CSC 510 Software Engineering: Project 1

Group-P

Monica Metro
North Carolina State University
3021 F Dorner Circle
Raleigh, NC
mgmetro@ncsu.edu

Zachery DeLong
North Carolina State University
2305 Horizon Hike Ct
Raleigh, NC
zpdelong@ncsu.edu

Zhangqi Zha
North Carolina State University
1800 Vienna Wood Dr
Raleigh, NC
zzha@ncsu.edu

Bikram Singh
North Carolina State University
Page Hall, Raleigh, NC
bsingh8@ncsu.edu

ABSTRACT

Choosing a textbook for a class is something that many professors do several times a year, but the vast number of available textbooks for any given subject makes choosing one that both meets the needs of the class and meets the budget constraints of today's students is a challenge. In this paper, we propose two methods for automatically learning topics from books, a set of evaluations for those methods, and a system that will use those methods in a web app that attempts to suggest textbooks to a professor that balance hitting all topics the course requires while still being economical for students.

1. INTRODUCTION

1.1 Background

One of the most basic tasks that a professor must do is identify what textbook to use for a given class. While some fields have textbooks that have become popular and are clearly the best in their subject matter, many fields (especially emerging ones) have no such exemplar and reading through all possible candidates would be too time consuming to be feasible. It would be a complex enough problem if there were not already a massive number of textbooks on sources such as Amazon, but self aware professors have attempted to use textbooks that are less expensive with hopes of allowing lower income students to attend more easily. This admirable intention serves to increase the already considerable time needed to find appropriate books, and there is no obvious heuristic to apply to searches to limit the field.

In the interests of making students lives less expensive, we propose exploring a system which can identify topics in books automatically, and which, given a set of requirements from a user (usually a professor), can suggest options of varying expense and completeness for a given set of topics. To do this, we propose exploring word clusters generated by Doc2Vec, a family of common natural language processing (NLP) algorithms that attempt to cluster similar words, and topics generated by latent dirichlet allocation (LDA), another common NLP algorithm that directly attempts to identify topics in text, to automatically infer the content of a set of textbooks.

We then intend to build a web app that will allow a user to specify a set of topics and which will use the mentioned algorithms to make suggestions while minimizing the cost of said textbooks. The actual web app will be a fairly simple single-page web app developed in Angular4 and using Bootstrap to hasten development of a modern UI.

Initially, the team expects LDA to find more appropriate topics than Doc2Vec simply because it is better purpose built. That said, Doc2Vec uses a sophisticated neural-network, which may behave better on shorter passages such as a table of contents or an index, allowing us to parse less of the book while still getting useful topics.

1.2 Deviations From Initial Plan

1.3 Planning

The Spiral Model.

Prototypes.

2. LITERATURE REVIEW

We propose two methods for analyzing and storing topics from books automatically: LDA and Doc2vec.

2.1 LDA

In this section, we describe the Latent Dirichlet allocation(LDA) method for storing topics against books.

LDA uses a Bayesian clustering approach to find topics relevant to the book. [?] The input to the method is a book

PLDI '13 June 16–19, 2013, Seattle, WA, USA

ACM ISBN 123-4567-24-567/08/06…$15.00

DOI: 10.475/123_4

(or later, a series of books) which consists of collection of words.

This method assumes that each book (or books) consists of topics and that each topic has several key words associated with it. The number of topics associated with a book can be variable, and can be changed for better optimization. In this association, the words comprising the text are thrown into a "bag-of-words" model such that grammer and word order are disregarded, but multiplicity is measurable. The calculated topics found from this bag-of-words are "latent" variables, that is they are not directly measurable but indirectly observed.

The performance of the method depends on the initial assumptions made. For example, some have assumed that books are random mixes of topics [?]. The number of topics can initally be chosen as constant for a given book, or as a function of a chosen Poisson distribution. [?] Topic distribution in the book is assumed to be a sparse Dirichlet function.

A sparse Dirichlet function implies that we assume that a topic will be discussed only in a small breadth of pages in the book, and also that words relating to this topic will predominantly figure in these pages.

The algorithm looks to identify unique words to identify as topic names. Words like "the" and "a" are common and will occur with equal probability throughout the text. However, in say one chapter of the book some words may occur much more frequently in that particular chapter than the rest of the book, which means that the probability of these words in that chapter is more, which then the algorithm can use to choose a topic name and also a list of words related to that topic.

2.2 Doc2Vec

Doc2Vec is a model that extends an existing model, Word2Vec. Doc2vec mirrors Word2Vec in most ways except that it adds context information. [?]

Doc2Vec is an implementation of the Paragraph Vector (PV) introduced by Mikolov and Le in 2014 - an unsupervised algorithm that generates vector representations of text. [?] The basic difference between LDA and Doc2Vec is that LDA tries to form a structure of related words (a topic name with the list of associated words) out of books mainly on the basis of frequency of words in the document. While a simple yet powerful approach, it has a few drawbacks, such as semantics. For example, the words "powerful, strong and Paris" are equally distant when considered by a bag of words model, however "powerful" and "strong" are semantically close. [?]

Paragraph Vector involves predicting words in the paragraph. It is "unsupervised", which means that it predicts words in a paragraph and then uses these predictions to attempt to form structures of related words without any knowing of accuracy of the structure(s). The PV utilizes fixed length feature vectors learned from text sources of variable length. Mikolov offers two models of PV that are based on the implementation of Word2Vec: Distributed Memory Model (PV-DM) and Distributed Bag of Words (PV-DBOW).

PV-DM is similar to the Continuous Bag of Words (CBOW) model in Word2Vec such that is predicts the next word based on the given context. Paragraph vectors are concatenated with word vectors from that paragraph. Each word vector is representing a word from the given context and a new vector is generated by concatenating all of those word vectors together to predict other words. These word vectors are able to remember context and semantics. The paragraph vector allows the algorithm to remember the topic or 'label' of the paragraph, whether it is a sentence or a long document.

PV-DBOW follows the skip-gram model of Word2Vec. In this model, instead of concatenating the paragraph vector with the word vectors formed in close proximity in order to predict the next word, random text from the paragraph is selected and a random word is selected from that text. Because this model does not save the word vectors and therefore does not retain as much information about the context and semantics of the text, it requires less storage. Mikolov and Le recommend using a combination of PV-DM and PV-DBOW for consistently accurate results. [?]

2.3 Related Work

Doc2Vec was recently tested as the algorithm behind a recommender system with the goal of recommending unseen Twitter messages (tweets) relative to a user's typical activity and interests. A graph based link prediction method was used to infer which unseen tweets the users would like by "predicting the presence or absence of edges between nodes" on the graph. [?] K values were tested in intervals divisible by 5 ranging from 5 to 35. The system preformed the best when K=30.

An LDA approach of a tagging system was implemented to improve tag recommendations. [?] Tagging systems are often used for organizaing a user or organization's data. When this data is shared, tagging can be used to find or search for relative content.This system is similar to our proposed book recommendation system that uses subject topics to search for relative material. The initial tag data set was comprised of a large sample creating a diverse tag set. This set was used to elicit latent tags of sources that did not have that many tags to describe the document. Often, the tags recommended were more specifc. By increasing the number of tags and increasing specific tags, their approach contributed to the usefulness of searching for new content.

3. ARCHITECTURE

4. PROGRESS

5. VALIDATION

6. EVALUATION PLAN

In this section, we will talk about methods to evaluate our solutions. This including two parts: methods used to evaluate our topic learning model and methods used to evaluating textbook suggestions which is our final product.

6.1 Evaluating Topic Modeling

In order to carry out a comparative evaluation for a set of candidate recommendation algorithms, evaluation metrics will be used to measure some quality and performance features, in our case, we will use prediction accuracy to evaluate the models. The most popular metric that is used for evaluating prediction accuracy is Root Mean Squared Error (RMSE). RMSE is used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. We will need to run this RMSE analysis on a data set with labels, and we are currently evaluating available data sets on Kaggle for fitness for this task.

$$RMSE = \sqrt{\frac{\sum_1^n (\widehat{y}_i - y_i)^2}{n}} \qquad (1)$$

6.2 Evaluating textbook suggestions

6.2.1 *Pre-Surveying*

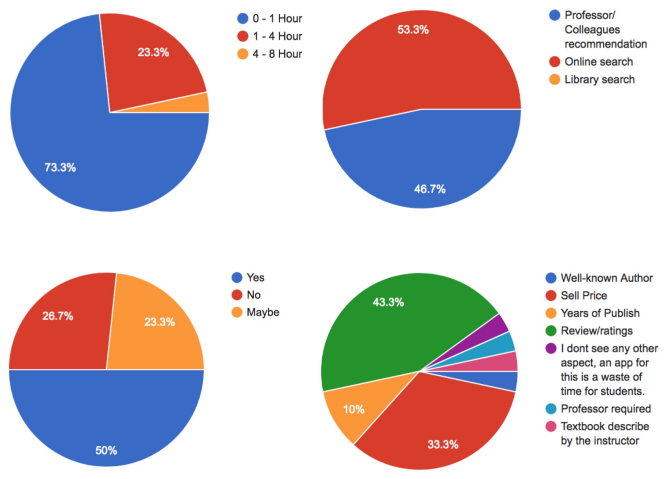
To evaluate our project idea, we conducted a pre-surveying to see what the problem we are facing and how people reacting about our proposed solution. We asked our participants how long they will spend on find the right textbook, what aspect do they think is most important when choosing a textbook (except the content, our learning model will do the best of this part). See Table 1 for a breakdown of the questions to ask.

Table 1: Questions in Pre-survey

Questions
1. Are you a student or professor?
2. How much time do you spend on choosing a textbook?
3. How do you choose a textbook?
4. What aspect do you think is most important when choosing a textbook (except the content)?
5. Would you consider an app that will give you recommended textbooks base on your course syllabus and your preference?

Among the 29 responses, we found that 73.3% responded as they interested to our application to recommend the textbooks. The answer to most important aspect during the textbook chosen, review/ratings and sell price were the highest concern. These preliminary results give us the guidance to design the application and to improve the core learning models. See Figure 1 for detailed results.

Figure 1: Result of preSurvey



6.2.2 *Experiments and Post-survey*

After we build the application UI and core learning model, we plan to run a few experiments with different groups of users to evaluate the solution.

We have briefly discussed how we intend to validate that our model suggests books well and the usefulness of our application overall, but we have not yet discussed how we intend on validating whether or not the textbook recommendations that our algorithm make are topical (cover the topics) and are frugal (are less expensive than other options). To do this, we intend to do some focus grouping easier by some custom UI in our web app.

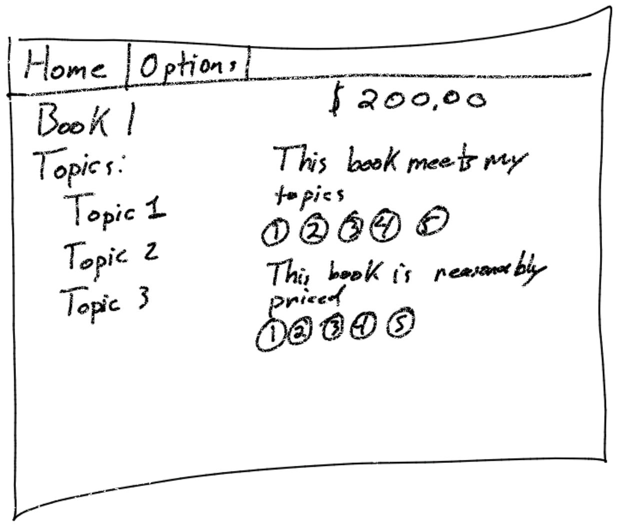
We intend to sit down a group of volunteers with our app and to give them access to our app with an anonymized user. They will be asked to find an appropriate textbook for a class using a provided list of topics. After choosing the appropriate topics and setting their own weighting on frugality and topic coverage, they will be asked to parse the results and rank their fitness for the particular purpose on a scale from 1 to 5 (with 5 being perfect fit) and rate their cost from 1 to 5 (with 5 being perfectly inexpensive compared to the alternatives). After rating the books that were suggested, the user will then be presented with the top 5 books that did not make the suggestion cut, and will be asked if any of these books should have replaced books that were presented and why they should have been on the list (for example, was the book expensive, but did it offer better topic coverage, or was there something unique about it that we should factor into our analysis). The test can then be repeated with the other algorithm.

This test will answer several questions. First, it allows us to evaluate if the topic mining is finding semantically appropriate items. By presenting the user with the top choices, then choices that were not considered optimal, we are attempting to see if the algorithm should have found a different book more relevant. We are also attempting to see if a book that was more or less expensive should have been on the list. More importantly, though, by rating suggestions on a 1 to 5 scale, we can easily compare the ratings of suggestions made by LDA to the ratings of suggestions made by Doc2Vec to see if there is a significant difference between them.

7. CONCLUSION

8. REFERENCES

Figure 2: Mock up of book details view with survey question radio-buttons


Home
Options
$200,00
Book 1
Topics:
Topic 1
Topic 2
Topic 3
This book meets my topics
1 2 3 4 5
This book is reasonably priced
1 2 3 4 5