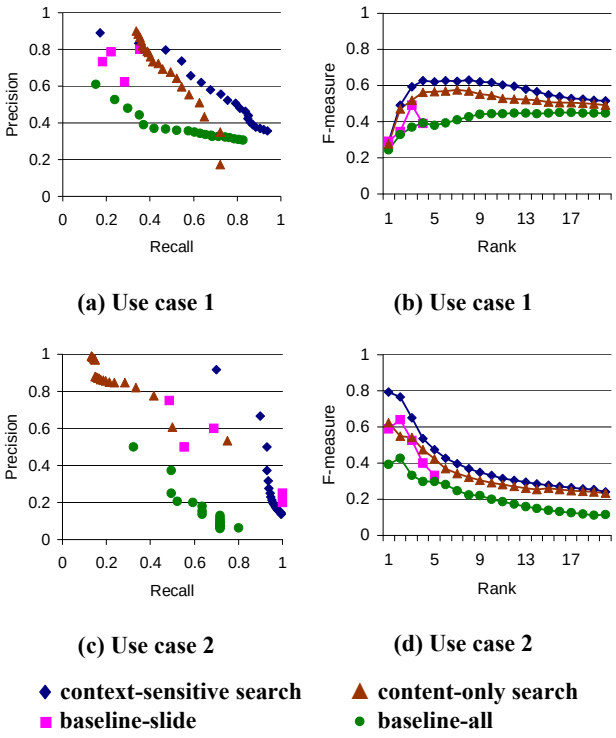


Figure 6. Evaluation results of outline-based search

For both use cases, *context-sensitive search* outperformed both the baseline and *content-only search*, demonstrating the promise of incorporating context information derived from outline structures for more effective slide-level search.

User Experience Evaluation: Methodology

We conducted an informal evaluation of the Outline Wizard user experience. We asked six people – five of them researchers or software engineers within our organization, one of them the retired head of corporate communications for a Fortune 50 company – to compose a presentation from existing materials using the Outline Wizard from within PowerPoint. Three of the participants were men, three women, all of whom use PowerPoint as part of their job. The repository used for this study was populated with the eight project development presentations used for evaluating outline-based search. The task involved constructing a presentation on a set of specified themes (e.g., "design requirements", "usage scenarios"), each theme to contain information on several projects. Thus, the task naturally contained a two-level information structure. After being given a short demo of Outline Wizard, the users were asked to use the tool to create a presentation outline and select slides. The users were observed using the tool, and then asked to complete a brief questionnaire. They were also asked to comment on the tool, and to offer suggestions for improvement.

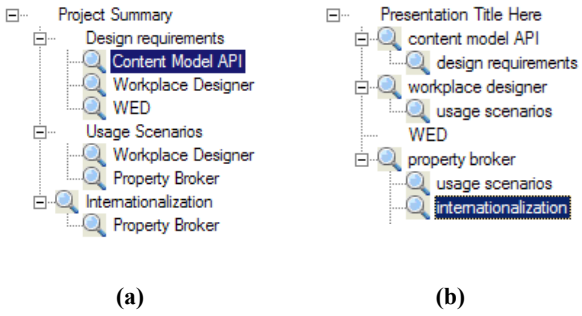


Figure 7. Different outline structures reflecting the same task

User Experience Evaluation: Results

The user study participants all successfully performed the task, with few difficulties. Four of the six produced hierarchical outlines. The remaining two produced flat structures, with all topics at the same level. When asked why, one had misinterpreted the instructions; the other stated that he had not read them carefully.

Three of the four participants that produced a two-level hierarchy, created one that very closely modeled the written structure of the task, which nested development projects within themes (such as "design requirements"). An example is shown in Figure 7(a). One participant re-factored the hierarchical structure – nesting themes within projects – to produce the outline shown in Figure 7(b). We noted that in both cases, similar sets of search results were returned, and the users were able to successfully complete the task. This indicates that our approach is not dependent on a particular hierarchical structure, but supports a variety of specifications.

We also noted two distinct working styles. Most participants typed in the outline, then when the outline was complete, went back and selected slides. A minority of the participants interleaved typing topics and selecting slides.

The users were all enthusiastic about Outline Wizard. Comments included, "*I like this very much*," and "*It would be very useful for big worldwide companies; it could save hours of time*." Several participants stated a belief that the tool would be particularly useful in sales, where assembling presentations to customers from libraries of presentation materials is a common practice.

Figure 8 shows results from the questionnaire. All questions received positive responses (numbers in parentheses are averages on a 1-5 Likert scale with 5 the most positive) – Q1: "*I could easily construct a presentation with Outline Wizard*" (4.3), Q2: "*The concept of outline-based search and composition is easy to understand and use*" (4.5), and Q3: "*I would use Outline Wizard for my own work, if available*" (4.3). The single response of "Undecided" for Q3 was from a participant who said that he always produces his presentations "from

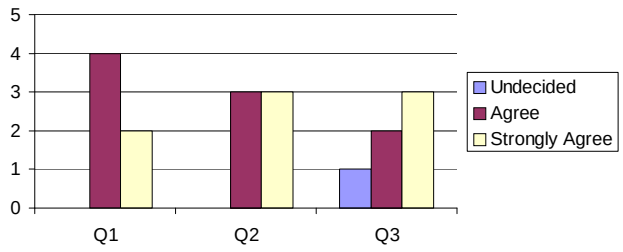


Figure 8. User responses to questionnaire items. Questions are documented in the text

scratch", so would not find a tool that supports reuse of value to him.

The users had a number of suggestions for improvements to Outline Wizard. These included: 1) visual indicators showing that slides had been inserted into topics, 2) user selectable options to control formatting for the generated outline slides, including level-of-detail control, 3) the ability to add additional search specifications (keywords, Boolean operators, scope, etc.) in addition to the topic titles, 4) support for easy reorganization of an outline, perhaps via drag-and-drop operations, and 5) adding social networking functions, such as display of ratings, tags, comments, for individual slides returned from the search.

7. CONCLUSIONS AND FUTURE WORK

We have presented an outline-based model for composition of presentations based on searching existing material. The user composes a presentation by specifying a hierarchically-structured free-text outline. The outline provides both search terms and contextual structure for a contextual outline-based search. The content to be searched is also represented hierarchically, by means of extracted outlines which are reverse engineered from the existing presentations. We have shown the outline extraction process to perform reasonably well on a random selection of presentations. Furthermore, we have shown the outline-based search technique to be effective at returning appropriate individual slides. Users of the Outline Wizard system, which embodies the outline-based search, successfully used it to create "new presentations from old", and were enthusiastic about the tool. Although the system as presented operates only on English-language text, there is nothing in either the extraction or search algorithms that is inherently language-dependent.

We have plans to extend this research in several directions. First, we wish to extend the outline extraction algorithm to handle presentations that do not contain agendas. We note that many presentations contain local regularity – for example, repeated keywords in sequences of slides – that could be used to extract local structure. Because our search algorithm is layered on a standard keyword search, partial structures, even very minimal ones, should yield an improvement. Second, we wish to support more than single slide results. In many cases, the results of a query should be a group of related slides, for example, several slides that compose a scenario. This would require identifying a tight relationship between these slides, delivering them together from the search engine, and providing the appropriate user interface elements for displaying and manipulating them. Related to this is our desire to support previously-created topics as searchable elements; the ability to search and compose using topics from a "topic library" would extend the power of the outline-based model. Finally, we would like to extend the outline-based model to include aspects of workflow typically encountered within an organization – enabling sharing, distribution, and collaborative editing via topics.

ACKNOWLEDGMENTS

We thank Rich Thompson and Robert Flavin of the SlideRiver team for suggestions and feedback, as well as development support. We also thank Jennifer Lai and Ramesh Gopinath for their support of this work.

REFERENCES

1. Apple Keynote '09, http://www.apple.com/iwork/keynote/
2. D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. "Searching XML documents via XML fragments." In SIGIR 2003.
3. C. Y. Chen, R. Ribier, and S. P. Liou. "An automatic poster summarization for microsoft powerpoint presentation." In IASTED EuroIMSA, 2005.
4. C. Y Chen. "An integrated system supporting effective indexing, browsing and retrieval of Microsoft PowerPoint presentation database." In 22nd Intl. Conf. on Data Engineering Workshops (ICDEW'06).
5. S. M. Drucker, G. Petschnigg, and M. Agrawala. "Comparing and managing multiple versions of slide presentations." In UIST 2006.
6. D. Holman, P. Stojadinović, T. Karrer, and J. Borchers. "Fly: An organic presentation tool." In CHI 2006.
7. S. Liu, Q. Zou, and W. Chu. "Configurable indexing and ranking for XML information retrieval." In SIGIR 2004.
8. T. Moscovich, K. Scholz, J. F. Hughes, D. H. Salesin. "Customizable presentations." Technical Report CS-04-03, Computer Science Department, Brown University.
9. I. Parker. "Absolute PowerPoint: Can a software package edit our thoughts?" The New Yorker, 2001.
10. Z. Wen, M. Zhou, and V. Aggarwal. "Context-aware adaptive information retrieval for investigative tasks." In IUI 2007.