is that a two or three word query does not communicate the context in which the student is trying to use the search engine. Thus, since general purpose search engines must have appropriate results for all users, they cannot provide the best possible results for a particular student in a specific educational context.

Our solution is to construct a specialized search engine for every academic course. We use a popular high school course, AP US History (APUSH)—taken by over 400,000 students in 2011 [**?**], as our target educational context and show how we can construct a search engine that captures the richness of the web while avoiding some of the problems associated with generic search engines. In order to run the search engine, we use the Google Custom search engine platform (`http://google.com/cse`) which allows us to restrict the corpus of our search engine to a set of url patterns or sites. The CSE platform takes care of the cumbersome tasks of crawling the web, building an index, and running the search engine. This has allowed us to not only build a search engine for APUSH, but to also make it widely available to students taking the course.

We first demonstrate the utility of this course-specific search engine by creating a reference search engine with a manually curated set of sites. In a blind side-by-side evaluation by domain experts (APUSH teachers), this search engine is overwhelmingly preferred to Google. However, manually curating thousands of sites is a time-consuming process and thus we develop two automated algorithms. In order to do so, we make the assumption there exists a textbook, or other authoritative source, which describes the course content.

We propose two algorithims that utilize such an authoritative source. The first evaluates sites based on their textual similarity to the textbook using TF-IDF weighted cosine distance. The second uses knowledge bases to identify APUSH-related topics, which are then used to identify APUSH-related sites. Our experimental results indicate that these algorithms significantly reduce the amount of manual work required to create a course-specific search engine.

1.1 Related work

There has been substantial work in the area of topic-specific ranking focused around exploiting the link structure of the web to identify clusters of documents that are on the same topic. For example, this structure can be exploited to create a topic-specific Page Rank for influencing ranking [**?**, **?**] or to influence the order in which pages should be crawled [**?**]. In contrast, our work uses the text content of an authoritative source (the course textbook) rather than link structure to define topic-specific ranking algorithms.

Focusing on the domain of search engines for the educational context, the most closely related systems are those of PubMed, Web of Science and Google Scholar, which search specialized corpora of scientific research [**?**]. Unlike these systems which include only peer-reviewed scientific articles, our course-specific search engines endeavor to include all of the useful educational material that can be found on the web.

We are aware of very little work on the use of knowledge bases to influence the results shown in search. Some exceptions are [**?**], which explores using ontologies for characterizing the user and [**?**] which show how search results can be augmented with snippets from a knowledge base.

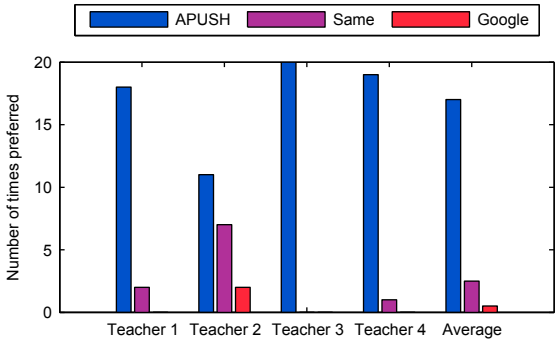

Figure 1: Side-by-side comparison of the APUSH search engine and Google

2. REFERENCE SEARCH ENGINE

In order to evaluate the relative performance of a search engine for the AP US History course, we first create a reference search engine based on a standard textbook [**?**], which is available in PDF form. As described above, we use the Google CSE platform which allows us to specify the corpus for our search engine as a set of url patterns.

If we knew all the queries that students will issue in the APUSH context, a brute force approach would be to curate every search result returned by Google for every query, manually picking the good sources. Clearly, this is not possible both because the queries are not known a priori and because of the magnitude of manual work that would be required. We therefore approximate this process by computing likely queries from the textbook and by curating at the site level rather than page level. This curated set of sites will then form the corpus of our reference search engine.

In order to compute likely queries, we observe that most history-related queries include one or more proper nouns (people, events, places, etc.). Thus, we use the digital version of the textbook [**?**] to extract proper nouns, using simple syntactic cues such as capitalization and punctuation. This method identifies 1206 distinct proper nouns, occurring a total of 11241 times in the text. We then form queries out of tuples of proximately occurring proper nouns and retrieve the top ten results from Google, resulting in 132,145 results from 23393 sites. There are 1757 sites which appear in at least 10 results and these account for 70.2% of all results. We manually examined each of these sites and classified them into two groups: those that were bad (either off-topic (e.g., trulia.com, yelp.com) or not appropriate for academic work (e.g., answers.yahoo.com, wikianswers.com)) and those that were good. 768 were off-topic or not appropriate, leaving us with 989 good sites; randomly selecting sites would thus result in 56% good.

We created a custom search engine using the good sites (available at `http://guha.com/apushcse.html`) and evaluated it on a collection of 20 APUSH-related queries included in Appendix **??**. We asked 4 history teachers to compare results from Google versus results from the APUSH search engine in a blind side-by-side test. They were asked to assign

one of the following ratings to each side-by-side: side A/B is better, side A/B are about the same. As can be seen in Figure **??**, the course-specific search engine is substantially preferred.

3. METHODS FOR AUTOMATION

Instead of manually labeling thousands of sites, our goal to create a course-specific search engine from the course textbook with minimal manual supervision. We wish to develop algorithms that distinguish between sites that produce off-topic results / sites that are inappropriate for academic usage and sites that may be used for academic work. Rather than attempting a binary classification of sites into "good" and "bad", we wish to rank sites based on the likelihood of them returning good results for APUSH searches. A search engine can then be configured to include just the top N sites or better, or to prefer sites proportional to their likelihood of returning good results in the APUSH context.

In order to automate the process of identifying these sites, we describe two methods for using the authoritative content provided by the textbook. The main intuition behind our algorithms is that by using the wealth of information provided by the textbook, we can drastically improve upon generic search scoring that is based solely on a two or three word query.

3.1 TF-IDF weighted text similarity

We begin with an approach that computes textual similarity between the textbook and a site via TF-IDF weighted cosine distance. Classical information retrieval is based on the notion of similarity between two pieces of text—typically, the query and the web page. Intuitively, by preferring sites that are more similar to the APUSH textbook, the search engine will be more likely to return better results for searches in the APUSH context.

One of the most commonly used similarity measures is the cosine similarity metric based on the vector space model of documents [**?**]. In this model, each document is a vector in an n-dimensional space in which each term in the corpus is a dimension. Using the popular TF-IDF weighting, the magnitude of the document vector along the ith axis is given by by a product of the term's frequency in the document (TF) and the inverse document frequency (IDF) of the term—the log of the inverse fraction of documents containing the term in the entire corpus. The cosine similarity between two documents in this space is given by the angle between the document vectors.

In order to compute the similarity of a site to the APUSH textbook, we first retrieve all pages from the site that are returned by Google for the queries used to create the reference search engine. After stripping the pages of HTML markup and javascript, we stem the words on each page (using the Porter stemmer [**?**]) and extract the terms from each along with their frequency of occurrence. We sum the document vectors across the entire site and compute the cosine similarity between the resulting site vector and that of the APUSH textbook.

3.2 Knowledge bases

Next, we consider an approaching using structured data from a knowledge base to identify APUSH-related categories, entities and proper nouns. The realization that most queries and web documents are about real word entities has led many of the major search engine providers to build various kinds of knowledge bases to augment their search results (for example, see [**?**]). In the academic sphere, there has also been significant work on the semantic web [**?**] and linked data [**?**], aiming to build a large distributed network of information about entities and the relations between them. In this section, we describe how knowledge bases can be used to identify topics related to APUSH. The identification of these topics can then be used improve ranking based on simple textual similarity.

First, we observe that some types of entities lead to more off-topic results than others. For example, places (e.g., Virginia, Maryland) are more likely to lead to results that are not about history compared to US presidents since the former will bring up real estate and local results. US presidents in turn are more likely to bring up off-topic results compared to Confederate generals, since the former are more likely to have institutions and places named after them. This relative preference (Confederate generals better than US presidents better than places) captures some of the APUSH context.

Our goal is to automatically identify the types that are more likely to give good results and give greater preference to the sites that contain these kinds of entities. In order to do this, we must first construct mappings from the proper nouns that we extract from the text to entities and from the entities to types of entities. To construct such mappings in a fashion that is not specific to history, we need a broad knowledge base about a large number of entities, along with information about the type of each entity. Wikipedia contains such information and DBpedia (available at `http://dbpedia.org/`) makes this information available as a structured knowledge base. In particular, each "thing" in Wikipedia corresponds to an entity in DBpedia and each "category" in Wikipedia corresponds to a DBpedia type. We will use DBpedia as the primary source of entities, types and for mapping proper nouns to entities and entities to types.

Now we consider the task of mapping proper nouns extracted from text to a set of candidate entities and types in DBpedia. In general, there are many different proper nouns that refer to the same real world entity—for example, President Lincoln, Abraham Lincoln, and Abe Lincoln all refer to the same person. Similarly, a given proper noun (e.g., Washington) could map to multiple different entities. We found that the most robust way of mapping from proper nouns to entities is to use search itself with the following method. For every proper noun P, we issue the query [P site:wikipedia.org] which returns a set of results with urls of the form "http://en.wikipedia.org/wiki/<entity-id>". We take the top 3 entities for each query as the candidate entities for the corresponding proper noun. Each entity may also have a number of categories associated with it—for example, the entity whose unique identifier is Abraham_Lincoln has 29 different categories associated with it: American_Presidents, Illinois_Lawyers, Assassinated_HeadsOfState, etc. We use the RDF dumps from DBpedia to construct the mapping from entities to categories.

Given these mappings from proper nouns to categories, we score each category according to the likelihood of a query with proper nouns in that category bringing up sites with on-topic results. The intuition behind the scoring algorithm is as follows. A course, such as APUSH, is about certain categories of entities and the relationships between them,

19th-century presidents of the United States
United States Presidential Candidates
Oneida New York
Presidents of the United States
Whig Party
Presidency of James Monroe
Christian denominational families

Table 1: DBpedia categories for APUSH

Andrew Jackson
Martin Van Buren
Thomas Jefferson
James Monroe
Abraham Lincoln
James Buchanan
Zachary Taylor
William Henry Harrison
Russia

Table 2: DBpedia entities for APUSH

for example American_Presidents. These categories should be assigned higher scores than categories that appear only incidentally, such as Illinois_Lawyers and Noble_titles. We would expect that a larger fraction of the entities in a category that the course is about will occur in the textbook compared to categories that appear incidentally. Thus we score a particular category with the ratio

$$CategoryScore = \frac{\#Textbook}{\#DBpedia} \quad (1)$$

where $\#Textbook$ and $\#DBpedia$ count the number of times entities in the category occur in the textbook and DBpedia, respectively.

For example, the textbook contains references to 33 entities in the category American_Presidents, which, in DBpedia is associated with 44 entities, giving this category a score of 0.75. On the other hand, even though the text contains references to more Harvard_University_Alumni (34), a total of 6533 entities in DBpedia are associated with this category, giving Harvard_University_Alumni a much lower score than American_Presidents.

Table **??** gives top categories considered most relevant to APUSH by this algorithm. As can be seen, of the hundreds of thousands of categories in DBpedia (which includes categories for rock stars, planets, etc.), the top scoring categories are indeed very apropos to US history.

We then score each entity by summing the scores for the categories that it is associated with. For example, the entity Abraham_Lincoln gets a contribution from each of the 29 categories that it is a part of. Table **??** gives the top-rated entities.

Next, we score each proper term by adding the scores of all the entities that it could refer to. So, since "Abraham Lincoln" could refer to the president or the movie with that name, each entity contributes a score. Table **??** gives the top-rated proper terms. Again, as can be seen, of the millions of entries in DBpedia, the ones chosen are indeed very highly apropos to the APUSH context.

Whigs in Congress
President Harrison
President Monroe
James Monroe
President Johnson
Thomas Jefferson
President Adams
Second Bank
Andrew Jackson
President Van Buren

Table 3: Highest scoring proper terms for APUSH

We then score each query by summing the scores of the proper terms in the query. Finally, we score each of the sites based on the scores associated with the queries for which they produced results. The score for the site is the average of the query scores. This gives us a ranking of sites by the likelihood of them being a good candidate for inclusion into the APUSH search engine.

3.3 Relevance feedback

The previous two algorithms address the fundamental problem of off-topic search results that we discuss in the introduction. As we demonstrate in the experimental section, they both make significant progress toward automated construction of the course-specific search engine for APUSH. However, they are not well-suited to distinguish between good sites and sites that are on-topic but either at the wrong academic level or inappropriate for academic purposes, such as answers.yahoo.com or ConfederateAmericanPride.com. To address these issues, we employ a classical method in information retrieval, relevance feedback.

Using relevance feedback to improve the performance of systems is an established technique [**?**]; as the search engine gets used, we can interpret clicks from the user as feedback. For example, if users repeatedly skip results from a certain site even when pages from that site are ranked higher, preferring results from certain other sites, they are expressing a judgement about the relevance of that site to the APUSH context.

However, evaluating relevance feedback algorithms requires large amounts of usage data, which is not available for a new search engine. Instead, we reuse the manually curated sites to evaluate the utility of relevance feedback in our context. Of the sites that had been manually curated, we randomly select 50 good and 50 bad sites. In practice, this list of good and bad sites would be obtained from usage logs.

In order to use relevance feedback to improve the text similarity algorithm, we aggregate the text of the good sites and bad sites into two large composite documents. Then, we score each site using TF-IDF weighted text similarity to generate a *GoodScore* and a *BadScore* based on the site's similarity to each document. We then form the *RelTextScore* as

$$RelTextScore = TextScore + GoodScore - BadScore \quad (2)$$

where *GoodScore* denotes the text similarity score between the site and the document of good sites, *BadScore* denotes the text similarity score between the site and the document of bad sites, and *TextScore* denotes the text similarity score

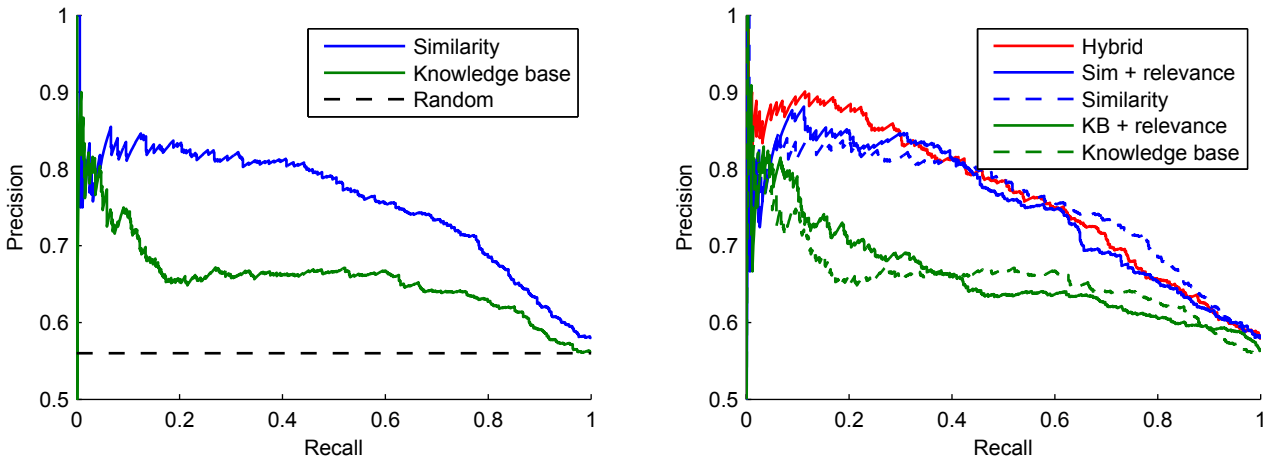

Figure 2: Comparison of proposed algorithms on classification of APUSH sites (left); augmented with relevance feedback and hybrid scoring (right). Precision represents the percentage of good sites when looking at the top n sites and recall represents the percentage of good sites out of the total number of good sites that are in the top n

between the site and the APUSH textbook. $RelTextScore$ denotes the score of a site based on it's textual similarity to the textbook and relevance feedback.

To augment the knowledge base approach, we compute a relevance feedback score for each category as follows. We take the categories associated with each query and propagate them to the sites associated with the query to get a set of categories associated with each site. The score associated with each category is

$$RelCategoryScore = CategoryScore + \#Good - \#Bad \quad (3)$$

where $\#Good$ is the number of good sites associated with the category, $\#Bad$ is the number of bad sites associated with the category, and $CategoryScore$ is the category based score of each site (described and calculated in section 3.2) . $RelCategoryScore$ represents the score of a category based on relevance feedback.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the ability of our proposed methods to identify high quality sites for a course-specific search engine. In practice, this would typically be done using a side-by-side methodology to evaluate the relative quality of search results similar to the procedure described in Section 2. However, given that the APUSH search engine constructed from a hand-labeled corpus is strongly preferred to Google (as shown in Table **??**), we instead focus on the ability of our methods to recover this hand-labeled corpus. For simplicity, we will evaluate our ranking of sites using a classification framework—of the 1757 APUSH-related sites, we consider what fraction of the 989 good sites are ranked above the 768 bad ones.

In Figure **??** (left) we see that both of our approaches significantly outperform the baseline of 56%, which comes from picking sites randomly. Although neither approaches provide perfect classification accuracy, for the purposes of constructing a course-specific search engine, it is clear that the course textbook provides a significant signal as to what is relevant for the APUSH educational context. Furthermore, the strongly positive results of the side-by-side in Table **??** suggest that even simplistic methods of incorporating this signal into search ranking may lead to significant gains in search quality.

In Figure **??** (right), we examine the effect of incorporating relevance feedback and hybrid scoring. We see that although relevance feedback improves precision at various points on the recall curves—for example, improving precision for the highest ranking sites from the knowledge base approach—it does not have as large an impact as might be expected. We believe the reasons for this are two-fold: 1) the simplistic nature of our use of the feedback information and 2) a lack of real feedback data. Ideally, we would improve on this result by using more sophisticated methods and a large amount of real search logs from a running search engine. Finally, we also investigate combining the textual similarity score with the knowledge base score by simply adding the two scores. As can be seen, there is a small improvement over just the similarity score and we believe that improving this hybrid scoring is another direction for future work.

5. ACKNOWLEDGMENTS

We would like to thank Professor Lam and Dr. Hangal from Stanford for guiding our research and for feedback on several iterations of the paper. We would also like to thank Mr. Tuomy, Mr. Johnson, Ms. Richard and Mr. Weisman for taking the time to evaluate the search engine.

6. REFERENCES

[1] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[2] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web (ldow2008). In *Proceeding of the 17th international conference on World Wide Web*, pages 1265–1266. ACM, 2008.

[3] W. Buntine, J. Lofstrom, J. Perkio, S. Perttu, V. Poroshin, T. Silander, H. Tirri, A. Tuominen, and V. Tuulos. A scalable topic-based open source search

engine. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 228–234. IEEE Computer Society, 2004.

[4] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of the 12th international conference on World Wide Web*, pages 700–709. ACM, 2003.

[5] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *Knowledge and Data Engineering, IEEE Transactions on*, 15e(4):784–796, 2003.

[6] C.-C. Hsu and F. Wu. Topic-specific crawling on the web with the measurements of the relevancy context graph. *Information Systems*, 31(4):232–246, 2006.

[7] P. Jacsó. Google scholar: the pros and the cons. *Online information review*, 29(2):208–214, 2005.

[8] X. Jiang and A.-H. Tan. Learning and inferencing in user ontology for personalized semantic web search. *Information Sciences*, 179(16):2794–2808, 2009.

[9] D. M. Kennedy, L. Cohen, and T. A. Bailey. *The American Pageant, Twelfth Edition*. Wadsworth Publishing, 2002.

[10] M. F. Porter et al. An algorithm for suffix stripping, 1980.

[11] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24:5, 1997.

[12] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[13] A. Singhal. Introducing the knowledge graph: things, not strings, 2012. http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html.

[14] Wikipedia. AP United States history, 2013. Available at: http://en.wikipedia.org/wiki/-AP_United_States_History.

APPENDIX

A. SIDE-BY-SIDE EVALUATION QUERIES

In Table **??** we list the queries used in our blind side-by-side evaluation between the APUSH search engine and Google.

Andrew Jackson
Battle of Saratoga
american strategies during world war 2
andrew johnson reconstruction
battle of vicksburg
bay colony
british involvement civil war
carpetbaggers
causes of the civil war
great awakening
industrial revolution
king cotton
marne
mclellan
republicans reconstruction
sectionalism and slavery
tea party
uss chesapeake
war of 1812
women's rights world war 2

Table 4: Side-by-side evaluation queries