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

Neel Guha
neelguha@gmail.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 them 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 performance measurements indicate that 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 in high school for subjects such as history, which often have projects that require the student to perform independent research.

Spliced together description of problems from two sections. Should be merged to flow with paragraph above, remove any redundancy, etc. Probably fine to launch directly into the expanded description of the three types of problems—easy to process this way.

Students typically use a search engine (such as Bing, Google or Yahoo!) to find pages relevant to their project. Unfortunately, students encounter certain difficulties with these keyword based search engines. Consider several 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". The query [benjamin franklin] brings up pages about Benjamin Franklin Plumbing. They also bring up pages that are targeted at elementary school students and pages from user generated sources such as Yahoo Answers (which are not considered good reference material by most teachers). Queries about places such as [gold country] are even more problematic, bringing up results related to tourism, classifieds, etc.

Generic search engines have a number of problems with searches that are done as part of research on an assignment or project for an academic course. There are three broad categories of problems, in decreasing order of severity.

Off-Topic Results. Often, many of the results are off topic. For example, [benjamin franklin] brings up results about a plumbing service with that name, [gold rush] brings up pages related to Gold country tourism, etc. The problem is especially severe with queries involving names of places, which bring up results about local services, real estate offerings, etc. Since students are still learning the topic and are often conducting exploratory searches and don't know exactly what they are looking for, they can't be expected to frame the best query.

Inappropriate sites. Teachers have an expectation of research material coming from reference sources. Web search results often include a number of sites that don't meet this bar. Some of the different types of sites that are not appropriate include user generated sites (such as forums and Yahoo answers), high school sites with essays from other high schoolers, biased sites (e.g., ConfederateAmericanPride.com). Unfortunately, these results are interspersed with results from much more reputable sites. The student is left to sort through all the results. Unlike off-topic results which are mostly just a nuisance, these kind of results can often lead the student astray.

Wrong level. Some of the pages returned are on topic and from reputable sites, but are targeted at the wrong level. For example, [thomas jefferson] returns a page from the children's version of the Library of Congress website. More detailed queries often return papers from graduate level work.

Next part should talk about existing solutions, query refinement as well as existing academic work on topic-specific search engines. The analysis and discussion of existing work should probably be expanded slightly. Make sure to contrast specifically prior work to this approach.

The typical solution to these problems is to construct 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.

Though there is substantial work in the area of topic specific

ranking in search, much of it is focussed around exploiting the link structure of the web to identify clusters of documents that are on the same topic. This structure can be exploited to create topic specific Page Rank [6] for influencing ranking or to influence the order in which pages should be crawled [7, 4]. More recent work [1] looks into topical analysis of query logs.

There has been very little work on the use of knowledge bases to influence the results shown in search. [9] explores using ontologies for characterizing the user. [5] and subsequent papers show how search results can be augmented with snippets from a knowledge base.

There has also been very little work on creating specialized search engines for educational purposes. The most closely related systems are systems like PubMed, Web of Science and Google Scholar, which are specialized corpura around scientific research. A description and comparison of these is at [8].

Should revise the next two paragraphs to be more of a summary of the next 3 sections so reader knows what to expect. Key points: course-specific search engine for APUSH which has good results. Main ideas for two algorithms and mention summary of the quantitative evaluation.

Our solution is to construct a specialized search engine for educational contexts. More specifically, we are interested in constructing a search engine for each high school course. In this paper, we use a popular high school course (AP US History) as our target and show how we can construct a web search engine, that is specific to this course, that captures the richness of the web while avoiding some of the problems associated with general purpose search engines. We use the Google Custom search engine (CSE) platform (`http://google.com/cse`) to run the search engine. With a CSE, we specify a corpus via a set of url patterns or sites and get back a search engine that searches only over this corpus. The CSE platform takes care of the cumbersome tasks of crawling the web, building an index and running a search engine. This has allowed us to not only build a search engine for AP US History (APUSH), but to also make it widely available to students taking the course.

The outline of the rest of the paper is as follows. We first analyze some of the common problems encountered during searches done in the context of high school courses such as APUSH. We then describe how we constructed and evaluated a reference search engine for APUSH by harvesting a set of sites from Google search and then manually curating it. While the search engine thus created is very useful, the manual curation of thousands of sites does make it expensive. This motivates the our algorithms for automating the construction of course-specific corpora.

2. REFERENCE SEARCH ENGINE

Our ultimate goal is to create a system, which, given a course textbook will produce a search engine that is appropriate for that course. In this work, we use [9?], which is available in PDF form, to create a search engine for the AP US History course, which was taken by over 400,000 students in 2011 [10?].

Building and running a search engine requires a substantial investment in engineering and infrastructure. Fortunately, Google offers the custom search engine platform (CSE), which allows us to create customized search engines at a small fraction of the cost of building a search engine from scratch. With a CSE, we specify a subset of the web as a set of URL patterns and get a search engine that searches only over pages that match one of these patterns. We rely on CSE for the crawling, indexing and serving search results. Our focus is on identifying the subset of the web that should be included in an APUSH specific search corpus.

In order to evaluate the relative performance of a search engine with a course specific corpus versus one that searched the whole web, and to create a baseline against with we could measure the automated identification of the course specific corpus, we first create a reference search engine.

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 we don't have all the queries 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). We then put this curated list of sites into a CSE to create our reference search engine for APUSH.

Most history related queries include one or more proper nouns (people, events, places, etc.). We used the digital version of the textbook [9?] to extract all the proper nouns, using simple syntactic cues such as capitalization and punctuation. We were able to identify 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 Good to include queries in an appendix 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. The results of this evaluation are shown in Table 1. As can be seen, the course specific custom search is substantially preferred.

Add bar chart here w/ 4+1 charts for each teacher

3. METHODS FOR AUTOMATION

Our long term goal is to be able to create a course-specific search engine from the course textbook. The reference search engine described above involved manual curation of thousands of sites, a process that is very time consuming and expensive. Our goal is to automatically 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 rank sites based on the likelihood of them returning good results for APUSH searches. A CSE can then be configured to include just the top N sites or better, to prefer sites that are ranked higher.

Links should be made here about topic-specific PageRank. Essentially this is a course-specific quality score.

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, i.e., the APUSH course. When an off-topic site (such as Benjamin Franklin Plumbing) comes up for a history query (such as [benjamin franklin]), the problem is that the query is inherently ambiguous and there is no way to communicate the context of the search (i.e., the APUSH course) to the search engine. Our overall approach is to use this larger context in an offline process to construct the right collection of sites.

We now describe a progression of algorithms that can be combined to improve on this baseline. We start with a classical information retrieval style analysis of the content of search results page and of the APUSH textbook. Sites that are more similar to the APUSH textbook are more likely to yield better search results. In the next step, we use a knowledge base of facts about the real world entities to improve the ranking obtained by just textual similarity. Finally, we present a relevance feedback based learning algorithm that improves the search engine based on user interaction with search results.

3.1 Authoritative documents

Instead of writing this section in terms of an explanation of TF-IDF and vector space models which are classic methods, I would rephrase this as an alternative way to compute a topic-specific quality score using an authoritative reference which is a new idea.

Classical information retrieval (IR), one of the primary mechanisms used by search engines such as Google, 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.

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 where each term in the corpus is a dimension. The coordinates of a document along the ith axis is given by by a product of the term's frequency in the document (TF) and the a measure of how significant that term is in the corpus. The later is usually measured by Inverse Document Frequency (IDF), which is the log of the inverse of the number of documents in the corpus that the term appears in. The similarity between two documents in this model is the cosine of the angle between the vectors between the two documents.

Aiand Bi are the tf-idf for each term i in the corpus, for the two documents. We compute the similarity of a site to the APUSH textbook as follows. We retrieved the content of 110,121 of the 132,145 pages returned by Google as results for the queries that we used to create the reference search engine (many pages could not be retrieved either because the sites blocked our robot or because the site was down). After stripping the pages of all the HTML markup and javascript, we stem the words on each page (using the Porter Stemmer [10]) and extract the terms from (the pages from) each site along with their frequency of occurrence. We also stem all the terms in the APUSH textbook. We then compute a similarity score between each site and the APUSH textbook using the cosine similarity metric.

While textual similarity based metrics are very powerful for identifying off-topic sites, they are not capable of identifying sites that are on-topic, but likely inappropriate for academic purposes (such as answers.yahoo.com or ConfederateAmericanPride.com). These sites are indeed on-topic, but pure similarity based metrics cannot detect the bias. For this, we need some form of human judgement. We next describe a mechanism for getting this by analyzing the search engine's usage.

3.2 Knowledge based approach

Most queries and web documents are about real world entities. This realization has lead many of the big search engines to build various kinds of knowledge bases to augment their search results Nice to cite blog post here. There has also been a lot of work in the area of the semantic web and linked data [2, 3] which aim to build a large distributed network of information about entities and the relations between them. In this section, we describe how this source of information can be used to improve the ranking based on simple textual similarity.

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 are pulled up for queries that contain references to these kinds of entities. Before we can do that, we need to map the proper nouns we extract from the text to entities and the entities to types of entities. Ideally, we would like to do this in a fashion that is not specific to history, but will work with little or no change to many other subjects. To do this, we need a broad knowledge base about a large

number of entities, along with information about the type of each entity. Wikipedia contains this kind of information about a large number of entities and DBPedia makes this information available as a structured knowledge base. Each "thing" in Wikipedia corresponds to an entity in DBPedia (www.dbpedia.org) 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.

There could be many different proper nouns corresponding to a single real world entity. For example, President Lincoln, Abraham Lincoln, Abe Lincoln, etc. 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 candidate entities is to use search itself. For every proper noun P, we issue the query [P site:wikipedia.org]. Each of the search urls has the form http://en.wikipedia.org/wiki/<entity-id>. The first three entities for each query are used as candidate entities for the corresponding proper noun. Each entity may 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 (e.g., American_Presidents, Illinois_Lawyers, Assasinated_HeadsOfState, etc.). This mapping is done using the RDF dumps from DBPedia.

The next step is to assign a score to each category. The score is a measure of the likelihood of a query containing a proper term that refers to an entity in that category bringing up a site 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. These categories should be assigned higher scores. An example of such a category would be American_Presidents. There will be a number of other categories that also appear, but only incidentally. These should get lower scores. Examples of such categories would be 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. The score for the ith category is given by

$$CategoryScore_i = \frac{\#Textbook_i}{\#DBpedia_i} \quad (1)$$

where $\#Textbook_i$ and $\#DBpedia_i$ count occurences of entities 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.

Put Table 4/5 in a single two-column table here in latex format

Table 4 gives some of the top rated categories for APUSH, i.e., the categories that are considered most relevant to APUSH. As can be seen, of the hundreds of thousands of categories in DBPedia (which includes categories for rock stars, planets, etc.), the ones that come up 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 5 gives some of the top rated entities.

Table 6 here

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 6 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.

We should tie this approach (and the topicality scoring approach) directly to the the 3 problems laid out in the introduction: off-topic, inappropriate, wrong level. In particular, its clear that both deal with off-topic results easily but not so clear how they work for inappropriate and wrong level.

3.3 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 appropriateness of that site for APUSH.

First, describe how we use the labeled data to generate fake relevance feedback. It would be best to use the equation environment to make it easy to parse.

Evaluating relevance feedback based algorithms requires large amounts of usage data, which we don't have. So, we instead reused some of the manually curated sites to evaluate the utility of relevance feedback in our context. Of the sites that had been manually curated, we randomly selected 50 good sites and 50 bad sites. In practice, this list of good and bad sites would be obtained from usage logs. We aggregate the good sites and bad sites, each, into a single composite document, so that we have 2 large documents, one corresponding to the good sites (GD) and another corresponding to the bad sites (BD). Then, for each site we are trying to score, we compute the similarity of the site to GD and BD. The net relevance feedback score is the Good Score minus the Bad Score.

Now, with the fake feedback data out of the way we can describe the two methods in a way that is agnostic as to

whether the data is fake agnostic as to whether its real or fake data.

The overall score for the site is the score for similarity to the APUSH textbook + the score for similarity to the good sites - score for similarity to the bad sites.

We can also incorporate relevance feedback to incrementally improve performance. As with relevance feedback for similarity scoring, we sample 50 good and bad sites from the manually curated list to simulate data from actual usage of the search engine. We compute a relevance feedback score for each category as follows. We take the categories associated with each query (from the previous step) 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 (Ng-Nb)where Ngis the number of good sites associated with the category and Nbis the number of bad sites associated with the category. Then, as before, we propagate the category scores to the sites.

4. EXPERIMENTAL RESULTS

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.

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] Judit Bar-Ilan, Zheng Zhu, and Mark Levene. Topic-specific analysis of search queries. In *Proceedings of the 2009 Workshop on Web Search Click Data*, pages 35–42. ACM, 2009.

[2] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

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

[4] Wray Buntine, Jaakko Lofstrom, Jukka Perkio, Sami Perttu, Vladimir Poroshin, Tomi Silander, Henry Tirri, Antti Tuominen, and Ville 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.

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

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

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

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

[9] Xing Jiang and Ah-Hwee Tan. Learning and inferencing in user ontology for personalized semantic web search. *Information Sciences*, 179(16):2794–2808, 2009.

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

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

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