Course-specific search engines: Semi-automated methods for identifying high quality topic-specific corpora

Neel Guha
neelguha@gmail.com

Matt Wytock
Google, USA
mwytock@google.com

ABSTRACT

Web search is an important research tool for many high school courses. However, generic keyword search engines have a number of problems that arise out of not understanding the context of search (the high school course), leading to results that are off-topic or inappropriate as reference material. In this paper, we introduce the concept of a course-specific search engine and build such a search engine for the Advanced Placement course on US History; the results of which are evaluated by subject matter experts (high school teachers) and judged to be overwhelmingly superior to existing search engines. We then develop two algorithms for identifying high quality documents: one based on textual similarity to an authoritative source and another using structured data from knowledge bases. Initial experimental results indicate that these algorithms can be used to automate the creation of course-specific search engines with minimal manual supervision.

1. INTRODUCTION

Over the last decade, the World Wide Web has become one of the primary sources of information for students. This is especially the case for high school subjects such as history, which often have projects that require the student to perform independent research. Unfortunately, students encounter difficulties with mainstream keyword-based search engines (such as Bing or Google); consider two examples of queries that might arise in the context of a course on US History: the query [boston tea party] brings up the home page for the Boston chapter for the political organization called the "Tea Party" and the query [benjamin franklin] brings up pages about Benjamin Franklin Plumbing. These queries also bring up pages that are targeted at elementary school students and user-generated content such as Yahoo Answers, which are not considered good reference material by most teachers.

Traditional methods in information retrieval rely on term-based scoring between the query and the document via the vector space model [12] or more sophisticated Ngram models to ensure on-topic results. However, for many reasonable queries in the academic research context, such methods are fundamentally limited. We group problems that arise into three categories:

Off-topic results. Often, many of the results are off-topic in the research context. For example, [benjamin franklin] brings up results about a plumbing service with that name and [gold rush] brings up pages related to Gold Country tourism. The problem is especially severe with queries involving names of places, which bring up results about restaurants, real estate offerings, and other local services. Since students are learning the topic by conducting exploratory searches, they cannot be expected to frame the best query which exacerbates the problem.

Inappropriate sources. Web search results often include a number of sources that do not meet the standard for research material in an academic course. Some egregious examples include user-generated content (such as forums and Yahoo answers), sites offering essays from other students, and biased sites (such as ConfederateAmericanPride.com). Unfortunately, these results are interspersed with those from reputable sites leaving the student to sift through result set. Unlike off-topic results which are mostly just a nuisance, these results often lead the student astray.

Wrong level. Even search results that are on-topic from reputable a site may be targeted at the wrong level. For example, [thomas jefferson] returns a page from the children's version of the Library of Congress website while more detailed queries often return papers from graduate level work. Typically web pages are not explicitly labeled with the level of the intended audience making it difficult to formulate an query that returns appropriate results.

In practice, the user must compensate for these search engine deficiencies by constructing more specific queries. Unfortunately, this both requires an intimate knowledge of the subject, which by definition the student does not yet have, and often eliminates potentially good results. The root of the problem 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, the general purpose search engine must have appropriate results for all users and 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 [14], 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-consuiming 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 algorithms that utilize this source: one using textual similarity and another using structured data from knowledge bases. 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 [5, 6] or to influence the order in which pages should be crawled [3]. In contrast, our work uses the text content of an authoritative source (the course textbook) 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 [7]. 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 [8], which explores using ontologies for characterizing the user and [4] which show how search results can be augmented with snippets from a knowledge base.

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 [9], which is available in PDF form. As described above, we use the Google CSE platform which allows us to specify the corpus for our a 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 go through the search results returned by Google and manually pick the good results. Clearly, this is not possible both because the queries are not known apriori and because of the magnitude of manual work that would be required. We therefore approximate this by computing likely queries from the textbook and then, by doing the curation at the site level as opposed to the page level. This curated set of sites will then form the corpus of our reference search engine.

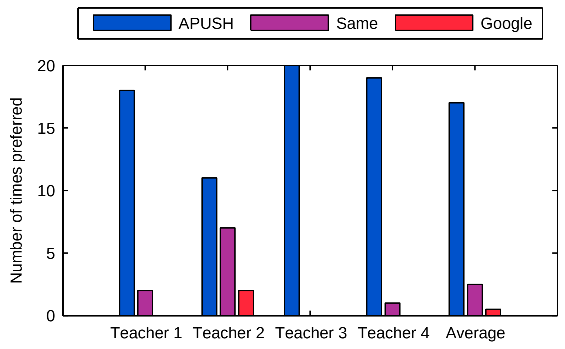

Figure 1: Side-by-side comparison of the APUSH search engine and Google

Most history related queries include one or more proper nouns (people, events, places, etc.). Thus, we use the digital version of the textbook [9] to extract all the 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. The 132,145 results for these queries come 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 to put them into two buckets: 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. So, if we were to randomly select sites, only 56% of the selected sites will be good.

We created a custom search engine using the good sites (available at `http://guha.com/apushcse.html`). In order to evaluate this search engine, we collected a set of 20 queries (included in Appendix A) and asked a set of 4 history teachers to compare results from Google versus results from the custom search in a blind side-by-side test setup. 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 1, the course-specific custom search engine is substantially preferred.

3. METHODS FOR AUTOMATION

Our long term goal is to be able to create a course-specific search engine from the course textbook with minimal manual supervision. In particular, instead of manually curating thousands of sites, 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 that are ranked higher.

In order to automate the process of identifying these sites, we consider different methods of using the authoritative text provided by the textbook in an offline process. For simplicity, we evaluate our ranking of sites using a classification framework—of the 1757 APUSH-related sites, we consider what portion of the 989 good sites are ranked above the 768 bad ones. The main intuition behind our algorithms is that by using the wealth of information provided by the textbook, we can drastically improve upon scoring that is based solely on a two or three word query. Thus, a reasonable baseline is to a consider a simple frequency-based approach in which we rank sites in descending order of the frequency by which they show up for the synthetic queries described in the previous section. However, as we discuss in the experimental results section, this simple frequency-based approach does not improve significantly over a random baseline.

Thus, in order to use the information provided in the textbook more effectively, we develop two new algorithms for topic-specific site scoring. We begin with an approach that computes text similarity via TF-IDF weighted cosine distance in order to prefer sites that are more similar to the APUSH textbook. Next, we consider an approaching using structured data from a knowledge base to identify APUSH-related categories, entities and proper nouns. We augment both approaches using relevance feedback which we approximate with our hand-labeled sites. As discussed in the experimental results section, we find that both algorithms improve significantly over the frequency scoring baseline and when combined in a hybrid approach, deliver results that approach the quality of the hand-labeled corpus employed in our reference search engine.

3.1 TF-IDF weighted text similarity

Classical information retrieval is based on the notion of similarity between two pieces of text, typically, the query and the web page. In our case, we compute the similarity between the pages on a site and the APUSH textbook. Sites that are more similar to the APUSH textbook should 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 [12]. 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)—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 [10]) 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

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 (see for example [13]). In the academic sphere, there has also been significant work on the semantic web [1] and linked data [2], 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 improve the 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, construct mappings from the proper nouns that we extract from the text to entities and from the entities to types of entities. To do this 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.

Next, 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 using 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

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 Buchanana
Zachary Taylor
William Henry Harrison
Russia

Table 2: DBPedia entities for APUSH

different categories associated with it: American_Presidents, Illinois_Lawyers, Assasinated_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, 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 1 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 2 gives some of the top-rated entities.

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

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 3 gives some of 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.

We then score each query based on 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 capable of identifying 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 [11]—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

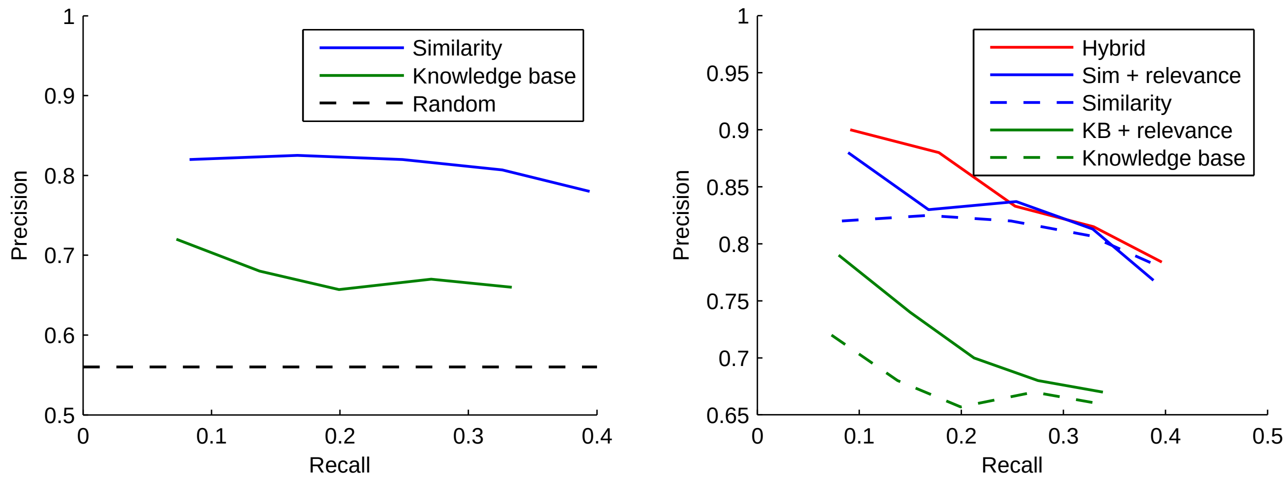

Figure 2: Comparison of basic algorithms on classification of APUSH sites (left) and augmented with relevance feedback and hybrid scoring (right)

and bad sites would be obtained from usage logs.

In order to use relevance feedback to improve the text similarity algorithm, we aggregate the good sites and bad sites into two large composite documents. Then, we score each site using the TF-IDF weight text similarity algorithm described in the previous section to generate a $GoodScore$ and a $BadScore$ for each document. We then form the $RelTextScore$ as

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

where $TextScore$ denotes the text similarity score between the document and the APUSH textbook.

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

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

where $\#Good$ is the number of good sites associated with the category and $\#Bad$ is the number of bad sites associated with the category. Then, as before, we propagate the category scores to the sites.

4. EXPERIMENTAL RESULTS

Clean this up

Of the 1453 sites we examined as part of the manual curation, 582 were good, 871 tended to give off-topic or inappropriate (i.e., bad) results. So, randomly picking sites for inclusion into the search engine will lead to 40.05% of the sites being good. This forms the baseline for evaluating algorithms for automating the curation.

Each of our algorithms produces a list of sites ranked according to their likelihood of producing good results for APUSH queries. We use the manually curated list of good sites to measure how well an algorithm performs. The ideal algorithm would rank each of the 989 good sites above the bad sites. Therefore, a good metric for evaluating an algorithm is the number of good sites included in the top 100, 200, 300, 400 and 500 sites picked by the algorithm.

We also created a baseline from simple frequency-based scoring, where sites are ranked based on their frequency. As visible in Figure XXX, this demonstrates that the most commonly occuring sites for any query are not necessarily good.

More details of experimental methodology here and discussion of results from previous sections and put here. Also include the description of the simple frequency-based scoring approach here as this is really a baseline not a proposed method. Include graph.

4.1 Relevance feedback and hybrid scoring

Include graphs and discussion of results here when you add in relevance feedback and hybrid scoring

As a final step, we investigate combining the statistical, similarity based score with the knowledge based score. We compute a combined score by simply adding the similarity score and the knowledge based score. Table 8 gives the results from this hybrid approach. As can be seen, there is a very small improvement over just the similarity score. We believe that there is room for improvement here and this is one of the directions of future work.

5. 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 4 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