Semiotic Topic-based Hybrid Learning Resource Recommendation

Lei Liu
Hewlett Packard Laboratories
Palo Alto, CA, USA 94304
lei.liu2@hp.com

ABSTRACT

Recommending learning resources to help readers understand any portion of the reading content where they have difficulty to understand is an useful and important task. Treating the whole unclear passage as the query and submit it to a search engine is unsuccessful since existing search engines were designed to accept small queries. In addition, as search engines usually transform the query and candidate resources into bags or vectors of words, the semantic topics underlying the content are totally overlooked. We believe that topics offer a better choice for truly understanding both the query and the candidate documents. In this paper, we propose a novel recommendation system for text content that facilitates the learning process by enabling search using as queries text passages of any length and retrieving a ranked list of resources (documents, videos, etc) that match the different topics covered within the selected passage. The recommended resources are ranked based on two criteria (a) how they match the different topics covered within the selected passage, and (b) the reading complexity level of the original text where the selected passage comes from. Our recommendation system has been built and being pilot from local universities and high schools, the user feedbacks from students who use our system in pilots for their courses suggest that our system is promising and effective. Beside this, we also provide a quantitative experimental evaluation, the results show that our proposed approach is promising and effective.

1. INTRODUCTION

Reading online content for educational, training, informational, entertainment or other purposes is a very popular activity. Readers of digital content enjoy several levels of interactivity. For example, they can select a passage and add an annotation. When readers have difficulty to understand any part of the text, they may want to find some learning resources to help them understand. One strategy is to treat the whole unclear passage as the query and submit it to a search engine. However, as existing search engines were designed to accept one or a few words as the query, the answer will be an error indicating that the query is too long to process. Figure 1 is an example when we try to search using a text passage with Google and Bing, in which Google returns an error message "query is too long to process", while Bing suggests user to "query is too long, please concise your query".

Figure 1: Errors generated with Google and Bing

The other way is to manually select a few words to form a query and search with any search engine, but this is inefficient and unreliable especially when readers do not understand the content. In addition, as search engines usually transform the query and candidate resources into bags or vectors of words, the semantic topics underlying the content are totally overlooked. Topics are a better choice for truly understanding both the query and the documents. In additional, existing information retrieval and recommendation systems rank resources based only on their relevance to the user request ignoring the suitability of the resources for the level of the particular reader. Measuring and taking into consideration the reading difficulty for each resource is a critical step to provide more accurate and useful recommendations.

In this paper, we propose a novel recommendation system for e-texts that facilitates the learning process by enabling search using as queries text passages of any length and retrieving a ranked list of resources (documents, videos, etc) that match the different topics covered within the selected passage. Our system makes the following contributions:

- Given any length of queries, our recommendation algorithms could recommend learning resources for each underlying query topic.
- Our methods measure relevance based on the topics underlying the query passage and the available resources. Topics are a better choice for truly understanding both the query and the documents as they capture not only their explicit relationships based on the common terms but also their implicit relationships.
- Our recommendation algorithms recommend resources with similar reading difficulty as the article where the queried passage originates from.
The remainder of this paper is organized as follows. Section 2 briefly reviews the related works on text retrieval and recommendation systems. Section 3 present the system architecture. The proposed recommendation methods are introduced in Section 4. Empirical results along with their discussions are presented in Section 6. Finally, conclusions and future work are discussed in Section 7.

2. RELATED WORK

Text retrieval systems match free-text queries to documents based on their content similarity. For long queries, a method is to split the article into contiguous blocks of text using a window [9]. Instead of computing the similarity of each document to the query, similarity is computed to each text block. The most similar blocks or their containing documents are returned as answers [1, 13, 20, 5]. These works still consider as input a short keyword query hence they differ from our approach. Also, although the article splitting idea could be utilized for the long passage query, their way to split the article into passages is based on window-based methods, which totally disregard the topics (concepts) underlying both the query and the candidate documents. In full text document retrieval, the target is to recommend the most relevant documents given an article as full text query [14]. This approach extracts the terms from all the documents and generates the document-term feature matrix using the Vector Space Model. The difference with our work is that we do not treat the query as a whole query text. Instead, we use a statistical model to extract the topics underlying the query text, and we find the most relevant documents for each topic.

3. SYSTEM ARCHITECTURE

Figure 2 shows the overall system architecture. The input is one or more passages from the e-text that a user is reading. The **Query processing** module is responsible for generating a set of topics, each of them indicating one latent topic underlying the selected passage. As each topic has an underlying word distribution, with a set of words with different weights to represent the topic, we select the top words for each topic to form a query. **Candidate resource generation** could be any online content or content in database. For example, it could be the search results with queried topics from any existing search engine. In our prototype, we use the Google API to get candidate content. Another example could be the content that we pre-crawled and stored in database from any specific learning related websites. The **Recommendation** module is used to select the best content resources from the candidates and provide the output results in a ranked list. The details of each component are discussed in section 4.

4. METHODOLOGY

4.1 Query Processing

The selected query passage is fed to the query processing module, which contains three components: preprocessing, topic generator, and topic compression. The preprocessing step involves noisy and stop words removal and stemming.

4.1.1 Topic Generator

Topic modeling algorithms are statistical methods that analyze the words occurring in the texts to discover the themes that run through them. The idea behind a topic model is that when a document is about a particular topic, some words should appear more frequently. Hence, documents are mixtures of topics, where each topic is a probability distribution over words.

Given a query passage, we use a topic model to discover the abstract topics underlying the query. After generating the topics, each topic will be represented by a set of words that frequently occur together. Examples of topic models are: the Probabilistic latent semantic indexing (PLSI) [8], and Latent Dirichlet allocation (LDA) [6, 11].

Figure 3 shows an illustrating example of query passage topic generation. Any topic model could be used here. In this paper, we use LDA [6]. In the figure, we use a document, from where the user can select different passages as queries, q1, q2, and so forth, colored differently. In this example, we generate four topics from these five queried passages, and each topic is represented by its top five words. Each cell \( T_{ij} \) in the query-topic matrix in the figure captures the probability score that query \( q_i \) covers topic \( j \). Intuitively, given that a selected passage is about one or more topics, one would expect particular words to appear in each topic more or less frequently. For instance, we observe that query \( q_5 \) focuses on a single topic while \( q_1 \) focuses on topic 1 and topic 2 with equal probability score 0.5.

4.1.2 Topic Compression

In LDA, the number of topics to be generated is given as input to the algorithm, and it depends on the document set where the model will be applied. In our problem setting, where users select a passage from a document to use as their query, we do not expect the passage to cover a very large number of topics. Hence, one solution is to fix the number of topics to a relatively small number, say 2 or 3. The rationale is that the more topics we extract from the passage, the more results we may need to show to the user in order to cover all the different topics. That may be counter-intuitive and less useful for the user. When a user is looking for resources related to a particular part of a document, it may be more meaningful to show results that focus on a few, most important, topics in the passage.

Still, as the number of topics associated with the queried passage is unknown, it is possible that multiple topics are generated but as-
associated with similar concepts. In order to remove such redundancy, we propose the idea of topic compression to reduce the topics into the most meaningful ones. One topic compression method is to consider the probabilities of the topics and prune topics that are not statistically important. In the example shown in Figure 3, topic 2 may be ignore for query 1 due to its low statistical significance. Furthermore, we can consider the word distribution of each topic, and remove duplicate topics if they are discussing similar concepts. To identify if two topics are about similar concepts, we can use a correlation (e.g., Pearson) or similarity (e.g., cosine) method to compare the topics. Since topics are probabilistic mixtures of words, another way is to use the Kullback-Leibler divergence [15, 19, 18] or Frobenius Norm [17, 4, 3], which is a non-symmetric measure of the difference between two probability distributions.

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Since topics are probabilistic mixtures of words, as a further topic compression method, we could consider the probabilities of the topics and prune topics that are not statistically important.

4.2 Candidate Resource Generation

Each underlying topic found from the previous step makes a query that is submitted to the candidate resource generation module. Specifically, we consider the top \(n\) words describing a query topic, and these words form a keyword query to be executed over the underlying search engine. In our prototype implementation, we use Google for this purpose. For each keyword query, the top \(K\) most relevant results are retrieved.

4.3 Recommendation

The responsibility of the recommendation module is to select the final resources to recommend to the user. Several factors need to be considered here in order to select the most representative and useful resources. First of all, the recommended resources should be related to the whole query passage. Furthermore, they should be close to the document where the query passage originates from in terms of reading difficulty. Finally, when more than one topic is discovered for the query passage, the recommended resources should cover all the topics achieving a good level of diversity. Two components tackle these issues: relevance ranking, and reading difficulty measurement and ranking.

4.3.1 Relevance Ranking

For each topic discovered from the query passage, a set of candidate resources is retrieved by forming a query comprising of the top words describing a topic. However, traditional search engines match documents to a query based on term similarity not topic similarity. Therefore, the retrieved documents need to be re-ranked based on their topic similarity to the query topic as well as with respect to the whole query passage. Hence a resource covering more topics in the query passage should rank higher.

For this purpose, each set of candidate resources along with the original query passage is treated as a content bucket. For each bucket, we generate a set of topics as the semantic features with the same topic generation method discussed earlier. Figure 4 provides an illustration of topic feature generator for one content bucket, where 4 topics are generated for the content bucket contains a query passage and three candidate resources. For each topic, we use a set of words to represent the topic concept.

We use the topic representations generated for the documents and the query in each content bucket to re-rank the documents of the bucket with respect to the query. Any similarity or distance function could be utilized here. We can then select the top \(k\) \((k < K)\) documents to show for each query topic discovered from the query passage. The final output can be biased to take into account the importance of each topic in the query passage.

4.3.2 Reading Difficulty Measurement and Ranking

The relevance ranking component ranks resources based on their relevancy to the query passage without considering if they are appropriate for the reader. Assessing the reading difficulty level of both the reading content where the query passage originates from and of the recommended resources is a key factor affecting the learning outcome. For example, assume a student selects a passage from an eighth grade physics book whereas a university student selects a query passage from a college-level physics book, and both queries may cover the same physics topic. Existing recommendation systems would provide the same relevant resources to both students. However, college-level resources may not be appropriate for eighth grade students whereas college students will find recommendations of eighth grade material not helpful.

Therefore, we perform a reading difficulty analysis on the article containing the query passage and the candidate resources. Readability measurements have been studied for long and can be broadly categorized into two groups: syntactical (e.g., Gunning Fog Index, Flesch Reading Ease, Flesch-Kincaid Grade Level Test, Automated Readability Index and Coleman Liau Index), and familiarity-based (e.g., New Dale-Chall Formula, Popularity-based Familiarity, Topic-based Familiarity and Genre-based Familiarity). No single method performs well on every content because they depend on the content type, subject, etc. All the existing readability metrics are supported in our system, and we use them accordingly. For instance, for the high school pilot, we are using New Dale-Chall (NDC) Score [2], which is designed for predicting the reading difficulty as the desired grade level. In our current system developed for the university pilot, as the courses are all designed for graduate level students, predicting the grade level is meaningless hence NDC is not appropriate. We use Flesch Reading Ease (FRE) [10], which estimates the reading difficulty based on the percentage of syllables and the average sentence size.

To recommend resources with the similar reading difficulty as the article containing the query passage, we create the readability similarity score: \(1/||\text{Rel}_c - \text{Rel}_a||\), where \(\text{Rel}_c\) and \(\text{Rel}_a\) are the readability scores of the candidate resource and the article containing the query passage, respectively.

To create the final list of recommendations taking into account both the relevance and readability similarity scores, we compute a recommendation score for each resource by combining the two scores using the following formula:

\[\alpha \text{Rel}_c + (1 - \alpha) 1/||\text{Rel}_c - \text{Rel}_a||\]

where \(\text{Rel}_c\) is the relevance score of each candidate resource to the query passage. Another

---

1we set \(K = 40\) in current system

2Cosine similarity is used in current system
method is to first select the top resources based on relevance and re-rank them based on their readability similarity score.

Given that we have retrieved $K$ results for each topic in the query, the responsibility of the recommendation module is to select the final resources to recommend to the user.

Several factors need to be considered here in order to select the best, more representative resources. A key question is how the retrieved resources relate to the whole query passage. Another question is how many resources to finally select in order to achieve a desired level of diversity and coverage by selecting resources that adequately represent the topics of the query when more than one topic is discovered. To tackle these issues, we present two components: topic feature generator, and relevance discovery and ranking.

The recommendation system proposed in this paper has been implemented and integrated into an online learning platform \([7, 12]\). We use Mallet v2.0.6\(^3\) to generate the topics, and in particular the parallel threaded implementation of LDA.

Our recommendation engine can recommend resources coming from different sources. One option is that the professor provides resources that supplement the core material of the book. In this case, these resources are indexed by our system in its local repository. Other options include using external general-purpose sites, such as Wikipedia or YouTube, or educational sources, such as CK-12 (www.ck12.org). In the former case, our system uses the Google API (ajax.googleapis) to fetch all the URLs from the google search results. For each keyword query generated for each topic in a query passage, the top $K = 40$ most relevant results are retrieved. From the list of URLs returned, we extract the Wikipedia content from HTML pages using BoilerPipe v1.2.0. We wrote a pattern matching code to extract the title, description, URL and publish date for YouTube videos. When recommendations come from educational sources, proprietary APIs may be used.

5. SYSTEM STUDY

The registered users can login our system. Professors can import the content and creates a book for their course, our system can automatically discover the illustration images with any selected content for book creation purpose. Figure 5 shows the interface of the system.

Once the books created, students can read the books that assigned to the courses which they enrolled. Figure 6 shows the students’ interface of our recommendation system. We demonstrate our system with a book that has been created by one of the professors in our pilots. Figure 6 illustrates how a user can interact with the recommendation system. A user can select any passage where she has difficulty understanding (an example passage is shown within a red dash rectangle). By right clicking, a menu appears that shows what recommendation options are offered for the particular book (depending on the professor choices as we described earlier). For example, in Figure 6, there are three options: “Learn from Course Material”, “Learn from Wikipedia” and “Learn from YouTube” for the user to select where the recommended learning resources will come from. Let us assume that the user chooses “Learn from YouTube” (shown in a blue rectangle). Then, example recommended learning resources are shown on the left side of the figure (green dash rectangle).

Depending on the type of recommended resource, different information is shown in the left pane. For example, for Wikipedia articles, the title, URL, and a snippet of the Wikipedia page are shown. The user can click the link to see the original page. For videos, the title, URL, and description of the video are shown. Additionally, the video is embedded in the results so that the user can watch it in place.

Since our system can recommend resources from different sources, each recommended resource is tagged using a different color. For example, we use a red color bar to tag video, blue for Wikipedia content, and gray for course material. Each passage where the student looked for recommended resources also gets tagged in a similar way. The color bar in front of the passage denotes the type of resources that have been recommended for this part of the content, like the bars in the green dash circles in Figure 7.

The system keeps a recommendation history, that is the last $X$ searches performed by a student ($X = 3$ is the default value). In this way, students can quickly jump to a page they sought recommendations by simply clicking the logs of the recommendation history (in the red dash rectangle in Figure 7). When a user mouses over one of the logs, a snapshot of the passage query shows up. The user can click on the passage to go to its actual location in the book and see the recommended resources. Finally, the students can save recommended resources in their personal notes.

6. EVALUATION

6.1 Experiment Setup

To show the effectiveness of our method, we consider 4 different ways to extract the key terms from the selected passage (summarized in Table 1) and 2 ways to discover the relevant resources (Table 2), consequently, we have implemented 8 scheme combinations from Tables 1 and 2.
The words are extracted with the largest \( tf \times idf \) value from the selected passage in S1, nouns and noun phrases are identified using off the shelf POS tagger. Both words and nouns are weighted by Frequency-based weighting \( (tf \times idf) \). The noun phrases are weighted by phrase weighting. The details of the weighting methods are provided as follow:

- **Frequency-based weighting**: For any term \( t_i \), the frequency based weighting \( f_s(t_i) \) is computed by using the \( tf \times idf \) weighting scheme widely used.
- **Phrase Weighting**: Let \( s \) be a phrase and \( t_i \in s \) be a term contained in \( s \). Then:

\[
    f_s(s) = f_s(t_i) \frac{\sum_{t_j \in s} f_i}{|s|}
\]

The phrase weighting \( f_s(s) \) considers two factors: \( f_s(t_i) \), the phrase TF-IDF score by treating each phrase as a single term, and the average frequency of all terms contained in the phrase \( \sum_{t_j \in s} \frac{f_i}{|s|} \), where \( |s| \) is the length of the phrase, i.e., the number of terms. The first factor considers the importance of the phrase as a whole unit \( f_s(t_i) \), the second factor is the relevance to the document reflected by the average frequency of its contained terms.

To evaluate the performance of different query extraction schemes, we conducted a user study with randomly selected passages from Wikipedia page as query. For each passage query, we generate the underlying topic with a set of key terms, then Google API is used to fetch top 40 results from Wikipedia and 40 results YouTube (although we could get any type content, we limit to these two sources in this paper) with such key terms. The title and body content are extracted from each Wikipedia page. We extract title and video description for each video page. With different relevance discovery methods (T1~T2), we select the top 20 results to show. The results are manually judged to be appropriate (relevant & best reading complexity) or not-appropriate.

We adopt **Precision** as our evaluation metric, which is the ratio of the relevant recommended results. In this paper, Stanford POS Tagger \(^4\) is used to extract nouns and noun phrases. The noun phrases are extracted by regular expression \((\text{Adjective} \mid \text{Noun}) \ast \text{Noun} \ast \text{Preposition})\).

### 6.2 Experimental Results

We repeat the experiment with ten different selected passages. For each passage, the precisions for all 8 combinations (S1~S4 with T1~T2) are calculated. Figure 8 plots the average precision for Wikipedia and YouTube. From the results, we have the following observations:

1. T1 is better than T2 with all key term extraction Schemes (S1~S4) in both Wikipedia and YouTube Results, which justify our argument that topics underlying the content offers a better way to truly understand both passage query and candidate resources (documents or videos).

2. Nouns and Noun phrases achieve similar performance, and they are better than words selected without POS tagger.

3. Our method (S4 with T1) achieves the best performance (precision=0.9 in Wikipedia and 0.65 in YouTube).

4. The overall accuracy of the YouTube results (0.5 ~ 0.7) suggests that only relying on the title and short video description is not sufficient to understand the video.

### 6.2.1 Case Study

To evaluate the ability of our method to deal with queries with more than one topic, we test a passage query with 80% content discussing about library and information science, the rest 20% focus on computer science topic, all the contents are collected from Wikipedia “Library and information science” and “computer science” \(^5\) page. We initially set the number of topics as three and find out there are two topics have a very high Pearson score, which indicates duplicate topics are detected. After removing the duplicate topic, we have two topics generated and provide five words to represent the meaning of each topic. We have topic 1 with word set “scientific, computation, computer, university, theory ”, topic 2 with word set “information, science, library, computer, field”. Then each set of words is put as a query and get the content from Google, rank them with the topic based similarity. To justify the effectiveness our method, we also select five words with largest \( tf \times idf \) values of the selected passage without considering the underlying topics. All the results are provided in Figure 9.

The first column of Figure 9 provides the top 5 recommendation results for topic 1, which mainly focus on computer science, while second column provides a list of library and information science related contents. The third column contains the results without

---

\(^4\)http://nlp.stanford.edu/software/pos-tagger.shtml

\(^5\)http://en.wikipedia.org/wiki/Library_and_information_science

http://en.wikipedia.org/wiki/Computer_science
<table>
<thead>
<tr>
<th>Topic 1 (scientific/computing/computer/uni/teachers)</th>
<th>Topic 2 (information/comm/lib/computer/field)</th>
<th>TF-IDf (information/science/lib/computer/field abbreviated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory of Computing - Computer Science...</td>
<td>Information science - Wikipedia, the free...</td>
<td>Library and Information science - Wikipedia, the free...</td>
</tr>
<tr>
<td>Computer science - Wikipedia, the ...</td>
<td>Library and information science - Wikipedia,</td>
<td>The phrase “library and Information science” is associated</td>
</tr>
<tr>
<td>Math/CS - Research Areas - Emory University</td>
<td>Library and information science - Wikipedia,</td>
<td>with schools of library and information science...</td>
</tr>
<tr>
<td>Numerical &amp; Scientific Computing - University...</td>
<td>Library science - Wikipedia, the free encyclopedia</td>
<td>Library school is an institution of higher learning...</td>
</tr>
<tr>
<td>Scientific Computing-School of Computer Science</td>
<td>Library and Information Science</td>
<td>Library-related Acronym</td>
</tr>
</tbody>
</table>

Considering the latent topics, which query term set is “information, science, library, school, abbreviated”, the recommended results are all related with topic “library and information science” and “computer science” related topic is totally disregarded.

From results in Figure 9, we can conclude that latent topics detection is necessary, especially when more than one topic exist in passage query, in which we should recommend the resources for each detected topic instead of recommending with respect to the whole passage.

7. CONCLUSION

In this paper, we proposed a novel semantic topic-based hybrid learning resource recommendation system, which can accept queries of any length. Our evaluation results suggest that key terms extracted by topic model combined with topic features based relevance measurement beats all other existing methods. Besides this, as more than one topic may exist in selected passage, our method is designed to recommend the learning resources to each underlying topic instead of to the whole passage, the case study in experiment section has justified the effectiveness of our method when we have passage with multiple topics.

8. REFERENCES


