LearningAssistant: A Novel Learning Resource Recommendation System

Lei Liu 1, Georgia Koutrika 2, Shanchan Wu 3

HP Labs, Palo Alto, USA

1 lei.liu2@hp.com 2 koutrika@hp.com 3 shanchan.wu@hp.com

Abstract—Reading online content for educational, learning, training or recreational purposes has become a very popular activity. While reading, people may have difficulty understanding a passage or wish to learn more about the topics covered by it, hence they may naturally seek additional or supplementary resources for the particular passage. These resources should be close to the passage both in terms of the subject matter and the reading level. However, using a search engine to find such resources interrupts the reading flow. It is also an inefficient, trial-and-error process because existing web search and recommendation systems do not support large queries, they do not understand semantic topics, and they do not take into account the reading level of the original document a person is reading. In this demo, we present LearningAssistant, a novel system that enables online reading material to be smoothly enriched with additional resources that can supplement or explain any passage from the original material for a reader on demand. The system facilitates the learning process by recommending learning resources (documents, videos, etc) for selected text passages of any length. 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 level of the original text where the selected passage comes from. User feedback from students who use our system in two real pilots, one with a high school and one with a university, for their courses suggest that our system is promising and effective.

I. INTRODUCTION

With the advent of portable devices such as tablets and e-readers, reading online content for educational, learning, training or recreational purposes has become a very popular activity. Readers of digital content enjoy several levels of interactivity. For example, they can add annotations, zoom-in on a picture, or play a video embedded in the content. More importantly, they may seek to read additional or supplementary online content related to a specific part of the e-text they have difficulty to understand or they wish to learn more about. However, this type of learning activity is not supported well.

One option is that the person treats the entire unclear passage as a query and submits it to a search engine. However, while existing search engines can answer queries with few words, for a long query, the answer will be an error message indicating that the query cannot be processed. Another option is that the reader manually selects a few words to form a query that can be answered by a search engine. This is inefficient and unreliable especially when readers do not understand the content. Furthermore, search engines typically transform the query and candidate resources into bags or vectors of words, hence overlooking the semantic topics underlying the content. In addition, 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.

To address these challenges, we created LearningAssistant, a novel system that facilitates the learning process by enabling search with text passages of any length and by recommending a ranked list of resources (documents, videos, etc) that match the different topics covered within the selected passage as well as the reading difficulty of the original text. In summary, the system makes the following novel contributions:

- Our recommendation algorithms allow recommending resources for queries of any length.
- Instead of measuring the relevance using typical IR models, such as bag of words, 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.

LearningAssistant is part of a larger system, the HP METIS platform1 that is an online learning environment. But it can also stand as an independent tool for resource recommendations for long queries. This paper starts with an overview of the LearningAssistant architecture, then describes the features to be demonstrated, and then provides an outline of our planned presentation to ICDE attendees.

II. DEMONSTRABLE FEATURES OF LEARNINGASSISTANT

A. System Architecture

Figure 1 shows the overall system architecture. The input is a passage from the e-text that a person is reading. The Query processing module is responsible for generating the set of topics covered by the selected passage. Each topic is described as a word distribution. We select the top words for each topic

to form a query. For each such query, the candidate resource generation returns candidate resources. These resources can be the search results from an existing search engine. They can also be resources (e.g., articles or web pages) stored in a database. The Recommendation module selects the best content resources from the candidate ones.

B. Query Processing

When a user selects a query passage from a text, 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.

1) Topic Generator: Topic modeling algorithms are statistical methods that analyze the words 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 often. Documents are mixtures of topics, where each topic is a probability distribution over words.

Given a query passage, we use the Latent Dirichlet allocation (LDA)[1] to discover its underlying topics. Figure 2 shows an illustrative example for query passage topic generation. In the figure, we depict two example query passages in different colors originating from the Wikipedia page on ‘data mining’ as accessed on September 2014. These passages cover three topics. We represent each topic by its top five words (shown in the lower right part of the figure). The query-topic matrix shown in the upper right corner of the figure captures how the query passages cover these topics. In particular, each cell stores the probability that query $i$ covers topic $j$. For instance, query 1 (in blue color) focuses on topic 1 with probability score 0.8, while query 2 is about topic 2 with probability 0.75.

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 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 2, topic 2 may be ignored 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[2], [4] or Frobenius Norm[3], [5], [6], which is a non-symmetric measure of the difference between two probability distributions.

C. Candidate Resource Generation

Each 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 topic, and these words form a keyword query to be executed over the underlying search engine. For each keyword query, the top $K$ most relevant results are retrieved.

D. 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.

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

<table>
<thead>
<tr>
<th>Query 1</th>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query 2</th>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15</td>
<td>0.75</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>
features with the same topic generation method discussed earlier. 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 use the cosine similarity. 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.

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 [7], which is designed for predicting the reading difficulty as the desired grade level. 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) [8], 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: \( 1/\|\text{Rea}_c - \text{Rea}_q\| \), where \( \text{Rea}_c \) and \( \text{Rea}_q \) 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:

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

where \( \text{Rel}_c \) is the relevance score of each candidate resource to the query passage. Another method is to first select the top resources based on relevance and re-rank them based on their readability similarity score.

E. Implementation Details

We have implemented our prototype system using Java and deployed it in the HP METIS platform, an online learning environment developed at HP Labs. We use Mallet v2.0.6\(^2\) to generate the topics, and in particular the parallel threaded implementation of LDA.

LearningAssistant 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 LearningAssistant 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, LearningAssistant 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.

III. INTERACTION WITH LEARNINGASSISTANT

Students and professors can register and login into HP METIS platform. The professor imports the content (articles, papers, etc) and creates an online book for their course. In order to enable LearningAssistant, they need to determine which resources can be recommended by the system.

Students can read the books assigned to them as shown in Figure 3. Figure 4 shows the 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 4 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 4, 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).

\(^2\)http://mallet.cs.umass.edu/download.php
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 learning 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 5.

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 5). 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.

IV. DEMONSTRATION

Our presentation will demonstrate LearningAssistant features using books from two pilots, one with a high school and one with a university. The latter is designed for graduate students in library and information science.

Our demonstration script will start with example passages from the university online book and recommended resources from Wikipedia and YouTube. We will demonstrate various interaction features that enable enriching online reading material with additional resources on demand related to passages from the original material. We will also give a taste of how students have used the learning resource recommendations.

Then, we will show how our recommendations are more accurate and useful compared to existing approaches. We have implemented three additional ways to extract the queries from the selected passage (summarized in Table I). $S1$ is the classical vector space model for web search, where vectors comprise terms. For $S2$ and $S3$, we use the Stanford POS tagger to extract the nouns and noun phrases. $S2$ is a variation of $S1$ where we keep only the nouns. For $S3$, we weigh the phrases using the following formula: $s_w(s) = f_w(s) \sum_{i \in s} f_i$, where $t_i \in s$ be a term contained in phrase $s$, $f_w(s)$ is the $tf \times idf$ for phrase $s$ by treating each phrase as a single term, $||s||$ is the length of phrase $s$ in number of terms. We will show and compare recommendations generated with our methods to these approaches. Furthermore, we will show recommendations when more than one topic exists in a selected passage. We will also demonstrate recommendations that are not only relevant but also match the reading difficulty of the high-school material or the university-level material per case.

Finally, for off-script presentation and discussion, we will provide interactivity, where the participants can interact and experiment with LearningAssistant over query passages, sources, and ranking schemes of their choice.

REFERENCES