Learning Assistant: A Novel Learning Resource Recommendation System

Lei Liu (HP Labs, USA)
With Georgia Koutrika and Shanchan Wu

Introduction

• Background:
  – While reading, people may have difficulty understanding a passage or wish to learn more about the topics covered by it.

• Limitation of existing approach:
  – Using a search engine to find resources interrupts the reading flow
  – Inefficient, trial-and-error process, as existing search engines or recommendation systems:
    • do not support large queries
    • do not understand semantic topics
    • do not consider the reading level of the original document a person is reading

• Our approach:
  – We present 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.
  – Our system ranks recommended resources based on two criteria:
    • How they match the different topics covered within the selected passage
    • The reading level of the original text where the selected passage comes from.

• Contributions
  – Our recommendation algorithms allow recommending resources for queries of any length.
  – Instead of measuring the relevance using typical IR models, our method measures relevance based on the topics underlying the query passage and the available resources.
  – Our recommendation algorithms recommend resources with similar reading difficulty as the article where the queried passage originates from.

System Architecture

Query Processing

• The selected passage is fed to the query processing module, which contains three components: preprocessing, topic generator, and topic compression.
  – Preprocessing:
    • Removing stop words, stemming, etc.
  – Topic Generator:
    • We utilize topic model to discover the abstract topics underlying the query
    • Latent Dirichlet Allocation is used in this paper

  \[ p(\beta_{1:n}, \theta_{1:n}, z_{1:n}, w_{1:n}) = \prod_{k=1}^{K} \prod_{i=1}^{M} \prod_{n=1}^{N} p(\theta_{k:n}) \prod_{n=1}^{N} p(z_{n} | \theta_{k:n}) p(w_{n} | \beta_{k:n}, z_{n}) \]

  \[ \text{We extract a small number of words (five in this paper), with highest probability score in } \beta \] to represent the concept of each detected topic.

  • Example:

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>3</td>
</tr>
<tr>
<td>is</td>
<td>2</td>
</tr>
<tr>
<td>able</td>
<td>1</td>
</tr>
</tbody>
</table>

Candidate Resource Generation

• Each topic found from the previous step makes a query that is submitted to the candidate resource generation module.
• 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.

Recommendation

• We consider several factors to select the most representative and useful resources.
  – The recommended resources should be related to the whole query passage.
  – They should close to the document where the query passage originates from in terms of reading difficulty.
  – 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.

To tackle these issues, we consider two components:
  – Relevance Ranking based on topic similarity
  – Reading Difficulty Measurement and Ranking
    – All existing readability metrics are supported in our system, and we use them accordingly.
    – The similarity in terms of readability is defined as:

  \[ \frac{\alpha \cdot \text{Relevance} + (1 - \alpha) \cdot \text{Readability} }{\text{Max(Relevance, Readability)}} \]

  where \( \text{Relevance} \) and \( \text{Readability} \) are the readability scores of the candidate resource and the article containing the query passage, respectively.

  \[ \text{Relevance} = \sum_{i} \frac{R_{i} \cdot (1 - \alpha)}{R_{i} + (1 - \alpha) \cdot \text{Readability}} \]

  \[ \text{Readability} = \sum_{i} \frac{R_{i} \cdot (1 - \alpha)}{R_{i} + (1 - \alpha) \cdot \text{Readability}} \]

Implementation Details

• We implemented our system using Java and deployed it in HP METIS, an online learning environment developed at HP Labs.
• We use Mallet V2.0.6 to generate the topics, and in particular the parallel threaded implementation of LDA.
• Learning Assistant can recommend resources coming from different sources.
  – One option is professor provide the resources and indexed in its local repository.
  – Other options include using external general purpose sites, such as CK-12 (www.ck12.org), Wikipedia or YouTube.
  – We use Google API (ajax.googleapis) to fetch all the URLs from the google search results.
  – From the list of URLs returned, we extract the content from HTML pages using BoilerPiple v1.2.0.
  – We wrote a pattern matching code to extract the title, description, URL and publish date for YouTube videos.