Knowd: A Tool for Organizing and Recommending Online Learning Resources

Adam Fanslau
Drew University, 2015

ABSTRACT

Traditional courses and degree programs are well structured with a syllabus or curriculum that gives students an overview of topics they should be learning. The internet is home to a vast number of resources for students learning a variety of topics. Search engines provide a way to find information if the user knows what to search for. However, with independent learning, there is no such structure to guide learners. In an attempt to solve this problem, we created a prototype web application called Knowd. The goal of this system is to guide students along a path of concepts and provide a place to organize learning resources. The main features of the prototype are the ability to save, tag and get recommendations for new learning resources and topics. Through data gathered manually by users as well as automatically from the web, we build a graph model of relationships between topics, resources and users, and traverse this graph to deliver personalized recommendations.

Knowd:
A Tool for Organizing and Recommending Online Learning Resources

Adam Fanslau

B.A. in Computer Science

An Honors Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of Bachelor in Arts with Specialized Honors in
Computer Science
at Drew University
May 2015

ACKNOWLEDGMENTS

I would like to thank Dr. Emily Hill for serving as a primary advisor to this project. She provided extensive help and guidance as well as freedom to explore ideas. I thank Dr. Minjoon Kouh and Dr. Jon Kettenring for reading drafts and providing feedback that helped us complete this research. I also want to acknowledge the time and effort provided by participants who were beta testers and submitted survey responses. Thank you to Drew University and its Department of Mathematics and Computer Science for providing me with the opportunity to conduct this research.

Contents

Chapter 1

Introduction

Technology has become more and more ingrained in everyday life, and education is no exception. In the course of learning, many students find themselves looking online for information. Many times, searching for resources online is like trying to drink from a firehose. Students are bombarded with information at varying levels of expertise and prerequisite assumptions. Sifting through this information to find resources at an appropriate level is difficult. There are two main problems that stem from the massive size of the internet: (1) finding new and relevant learning resources and (2) keeping track of learning resources once they are found.

1.1 Finding new learning resources

To illustrate these problems, imagine a student named Alice. Alice wants to learn about a web development framework called Ruby on Rails. There are many resources available online to start learning about the topic. However, they do not take into account the fact that Alice already has experience with a similar framework called Django. Ideally, Alice would like something that creates a comprehensive guide to Ruby on Rails that is personalized based on her previous knowledge.

Currently, Alice might start by searching Google for "Ruby on Rails". In the search results would be the official documentation, a getting started guide and maybe a few tutorials. While this is a good place to start, it is not personalized. Alice wants to learn quickly and does not need to waste her time doing tutorials aimed at beginners. She might have to refine her search query several times to find a resource to learn Ruby on Rails in the context of what she already knows.

Imagine Alice has learned the basics and decides to try her hand at an app idea she has been thinking about. She runs into a problem setting up her database and searches the error on Google. She clicks a result from Stack Overflow, a question and answer site for the programming community. This question describes her problem and the accepted answer seems to solve it. However, she does not have enough experience with the technology to understand why it solves her problem. There might be words or concepts in the answer that she has not learned yet. To understand the context, she might search Google for each word and look through several resources to find one at the right level for her current understanding. Eventually after several web queries, she finds a good explanation, but it takes significant time and effort.

1.2 Keeping track of found learning resources

Another problem arises when Alice becomes more experienced and wants to recall the solution she found previously. After about a month, she finds herself running into a similar problem to the one described in the prior section. She remembers reading about it, but trying to find that good explanation again would require a similar amount of scouring the web. She might have bookmarked the page, but searching browser bookmarks is not an intuitive experience. Bookmark folders quickly become a mess and can be overwhelming to try to organize. There are also issues with accessing these resources from different devices. If Alice could save and search her own bookmarks alongside web results, she would have a much easier time finding resources that are specific to her previous experience, and remembering what she already knows.

In addition to remembering what she already has learned, Alice needs a way to expand her knowledge by learning from others. When starting to learn a new topic, Alice would benefit from not just a list of search results, but a coherent lesson plan about the topic she is searching for. These topic lessons might be curated by an individual, or by a machine aggregating resources from many different sources. Instead of having to develop her own curriculum on the fly, she could find the relevant sub-topics and get suggestions for how to learn them. There are existing tools that solve some of these problems, but none that bring them together in an intuitive way.

1.3 Proposed Solution

To address the challenges of saving and finding personalized resources, we present a web application called Knowd. We have created a prototype that allows users to save and organize their resources into topics and provides recommendations based on what the user has saved.

The remainder of this thesis will describe Knowd in more detail. First we introduce the body of research that surrounds the tool, including areas such as online learning, recommender systems, semantic ontologies, topic modeling, and knowledge graphs. These research areas serve as inspiration and context from which to understand Knowd. Then we explain the process and challenges in designing, implementing and evaluating the current prototype. It is implemented as a web application based on a graph structure of topics and learning resources. We evaluated the user interface with a survey given to computer science students at Drew. We also developed a preliminary model for representing a topic ontology and adapting it to user feedback, which we evaluated using available user data from Stack Overflow. Finally, we will discuss the larger vision of Knowd and future work in reaching that goal.

Chapter 2

State of the Art and Practice

Knowd is a practical tool designed for life-long learners to help record, retrieve and share their learning resources. There are existing tools, generally referred to as Learning Management Systems (LMSs) for teachers to manage and disseminate content related to their courses. Many universities also adapt their courses for online and distribute them—freely or for a fee—via sites like Coursera, Udacity and EdX. These courses are often referred to as Massive Open Online Courses or MOOCs. They have gained traction recently as people are increasingly using technology to circumvent the high costs of college.

In addition to the state of the practice, research in areas like semantic web, knowledge graphs, and recommender systems are relevant to the research goals of this project. To represent and recommend knowledge sources about particular topics, Knowd uses a graph structure of Topics, Resources, and Users.

2.1 State of the Practice

2.1.1 Learning Management Systems

Learning Management Systems (LMSs) are widely adopted in schools and universities as a solution to hosting course content online [16]. Among the most popular are Moodle,[1] Blackboard,[2] and Edmodo.[3] These systems are used to host course content, communicate with students, and perform administrative duties. Teachers can build their syllabi and students can consume and discuss content. LMSs are good for hosting files like presentation slides, assignment instructions and submissions. They are also good for keeping track of grades [8].

Despite their wide use, LMSs are designed to be an environment which is closed to anyone outside the class. While this meets the needs of managing sensitive data like grades and class-specific data like class activities, the actual content that is being taught is relevant to many parties other than the students in the course. Students would benefit from not only the content distributed by their professors, but also content from external sources on the web, including courses at other institutions on the same topic. This feature would also be beneficial to teachers to help them structure their courses based on similar content that already exists.

There is currently no place where experts and novices alike can help create open source content to teach a topic. Instead of the expert being the curator and the student being the consumer, both groups should be able to play both roles. In

[1] `http://moodle.com/`
[2] `http://www.blackboard.com/`
[3] `https://www.edmodo.com/`

addition, there is no recommendation component that compares resources added by students and teachers and searches for resources that might help explain the concepts from a different perspective, or expand into new areas of knowledge. These features would make it easier for course content to adapt to student needs and the external context of the topic.

2.1.2 Online Courses

Sites like Coursera and EdX are helping solve the issue of openness of content. These sites allow universities to host courses referred to as Massive Open Online Courses (MOOCs) and distribute their content to millions of people. While access to content has been vastly expanded [19], the conversation between educators and students is still unidirectional in the sense that the teacher decides what topics are most important to cover, and this decision holds for every student in the class. In reality, the backgrounds and intentions of students enrolled in MOOCs are varied [4].

2.1.3 Implications

Each of these existing technologies answers a piece of the question, but leaves others open. Our end goal is to build an adaptive system that automatically personalizes learning for the individual instead of relying on a one-size-fits-all model. There are several areas of research that are relevant to this goal. The following section describes how the existing body of research is related to Knowd.

2.2 State of the Art

2.2.1 Recommender Systems

Currently, there are many academically developed e-learning systems, but very few are widely implemented and used. Many of these research projects apply traditional recommender algorithms to educational content [3, 21, 22]. Recommender systems generally use two main paradigms: content-based similarity [9] and collaborative filtering [15]. Hybrid methods are also used [2]. These techniques serve as a research base for this project.

Content-based systems use expert-defined features of the items to find pairwise similarities, then recommend similar items to those that the user has liked before. Content-based systems are good for when data exists about each item. For example, Pandora Radio employs a team of experts to manually label their library of songs with musical attributes.[4] Content-based methods have advantages in their ability to analyze new items independent of any user ratings. However, the need to manually extract features from the items poses additional challenges [9].

Collaborative filtering bases its recommendations on the assumption that similar users like similar things. For example, if user A likes movies 1, 2 and 3, and user B likes movies 1 and 3, a collaborative filtering algorithm will recommend movie 2 to user B. This fundamental assumption is formalized and used in systems like Netflix [1]. Collaborative filtering works well when there is a large dataset with many users and many items. The recommendations are inherently dependent on multiple

[4] `http://www.pandora.com/about/mgp`

user ratings. Therefore, collaborative filtering suffers with limited data or new items that have yet to be rated. This is often referred to as the cold start problem. Hybrid systems attempt to solve some of these problems. For example, Schein et al. [13] combine the two approaches to alleviate the problem of recommending new items.

In the area of education, there have been many examples of both collaborative filtering and content-based methods being used to recommend resources to students [22, 21, 3]. However, one problem that both collaborative and content methods face is the tendency to avoid diversity in recommendations. The fundamental assumption in traditional recommenders is that users want similar items. In learning systems, this is especially important because the goal of learning is always to expand one's knowledge. In this context, students need content that is different but related to what they have already learned. To create a successful educational recommender, we have to address this issue.

2.2.2 Ontologies

In this thesis, we propose a way to recommend resources and present related topics using a content-based method in combination with a domain ontology model. Categorizing resources into topics and mapping the relationships between topics will facilitate the ability to recommend similar resources within topics, as well as related topics and resources that connect topics.

An ontology is a graph data structure used to map relationships between concepts of a particular domain in a well-defined, machine readable format. Building an ontology is often a difficult task [17]. Techniques range from manual, crowdsourced [11], to

fully automatic extraction from text [7, 18]. There has also been research on merging ontologies from different sources [5].

Once the ontology is built, it can be used to recommend learning resources for a student. For example, Sosnovsky et al. create an ontology extracted from semantic structure in textbooks and web pages [14] and use it to recommend readings for students answering questions. In this work, we try to build the ontology from user bookmarking activity in combination with simple text analysis, then use learned relationships to recommend learning resources.

Research by Jiang and Tan shows the possibilities of using personalized user ontologies to represent user profiles in learning [6]. Their model uses a form of TF-IDF to represent each document in a keyword and concept vector space. TF-IDF stands for Term Frequency Inverse Document Frequency and measures the frequency of each term in each document divided by the frequency over all the documents. Words that occur many times in all the documents are drowned out so that a high TF-IDF score signifies relative importance of terms in a given document. A term with a high document frequency causes the TF-IDF score to decrease. TF-IDF is high when the frequency in the document is high and the document frequency is low. A more detailed explanation of TF-IDF can be found in Chapter 4.

Jiang and Tan use keywords as well as domain concepts to categorize documents into an ontology which is then used as a basis for personalization. The weights of this ontology are calculated based on user interest. They evaluate their model using semantic search techniques. They use this personalized ontology model to provide search results, which is similar to one of our goals. Knowd is an attempt to not only

provide search results, but to predict and recommend queries that will expand the users' knowledge. Despite extensive academic research in the area, there are not many practical tools that implement the technology. This research attempts to implement a recommender system as a practical tool to help learners manage learning resources.

Chapter 3

Knowd: A Tool for Managing Learning Resources

Knowd is an education tool designed to help students keep track of what they know and find new topics to learn about. Knowd is currently running live on the web at `https://cs.drew.edu/knowd`.

By providing a space for students and teachers to save resources and organize them into topics, Knowd can learn user preferences for different topics and recommend resources and topics for students to learn more about. With data gathered from individual users, Knowd builds a model of what internet resources a student has explored, then compares it with an aggregated model to decide what is the best concept to learn next.

To solve the problem of organizing multiple sources and personalizing one's learning, we developed an initial prototype that allows students to save resources that they

find helpful and organize them into topics. The process of students documenting their learning allows Knowd to build a user profile that serves as a representation of that student's knowledge. We have implemented a simplified variation of existing techniques to build an ontology from the content that users contribute as well as content retrieved from web searches.

Knowd provides a place to store and find resources, but more importantly it serves as an interface to build an ontology of topics that users care about. As users add and organize resources, the system incorporates each user's opinion to form an aggregate knowledge base. The remainder of this thesis will discuss the challenges and results of designing and implementing both the interface and the algorithm to aggregate knowledge gained from many sources.

The goal is to build a knowledge graph that learns from user feedback to enable personalization. The main research questions of this project are as follows:

1. Can we design a platform that incentivizes students to document and share their learning resources?

2. Can we create a usable application to organize those learning resources?

3. Can we learn from user activity to create a personalized ontology of topics and recommend additional useful learning resources?

In order to answer these questions, we built an interface for users to save and organize learning resources. This prototype acts as an infrastructure to expose the ontology to users in a way that makes contributing to that ontology intuitive. Because we need user feedback to learn topic relationships and user preferences, the user

interface needs to be easy to use, and it needs to clearly present the value proposition of Knowd to its users.

Figure 3.0.1 shows a screenshot of the general look and feel of the Knowd web application. The interface is simple so as to not distract the user. There are four main features of this prototype: tagging existing resources, bookmarking new resources, viewing existing resources and recommendations, and searching for resources.

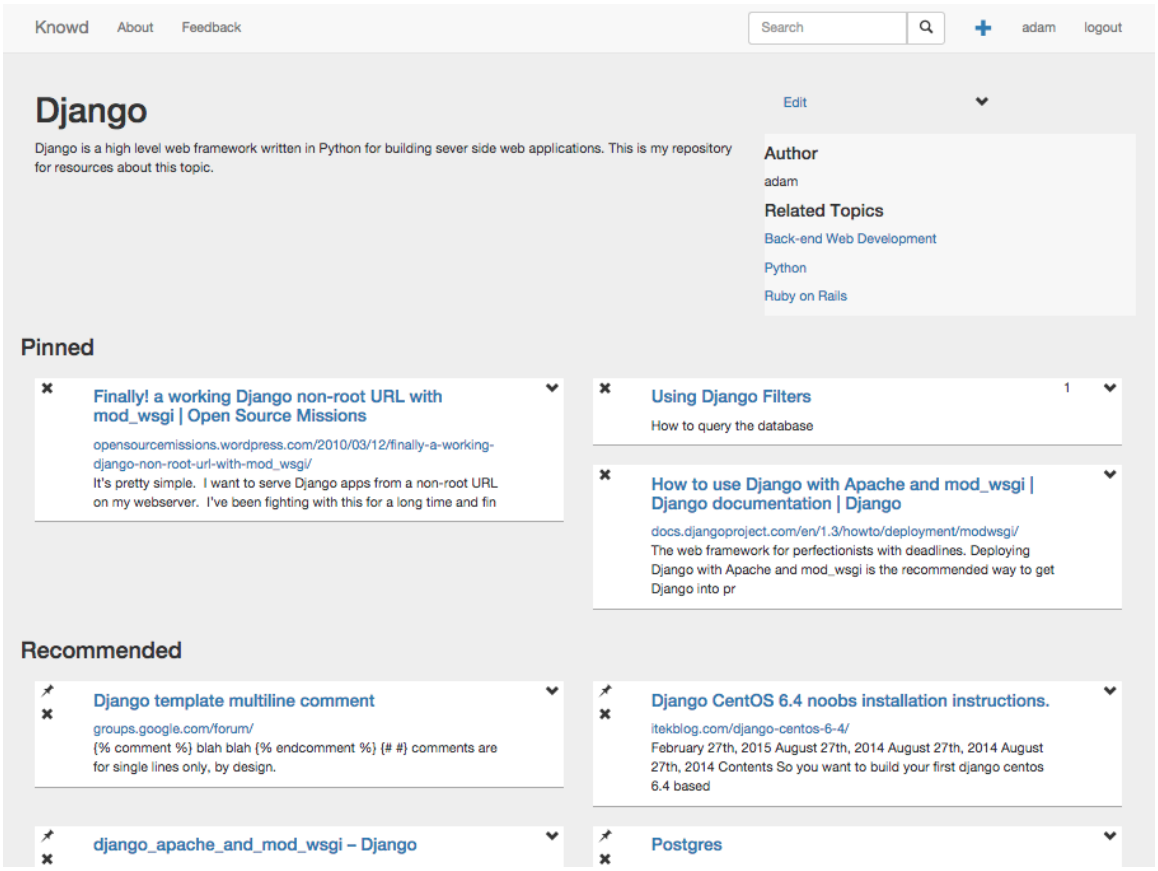

FIGURE 3.0.1: Knowd UI for Topic Page: Django

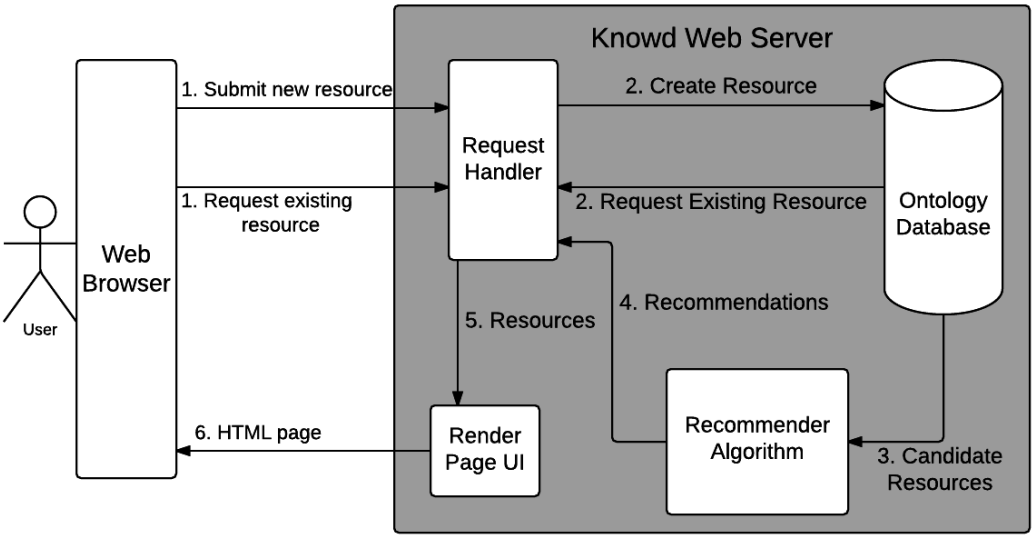

FIGURE 3.1.1: Knowd System Architecture

3.1 Architecture Overview

Knowd is built using a web application framework called Django which uses the programming language Python to create dynamic websites. We host Knowd on a local server on campus that runs CentOS, which is a Unix-based server operating system and Apache web server. Data for Knowd is stored in a PostgreSQL database.

Figure 3.1.1 shows the architecture overview of the system. There are two main ways the user interacts with the system: submitting a new resource, or requesting an existing resource. When submitting a resource, Knowd will go out and retrieve the contents of the web page, and save the data to its database. Then, when requesting an existing resource, Knowd first queries the database for that resource, then it finds related resources using the recommender algorithm described in Chapter 4.

The ontology database stores all users, documents and topics added to the system, as well the relationships between them. There are three main entities in Knowd's database: User, Relation, and Resource. For the purpose of a mental model and simple vocabulary, the terms tag and document are used to describe one word resources and multi-word resources respectively. However, from an implementation standpoint, these two entities are both instances of the Resource class: a tag is simply a Resource with only a one word title field. Relations are used to model an edge between two resources from the perspective of a given user as well as the relationship between a user and a resource.

Figure 3.1.2 shows our data model with an entity relationship diagram. The arrows represent a one-to-many relationship, and the circle represents an optional field. Each relation must have a parent_resource and a user_perspective. When a particular user tags a document, the Relation object assigns the tag as its parent_resource, and the document as its child_resource. The user is set as the user_perspective field, so that a Relation object exists for each unique user who made that tag action. An empty child_resource represents a user-resource relationship. In this case, the relation object is used to store a user's interactions with a resource such as the number of times that user has visited the resource.

3.2 Features of Knowd

The current prototype enables users to submit resources, tag or pin them to each other to organize topics, and get recommendations for related topics and resources to

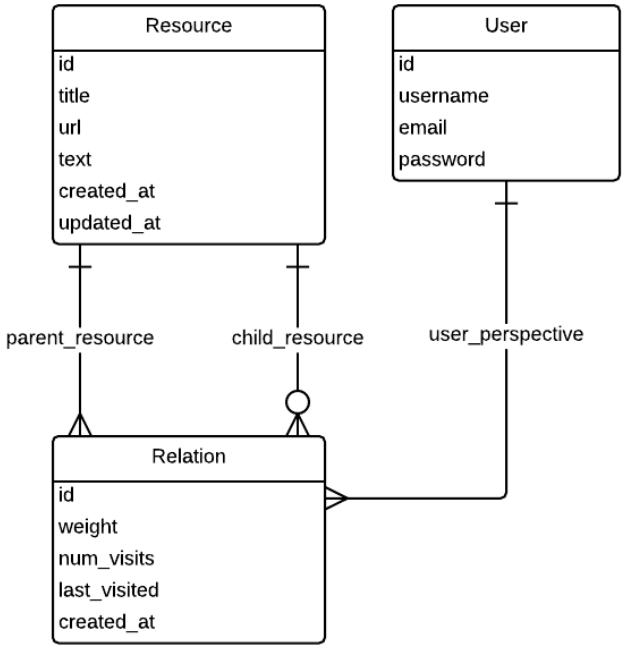

FIGURE 3.1.2: Knowd Data Model

learn from. Relations can be created between any two tags, any two documents, or one tag and one document. Due to this flexibility, in the context of pinning, the term "resource" is used to refer to either a document or a tag. The verbs "tag" and "pin" are also used interchangeably. Both words mean to create a relation object between two resources.

3.2.1 Tagging Existing Resources

Each user has a home page where he or she can save the resource he or she uses most. For example, Alice might store tags like "Web Development" or "Databases"

so she can access them easily. The screenshot in Figure 3.0.1 shows an example topic called "Django", which is the web development framework. The topic page consists of a section of resources that the user has pinned to this topic and a section with recommendations. Users can add or remove resources by using the pin button (A) and the X button (B) located at the corners of each resource. The user interface for this feature is shown in the screenshot in Figure 3.2.1.

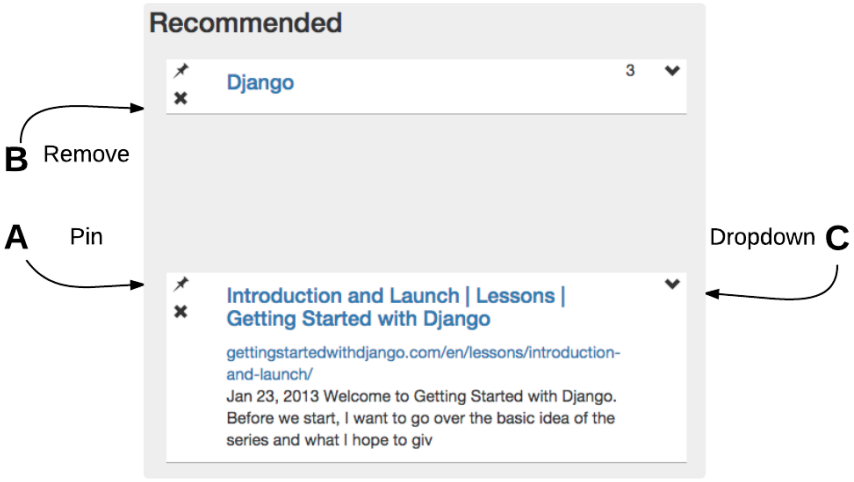

FIGURE 3.2.1: Interface to manage pins

The pinning feature can be used to connect any resource to any other resource. For example, a resource called "Django" might have several resources pinned to it, including web documents such as documentation or code examples. Users might also pin sub-topics such as "Model View Controller", which is a design pattern used in Django to separate data from presentation. In addition to pinning to the current topic, the user can also pin to their home page, or another resource by clicking the menu button in the opposite corner of each resource (C). The menu is shown below

in Figure 3.2.2. Clicking "Pin to another Topic" brings up a popup where the user can select a destination topic. This popup box is shown in Figure 3.2.3.

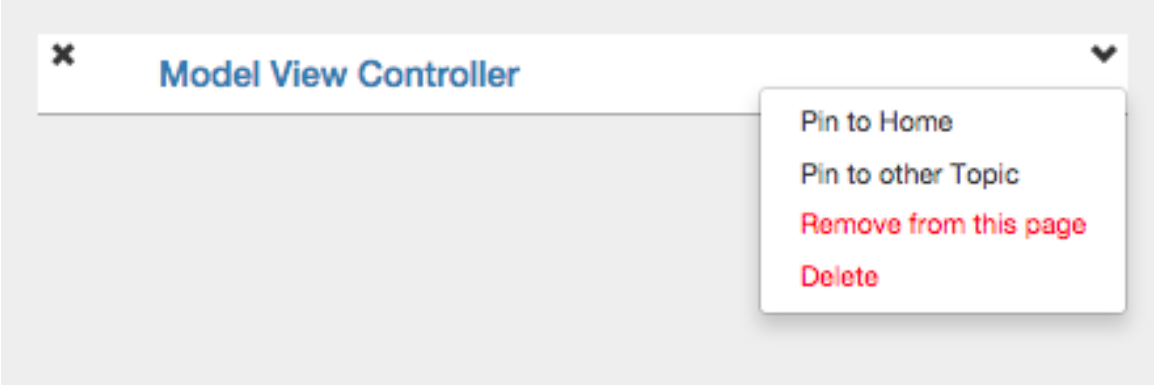

FIGURE 3.2.2: Resource menu - more options

The relationships between these resources are stored using the data model in Figure 3.1.2. The topics "Django" and "Model View Controller" are both represented by the title field in their respective Resource objects. Resources can also store other data like a url or body text. When the user makes this connection from the UI, Knowd creates a relation object in the database that holds data about that relationship such as the parent and child resources, the user who created it, and the time it was created. Pressing the X button will cause the relation object to be removed. For example, in learning about "Django", the user might pin several resources, then decide that one is irrelevant and remove it from his or her view. These actions are used to inform the system about a user's opinion of what topics are related and to improve the recommendations given to that user. A detailed description of this process is presented in Chapter 4.

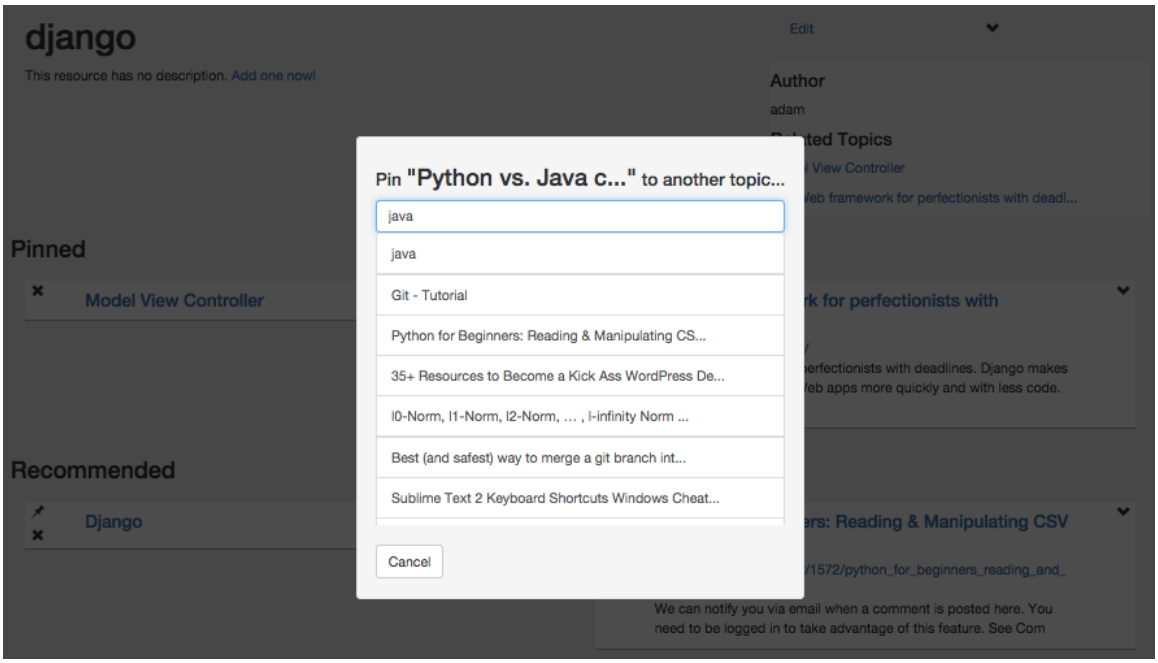

FIGURE 3.2.3: Interface to choose a destination topic

3.2.2 Bookmarking New Resources

Users can save resources from the web by clicking the + button on the top toolbar (see Figure 3.2.4). Resources can be created using any combination of these fields including url, tags, title, or custom text. Knowd will attempt to fill in any missing details by scraping content from the web page. This allows users to save web resources, as well as save their own notes on a topic. From this form, resources can be saved to either the user's home page, or the resource currently being viewed.

When the user submits this form, several processes are triggered on the back end. First, Knowd will create a resource object with all the provided information. If the user submitted a URL, the system will then access the url and cache the contents of the web page. It uses the web page contents to fill in any missing information such

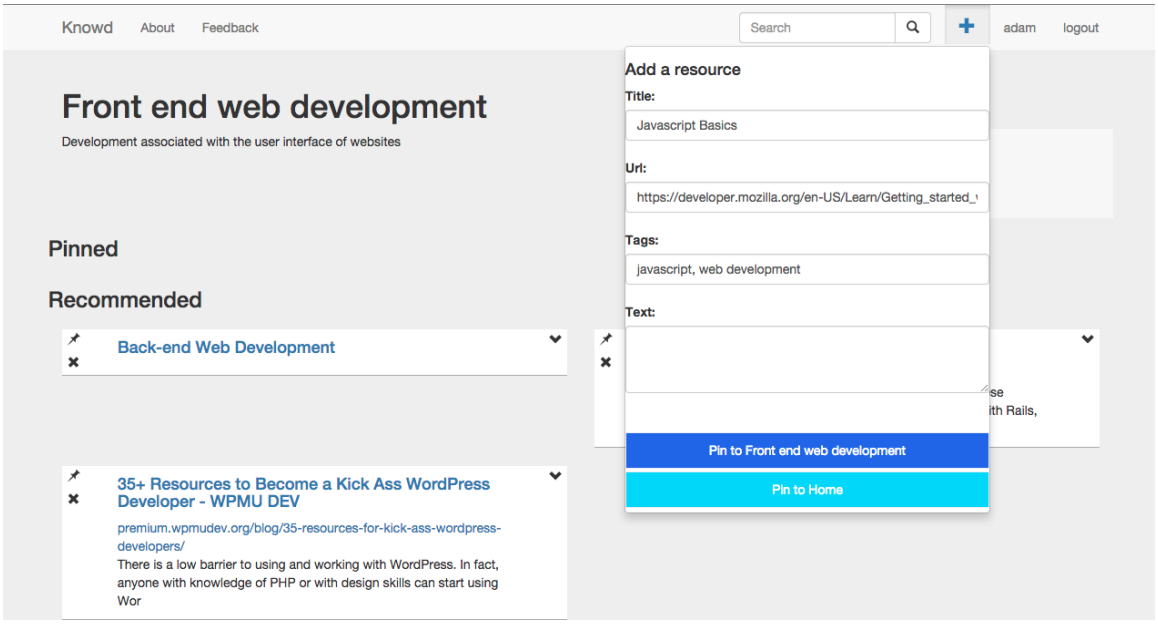

FIGURE 3.2.4: Interface to add a new resource

as title and description. The logged in user is set as the author for that resource and the resource is saved to the database.

If any tags were provided, a relation object is created between the new resource and each tag in the list. If the tag does not exist, it gets created on the spot. The system also uses its recommender engine to predict tags for the new resource. These relations get created with a user_perspective field as the default system user. All other relations are created from the perspective of the logged in user.

Depending on how the user submits the form, different relations will get created. If the user clicks the "Save to <Current Resource>" button, or simply presses the enter key to submit the form, a relation gets created between the new resource and the resource or tag currently being viewed. The page is then refreshed to reflect this change. If the form is submitted using the "Pin to Home" button, a relation is created

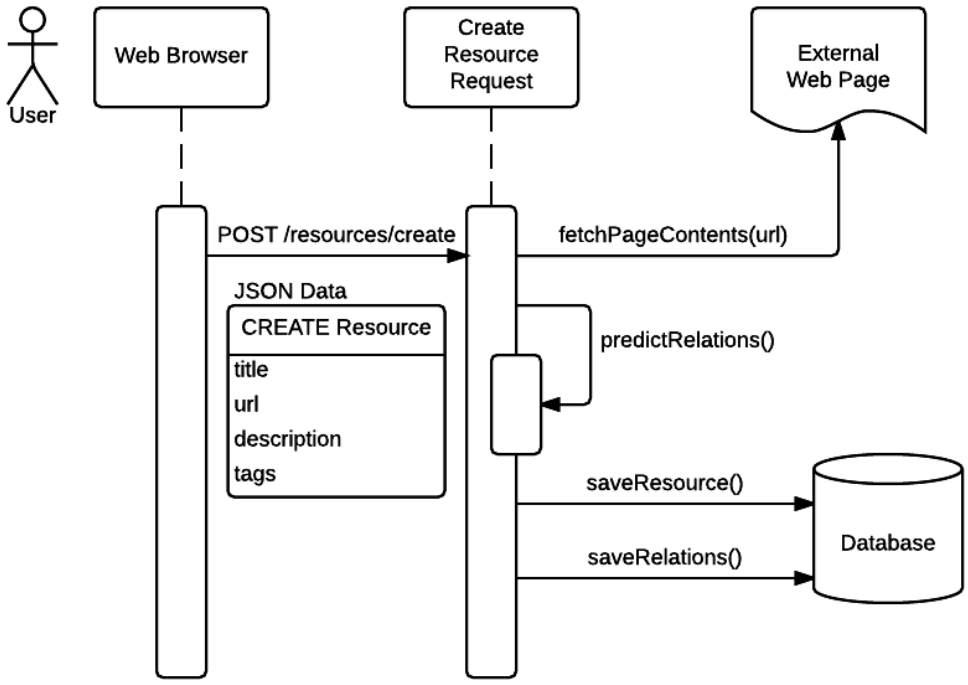

FIGURE 3.2.5: Implementation of adding a new resource

between that user's Home tag and the new resource. An alert appears that informs the user that the resource was saved successfully.

This process is outlined visually in the sequence diagram in Figure 3.2.5. When the form is submitted, the data is sent to the server as JSON, which is simply a way to format structured data a textual representation. The server then goes out to fetch the web page contents if a URL was submitted by the user. It then attempts to fit the resource into the ontology by predicting relationships to existing tags or documents. It finally saves all these entities to the database.

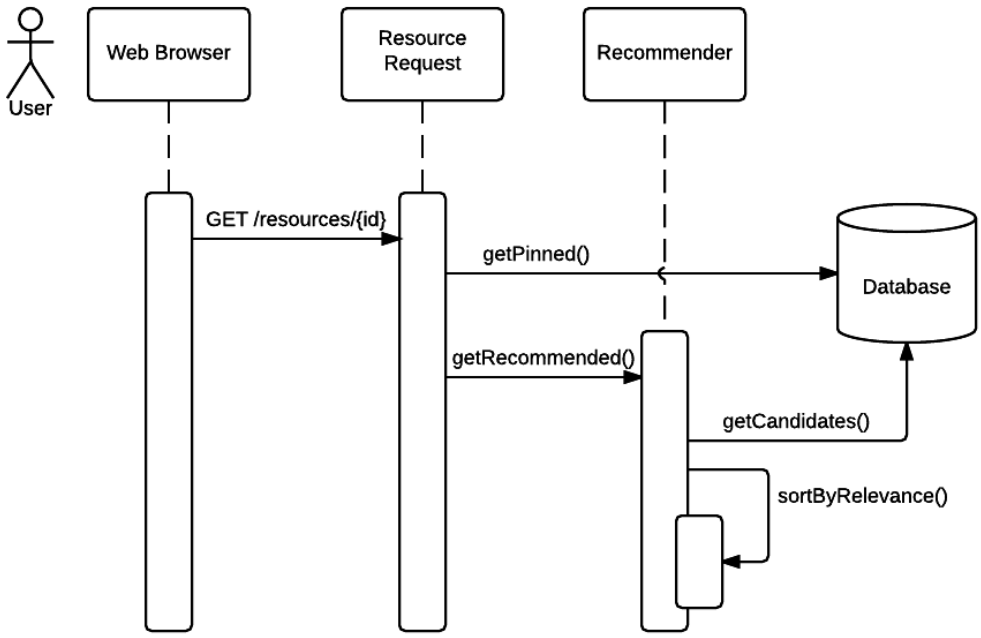

FIGURE 3.2.6: Implementation of viewing a resource

3.2.3 Viewing Existing Resources

Figure 3.2.6 is a visual representation of the following process. When users click on the title of a resource, they are taken to a web page that shows the details of that resource, any resources that the current user has pinned to it, and recommendations of related resources. When this page is requested, Knowd accesses this resource in the database, and follows all relations associated with that resource. The system then gathers recommended resources and displays them to the user. In the current system, the recommendation process happens by following relations created by all the users and gathering search keywords. The database is then searched with these terms, and the results are sorted by a relevance score. This process is described in Chapter 4.

3.2.4 Searching for Resources

The database search process uses the open source Django library django-watson,[1] which implements keyword-based search in PostgreSQL. In a simple keyword search, django-watson ranks the results based on similarity to the query.

When recommending resources, search terms are generated by following Relations created by all users. For each query, the top ranked results are added to a candidate set. These candidates are then ranked by a weighted sum of various features including textual similarity to the current Resource and interactions by users. This preliminary relevance score is used as a placeholder for a more sophisticated recommender algorithm that attempts to personalize the recommendations, described in Chapter 4.

[1] `https://github.com/etianen/django-watson`

Chapter 4

Representing the Ontology and Generating Recommendations

With both collaborative filtering and content-based recommenders focused on similarity, users get limited diversity in their recommendations [20]. These algorithms recommend items similar to the ones liked by users in the past. This eliminates the possibility of stumbling upon loosely related resources to expand the user's knowledge. This is especially important for Knowd because of the focus on expanding the student's knowledge. The challenge is to provide users with resources that are similar to what they know, and yet introduce something new in order to further their learning.

We propose a solution to this problem by using a topic ontology to not only provide similar recommendations, but to also expand into new areas. To accomplish this, Knowd uses a data model based on a graph structure where nodes can be a user, tag, or document. We use the word "topic" to mean a general area of interest.

"Tag" is an object used to label a document as a particular topic. These two terms are interchangeable.

Each node in the topic ontology graph can have a weighted edge to any other node. Weights for relationships between users and tags can be interpreted as that user's preference or interest in the topic. Tag-document relationships describe how much that document is about a particular topic. Edges between the nodes of the same type represent the similarity between them. Figure 4.0.1 shows an example hierarchy of tags, users and documents. Each line represents an edge in the graph and has a weight which measures the strength of that relationship. In practice, the graph is fully connected, meaning there is a weight between all pairs of nodes. However, for clarity, the figure only shows the edges where the weight is above some threshold. These weights are formally defined below in Section 4.1.

For example, the user Bob has a high weight to the tags "Back-End Development" and "Django". His weight to the other topics would be close to 0, quantifying his lack of displayed interest. One goal of this system is to predict the user's future interest in a particular topic and guide them to that learning with resources that are relevant to what they know and what they are trying to learn. The dotted edges in Figure 4.0.1 show an example of tags or documents that might be recommended to the user. Since Alice has a high weight to "Front-End Development", Knowd might recommend resource number 7, which is about both Javascript—something related to what she already knows—and Ruby on Rails, which is something new.

Knowd users can privately organize their resources by tagging them. Using this tagging action, the system creates a relationship between the tags and the resources.

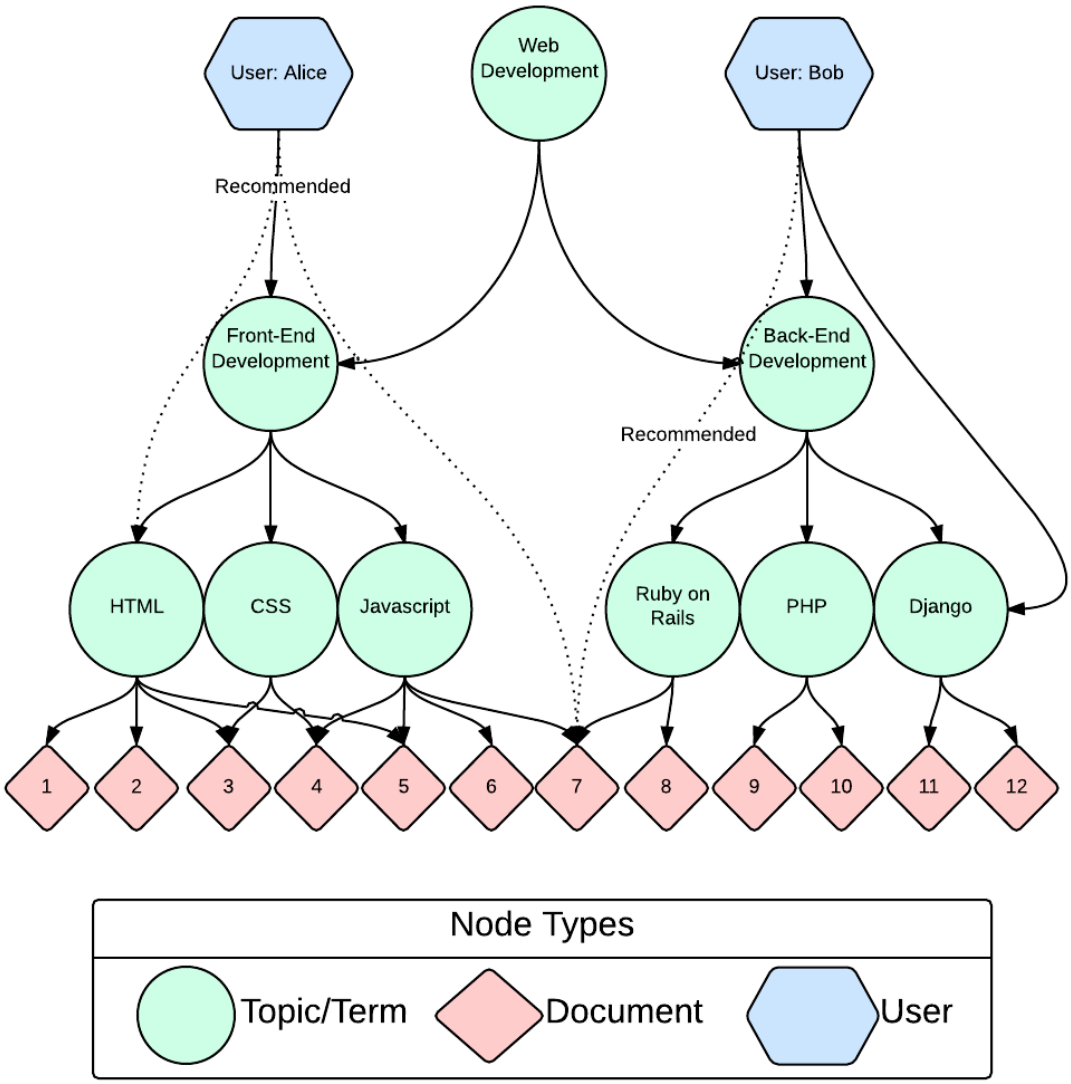

FIGURE 4.0.1: Knowd Data Model Diagram

By remembering user activity like tagging and adding resources, Knowd attempts to predict how users might tag another resource, or that the user would enjoy learning about a certain related topic. In the process of organizing, each user creates relation-

ships between Tags and Resources from their perspective. For example, our example user Alice might save a resource under the "Django" tag, but someone else might think differently and tag it more generally as "Back-end web development". Each of these tags creates a relationship in the graph and they are aggregated along with textual analysis to form an overall opinion about the topic of that document.

Figure 4.0.1 shows an example of how these topics and resources might be organized. Relationships in the graph represent the organization of topics and resources from the perspective of different users. By aggregating these relationships in different ways, we can personalize recommendations as well as model how topics are related to each other.

The next section describes in more detail how the weights of these relationships are calculated and how the graph is used to generate recommendations.

4.1 Calculating Edge Weights

From the perspective of the user interface, users take an action to tag a given resource. For example, a resource that explains how to make a button on a web page trigger an action might be tagged as "javascript" and "html". These one-word tags broadly represent the topics that classify the resource. Because these words also occur in the text of the documents, we use both user tag actions and the term frequency in the document itself to assign edge weights. In context of the following section, we will refer to tags as "terms" and resources as "documents". This is standard terminology for text analysis methods. Specific to Knowd, we use the word "topic" to refer generally

to what a document is about. "Tag" represents the entity that users interact with and "term" is the actual word that is counted in the text and that is associated with the tag or topic. The action of tagging influences the edge weight between the term and the document.

To determine the weight, or relatedness of a document to a particular term or topic, we first use a commonly used measure called TF-IDF, or Term Frequency Inverse Document Frequency [12]. This is commonly used in fields of text mining and information retrieval [10]. TF-IDF describes term importance in a document by measuring the frequency of each term in the document relative to the frequency in all other documents in the system. TF-IDF is calculated by:

$$\text{tf-idf}(t_j, d_i) = freq(t_j, d_i) * idf(t_j, D) \tag{4.1.1}$$

$$idf(t_j, D) = \log \frac{\|D\|}{1 + n_j} \tag{4.1.2}$$

where $freq(t_j, d_i)$ is the number of times term t_j occurs in document d_i.

IDF stands for inverse document frequency. Document frequency is defined as $df(t_j) = \frac{n_j}{\|D\|}$ where n_j is the number of documents in which the term appears and $\|D\|$ is the total number of documents in the set of all documents D. Document Frequency is simply the percentage of all documents that the term appears in. The log function is used to smooth out comparisons between very large values and very small values. Because df is less than one, we invert the fraction to get rid of the

resulting negative value from the log function. We also add 1 to n_j to prevent division by zero in the case where the term does not occur in any documents.

The initial vocabulary set is generated by taking k terms with the highest *idf* values, which eliminates stop words that are used frequently across all the documents. For example if the word "code" appears in 90 percent of the documents, it will have a lower *idf* than the word "html" which might occur in 20 percent of the documents. In a set of documents about computer science, the frequency of the word "code" gives very little information about the contents of that document.

The terms in the resulting vocabulary are then used as features to represent documents and users as vectors. Document and user vectors are composed with the edge weights from that document or user to each term in the vocabulary. For example, document vector d_i and user vector u_k are respectively defined as,

$$d_i = \Big[W_{TD}(t_1, d_i), \quad W_{TD}(t_2, d_i), \quad ... \quad W_{TD}(t_n, d_i)\Big] \tag{4.1.3}$$

$$u_k = \Big[W_{TU}(t_1, u_k), \quad W_{TU}(t_2, u_k), \quad ... \quad W_{TU}(t_n, u_k)\Big] \tag{4.1.4}$$

where the elements of vector d_i are the weights $W_{TD}(t_j, d_i)$ between that document d_i and each term t_j in the vocabulary set. These weights are defined in Equation 4.1.5.

$$W_{TD}(t_j, d_i) = (1 - \alpha_U) * \text{tf-idf}(t_j, d_i) + \frac{\alpha_U}{\|U(t_j, d_i)\|} \sum_{u_k \in U(t_j, d_i)} W_{TU}(t_j, u_k) \tag{4.1.5}$$

We use tf-idf to initialize the weight W_{TD} between term t_j and document d_i.

As users interact with different documents, this weight is updated to reflect that feedback. For example, if our user Alice is learning about front end web development, there might be a document that has the same tf-idf score for the terms "html" and "css", but Alice as a human knows that this resource is best used for learning "css". She explicitly tags it as "css". We want this human-created feedback to inform the system's representation of that topic. If many other users take the same action, we want to trust the human opinion over the machine opinion generated from tf-idf. To do this, we use both the tf-idf score and the users' tagging decisions to calculate the weight between a term and a document.

The term-document weight is calculated as in Equation 4.1.5, where $U(t_j, d_i)$ is the set of users that have tagged document d_i with term t_j, and $W_{TU}(t_j, u_k)$ is the weight between term t_j and user u_k (as defined in Equation 4.1.6).

$$W_{TU}(t_j, u_k) = \frac{\alpha_A}{\|A(u_k)\|} \sum_{d_i \in A(u_k)} W_{TD}(t_j, d_i) + \frac{\alpha_V}{\|V(u_k)\|} \sum_{d_m \in V(u_k)} W_{TD}(t_j, d_m) \quad (4.1.6)$$

The α_U parameter is bound between 0 and 1 that determines the relative strength of the users' collective opinion. If α_U is high, tag actions taken by users hold a higher weight than the machine text analysis of the document. The second term in this equation takes the average expertise over all the users who tagged this document with term t_j. The more interest a user has in a particular topic, the system should trust that user's tag action more than someone who has no interest in the topic. W_{TU} measures this interest or expertise and is defined in Equation 4.1.6.

As users tag and contribute content to the site, the system calculates their interests in each topic t_j by Equation 4.1.6, where $A(u_k)$ is the set of documents that user u_k has added to the system and $V(u_k)$ represents the documents the user has viewed.

The intuition here is that a user should have a high weight for a particular topic if he or she interacts with many resources about that topic. W_{TD} defines the edge weights between each topic or term and each document. If the user interacts with documents that have high weight for a particular topic, that user is interpreted to have high interest in that topic. Again, the α parameters are used to vary the relative strength of the terms. For example, if $\alpha_A > \alpha_V$, W_{TU} will be influenced more by documents added than documents viewed. We divide each sum by the size of the set $\|A(u_k)\|$ and $\|V(u_k)\|$ to get the average W_{TD} over the documents in each set. This score broadly measures on average how related the topic is to all documents the user has interacted with. Because this weight is determined by both the documents the user views and adds to the system, it can be interpreted as both interest and expertise.

The weights between two nodes of the same type, for example, two users, two topics or two documents, are simply the similarity between the two nodes. We use a standard cosine similarity measure shown in Equation 4.1.7, where v_1 and v_2 can be either a document vector d or a user vector u as defined above.

$$sim(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \, \|v_2\|} \tag{4.1.7}$$

To find the similarity between two terms, we use the same similarity measure, where v_1 and v_2 are vectors composed of the weights between the term and all the

documents in the system. These are column vectors from the term-document adjacency matrix W_{TD}. The similarity between two terms can be thought of as the frequency with which they co-occur together in the same documents. If they often occur together, the W_{TD} values will be high for both terms in many documents, thus the dot product will be high. Conversely, if they never occur together, the dot product will be low. Because W_{TD} includes both tf-idf and user tags, the similarity between two terms or topics includes information from both the content of the documents and users' opinions. This similarity is then used to define a term vector where $W_{TT}(t_i, t_j)$ is the similarity between term t_i and t_j.

$$t_i = \left[W_{TT}(t_i, t_1), \quad W_{TT}(t_i, t_2), \quad ... \quad W_{TT}(t_i, t_n) \right] \tag{4.1.8}$$

Once these weights have been initialized, they will be updated as users continue to interact with the system. As users interact with content on Knowd, the term-user weights W_{TU} will be updated. Since the term-document weights W_{TD} also depend on the term-user weights and vice versa, this becomes a recursive process. If a document that a user has saved is tagged similarly by other users, the Equation 4.1.5 shows that the term-document weight will increase. For example, a resource that Alice submits might start out with a 0.5 weight to the term "html" and a 0.5 weight to "css". This initial weighting will come from tf-idf. However, after many users tag it as "css", the weight to that term might increase to 0.8. Because this document is now interpreted as being about "css" more than "html", in the next iteration, Alice's term-user weight to "css" should increase.

By interpreting these as edge weights in a graph, the system can now recommend resources to users by traversing the graph. The following section describes this traversal recommendation process. See Section 4.3 for a concrete example walking through these equations with actual numbers. Refer to Appendix B for a table listing all the variables used in this chapter.

4.2 Making Recommendations

Once the graph of document and topic relationships has been created, recommendations are generated by traversing this graph starting from the current topic t_{cur} being viewed. In contrast to traditional recommenders like Netflix that recommend items the user might like, Knowd is trying to recommend topics or documents that the user might be interested in and that are related to the current context.

Algorithm 1 Recommendation Algorithm for Topics

1: **procedure** RECOMMENDTOPICS(t_{cur}, u_{cur})
2: CandidateTopics $\leftarrow \{t_{cand} \in T : W_{TT}(t_{cand}, t_{cur}) > \theta_T\}$
3: $Rel_{TT} \leftarrow \{\}$
4: **for** $t_{cand} \in$ CandidateTopics **do**
5:

$$relevance \leftarrow W_{TT}(t_{cur}, t_{cand}) * W_{TU}(t_{cand}, u_{cur}) \tag{4.2.1}$$

6: $Rel_{TT} \leftarrow Rel_{TT} \cup \{(t_{cand}, relevance)\}$
7: **end for**
8: **return** Rel_{TT}
9: **end procedure**

Algorithms 1 and 2 show the process used to recommend topics and get related resources. To recommend topics related to the current topic and the current user, we

first gather a set of candidate topics by following all edges from the current topic with weights W_{TT} above some threshold value θ_T. Then we compute the relevance score for each candidate topic with Equation 4.2.1. The weight $W_{TT}(t_{cand}, t_{cur})$ measures the similarity between the two topics, and the weight $W_{TU}(t_{cand}, u_{cur})$ represents the user's preference for the candidate topic. By multiplying these two values, both of which are between 0 and 1, we are giving the candidate topic a high relevance if it is both related to the current topic, and preferred by the user. If either one of those parameters are low, the relevance will also be low. It may be helpful to think of this relevance as the combined cost of following a path from topic t_{cur} to t_{cand} to u_{cur}, each of which are nodes in the graph.

Algorithm 2 Recommendation Algorithm for Documents

1: **procedure** RECOMMENDDOCUMENTS(t_{cur}, u_{cur})
2: $Rel_{TD} \leftarrow \{\}$
3: **for** $t_{cand} \in$ RecommendTopics(t_{cur}, u_{cur}) **do**
4: RelatedDocuments $\leftarrow \{d_{cand} \in D : W_{TD}(t_{cand}, d_{cand}) > \theta_D\}$
5: **for** $d_{cand} \in$ RelatedDocuments **do**
6:

$$relevance \leftarrow (1 - \beta_{diversity}) * W_{TD}(t_{cur}, d_{cand}) + \beta_{diversity} * W_{TD}(t_{cand}, d_{cand}) * W_{TT}(t_{cand}, t_{cur}) * W_{TU}(t_{cand}, u_{cur}) \quad (4.2.2)$$

7: $Rel_{TD} \leftarrow Rel_{TD} \cup \{(d_{cand}, relevance)\}$
8: **end for**
9: **end for**
10: **return** Rel_{TD}
11: **end procedure**

Recommending documents follows a similar pattern. First we get a set of recommended topics using the above method presented in Algorithm 1. Then for each

of those candidate topics, we gather a candidate set of documents with weights $W_{TD}(t_{cand}, d_{cand})$ greater than some threshold θ_D. We compute a relevance score for every document according to Equation 4.2.2. This equation is similar to Equation 4.2.1 in that it is computing the cost associated with following edges to get from the current topic t_{cur} to each candidate document d_{cand}. The first term is the weight or relatedness of the candidate document to the current topic. The second term is the weight or relatedness of the document to its corresponding related topic combined with the current user's interest in that topic. By varying the $\beta_{diversity}$ parameter, we can tune the system to recommend documents related to the current topic, or documents that are related to a similar topic. A high $\beta_{diversity}$ value will lean toward the latter situation, and a lower value focuses the recommended documents to only the current topic.

4.3 Concrete Example

To show this in a concrete example, we use a contrived tf-idf matrix and use it to initialized the weights W_{TD}. The matrix below shows these weights, where each row represents a document vector, and the columns represent terms. Recall that in our approach terms are treated as topics. The terms used in this example are as follows, where the term in position j corresponds to column j in the matrices.

$$T = \begin{bmatrix} WebDevelopment & Django & HTML & Databases & Algorithms \end{bmatrix} \quad (4.3.1)$$

$$W_{TD} = \begin{bmatrix} 0.23 & 0.18 & 0.27 & 0.05 & 0.00 \\ 0.28 & 0.19 & 0.22 & 0.00 & 0.00 \\ 0.13 & 0.03 & 0.38 & 0.04 & 0.00 \\ 0.05 & 0.05 & 0.38 & 0.05 & 0.09 \\ 0.03 & 0.27 & 0.04 & 0.28 & 0.06 \\ 0.06 & 0.21 & 0.00 & 0.34 & 0.00 \end{bmatrix} \tag{4.3.2}$$

In the example W_{TD} matrix in Equation 4.3.2 above there are 5 columns for the 5 terms in the vector T and 6 rows for the 6 hypothetical documents in our example.

$$W_{TT} = \begin{bmatrix} 1. & 0.71 & 0.76 & 0.27 & 0.14 \\ 0.71 & 1. & 0.47 & 0.82 & 0.43 \\ 0.76 & 0.47 & 1. & 0.21 & 0.53 \\ 0.27 & 0.82 & 0.21 & 1. & 0.44 \\ 0.14 & 0.43 & 0.53 & 0.44 & 1. \end{bmatrix} \tag{4.3.3}$$

Equation 4.3.3 shows the topic similarity matrix, W_{TT}, for our example. Topic similarity is calculated using cosine similarity as defined in Equation 4.1.7. This produces a symmetrical matrix. For example, the element at row 2 column 4 represents the similarity between the topics "Django" and "Databases" (0.82).

Each of the documents in D is added by a user. In our example, we talk about a user Alice who authors documents 2, 5 and 6. Another user Bob might add documents 1, 3 and 4. Each user's documents are used to determine their weights to each topic by taking the average topic-document weight over all the documents that the user adds.

The W_{TU} matrix in Equation 4.3.4 shows these weights. Rows one and two represent Alice's and Bob's preferences respectively. Alice's preferences are represented in the second row of this matrix. For example, her preference to topic 2 ("Django") is 0.22.

$$W_{TU} = \begin{bmatrix} 0.12 & 0.22 & 0.08 & 0.21 & 0.02 \\ 0.14 & 0.09 & 0.34 & 0.05 & 0.03 \end{bmatrix} \tag{4.3.4}$$

Figure 4.3.1 shows some of these weights graphically. For example, Alice is currently viewing a topic "Web Development" (term #1). The bolded circle represents the current topic, and the bolded hexagon represents the current user. This topic is related to other topics like "HTML" (term 3), and "Django" (term 2). These edges might have a higher value for W_{TT} which signifies they are highly related. For example, in the W_{TT} matrix above in Equation 4.3.3, the weights are 0.76, and 0.71, respectively. The terms "Web Development" and "Databases" (terms 1 and 4, respectively) are not as tightly related and thus have a lower weight W_{TT} of 0.27. The weight to "Algorithms" (term 5), which is mostly unrelated, is 0.14.

To recommend topics following the algorithm described above, we collect related topics to "Web Development" by following edges over a threshold. If we set $\theta_T = 0.2$, our candidate set would consist of "HTML", "Django", and "Databases", but not "Algorithms". The figure shows only the edges above this threshold, however the graph is fully connected in the sense that one can describe a weight between a node and any other node in the graph. The relevance of each of these candidate topics to the current topic "Web Development" depends on the weights between the topics W_{TT} and the current user's interest W_{TU} in that topic.

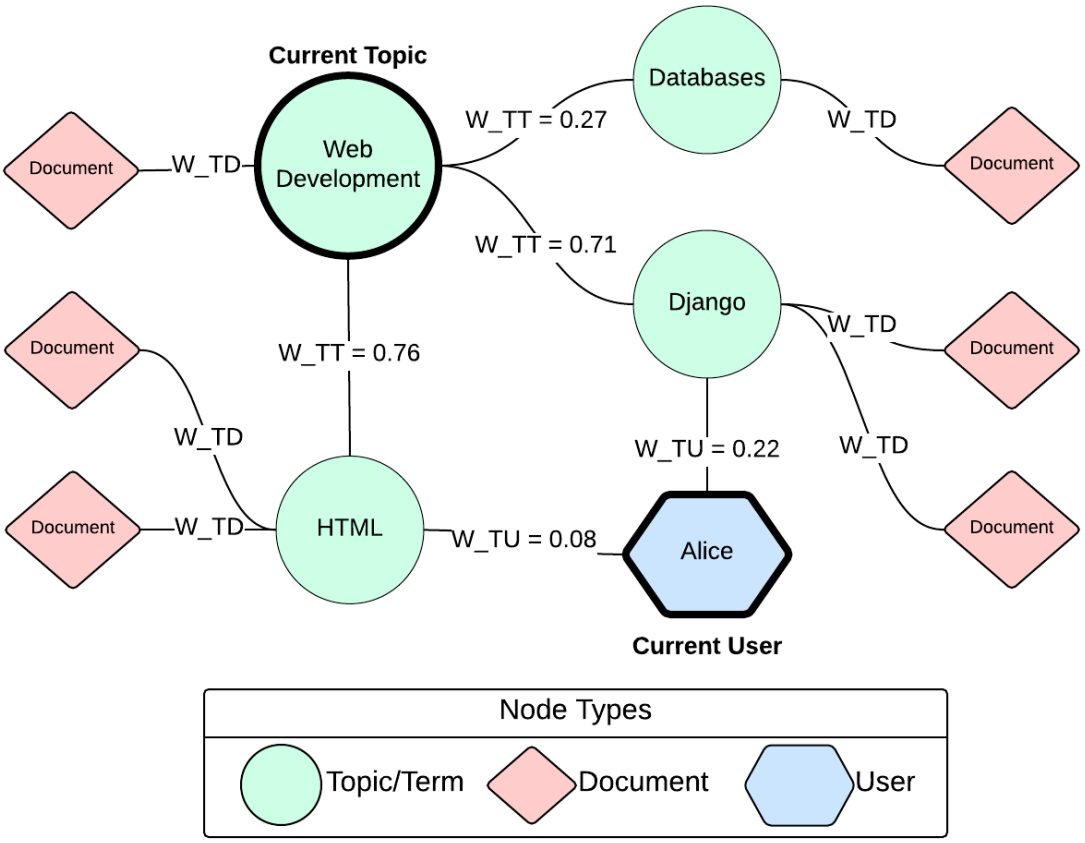

FIGURE 4.3.1: Example topic graph with weights

For example, the relevance score for the topic "HTML" would be W_{TT}, which is 0.76, multiplied by W_{TU}, which is 0.08, giving a result of $Rel_{TT} = 0.06$. Because Alice is not very interested in HTML, the relevance is lower than something she is interested in, like "Django", which has a relevance of $0.71 * 0.22 = 0.16$. Even though HTML is more related to the current topic, the user's preference for "Django" overpowers this relationship. As the user-topic weight increases, the relevance of that candidate will also increase, thus personalizing the recommendations to something that is both

related to the current topic, and preferable to the current user.

This example shows how relationships between topics, users and documents are used to personalize recommendations based on what the user likes, as well as expand into areas that are related by traversing the graph. In the next section we discuss evaluations for both the user interface and the recommender algorithm.

Chapter 5

Evaluation

To put these ideas into practice, we conducted two preliminary evaluations: one for the user interface, and one for the recommender algorithm. To test the clarity of the user interface, we gave the prototype to students in two different computer science classes at Drew. We asked them to save resources they found helpful in learning course concepts. We then invited them to fill out a survey to provide feedback on their user experience using Knowd.

To evaluate the recommender algorithm, we used data from the question and answer site StackOverflow.com to try and predict terms that each user would ask or answer about. We found that our algorithm performed significantly better than using only term similarity or predicting random terms. The following sections describe these evaluations in more detail.

5.1 User Interface Feedback

5.1.1 Initially Populating Knowd Topics

To test this idea with real users, we needed to initialize the system with existing data. We used two main sources of data in addition to manually seeding the system. To build the ontology, we started by using data from the Wikipedia API. The API made it easy to grab page content as well as categorical relationships from Wikipedia. We also used data sourced from Stack Overflow to associate Tags with content from questions and answers.

Initially, we tried to automatically generate this topic hierarchy by using clustering algorithms, but found it difficult to find topics that make sense to human users. We used k-means clustering with cosine similarity between term vectors generated from Stack Overflow questions. The resulting document clusters were of mixed quality. Some groups had documents all about the same topic but others were not as coherent. However, the main problem with this was that there was not a good way to give the clusters a human readable name that represented that topic. We tried to take the term with the highest TF-IDF value, but again had mixed results.

Because of these issues, we took a user-centered approach that allows users to create their own topics and provide feedback to build the ontology. By relying partially on user feedback, the system can learn from what users find important. However, it does create an issue where the system is only as good as the people using it. We seeded the system with the results of manually crafted sample queries from the Bing Search API. Documents from these three sources were seen by users in the UI evaluation.

5.1.2 User Interface Survey

To evaluate the effectiveness of the user interface in communicating the problem, we tested Knowd with 22 students at Drew. The initial test was done in the Software Engineering class. In two class activities, students learned about two different technologies called Git and Wordpress. Git is a software system that developers use to manage versions and collaborate on source code. Wordpress is a blogging framework that is widely used to manage dynamic web content. We also ran the activity in an Introduction to Java course. Students in that class saved resources to help teach other students Object Oriented Design, which is a core concept in computer programming.

Students created accounts on Knowd and were asked to save any resources they found helpful during their learning. After the activities, participants were invited to complete a survey to provide feedback on their experience. Through these class activities and the resulting surveys, we got a sense of how students might use Knowd and how we could improve the prototype.

The survey was designed to gather a general sentiment towards the application as a whole including questions aimed at specific components of the application such as adding and pinning resources. A portion of the survey was posed as numerical response questions on a Likert scale between 1 and 7. In addition, participants were asked open ended questions designed to evaluate if they were part of Knowd's target user base as well as what they liked and disliked about the tool. The full survey can be found at `http://goo.gl/forms/Mjd4rOSvYl` and in Appendix A.

Figures 5.1.1 and 5.1.2 show results from the numerical survey questions. Figure 5.1.1 displays results measured on a 1–7 Likert scale, 7 being the most positive

answer. For the questions that ask how easy or complicated was an aspect of the system, a 1 represents very complicated and a 7 represents very easy. Shown in Figure 5.1.2, we also asked questions about the likelihood of different future actions like saving things to Knowd, searching on Knowd, or revisiting the resources saved in this activity. The box represents the middle 50% of the data, the heavy middle line represents the median, the small square represents the mean, and "+" indicates outliers.

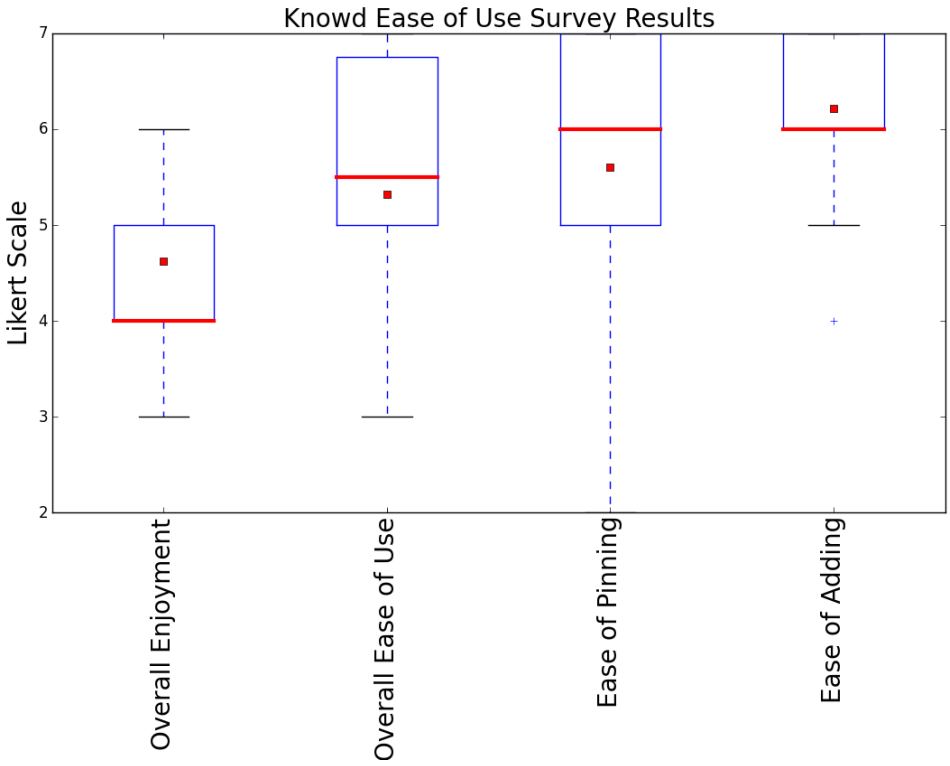

FIGURE 5.1.1: Knowd ease of use and enjoyment

These results show that the sentiment toward the concept of Knowd is generally good. Students found the pinning and saving functionalities easy to use, with mean responses of 5.6 and 6.2, respectively. In contrast, the means for overall enjoyment

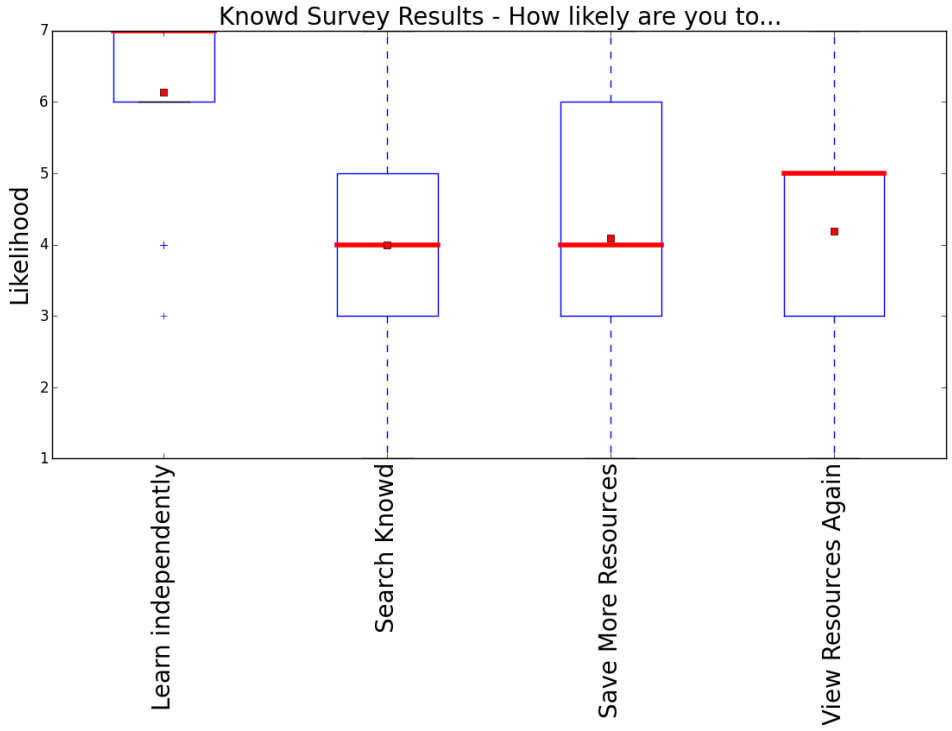

FIGURE 5.1.2: Likelihood learning independently and using Knowd to do so

and overall ease of use were 4.6 and 5.1. Because pinning and adding were described as easier than the overall tool, we can conclude that other areas of the interface were more difficult, such as navigating from page to page. Students also said they were ambivalent about using Knowd again, with a mean of 4.0 for the first three questions in Figure 5.1.2. Almost all the students said they were very likely to learn independently (mean of 6.1), which represents the target user audience for Knowd.

The users also gave qualitative responses through open-ended questions. Students use existing tools like Google and Stack Overflow to find resources, but can't always find what they are looking for. One student mentioned a tool specific to learning:

> "I use Team Treehouse and tutorials I find online for whatever I'm building. TeamTreehouse is good at taking me step by step, but it only teaches you to build a simple thing".

Team Treehouse [1] is a for-profit online school for computer science with video tutorials for various subjects. These courses, while helpful for starting out, suffer from the same problem as LMSs in college courses: students cannot contribute content, nor does the course adapt to students' previous knowledge. The survey responses have anecdotally confirmed the existence of this problem as well as shown that a tool such as Knowd might be able to provide a solution.

In addition to highlighting the problems with existing tools, the survey and activities provided feedback to inform future updates to Knowd's user experience (UX). The general sentiment about the UX was that it was easy to add and pin resources, but navigating the site was sometimes confusing. One way to alleviate this confusion might be to separate resource recommendations from navigating the topic graph. This might help communicate more clearly the relationships between topics and allow users to have more control over how much detail they see.

When asked about the quality of the recommendations, one student said, "They're good, but since there's so much writing it takes a while to read through them. They should be more simple and clear". While this survey was not meant to evaluate the recommender algorithm, it did reveal that users want the recommendations displayed in a cleaner interface with less text.

Another important and frequent piece of feedback was that students wanted a

[1] `http://teamtreehouse.com/`

page where they could all post resources as a group that were shared with the class. This highlights the overall goal of creating a learning management system that is collaborative and adaptive to student needs.

The main takeaway from this survey is that the concept of Knowd was well received as a possible solution, but the prototype has room for improvement.

5.2 Recommendation Algorithm Evaluation

To test the theory described in Chapter 4 with real data, we gathered a set of users and their associated posts from Stack Overflow. This dataset has 1,295,620 different users and 10,338,371 posts in total. We used only a small fraction of these data points in our evaluation.

5.2.1 Experimental Design

To get a document set, we took all the posts that were authored by any of the users in the sample. The posts can be either questions or answers. For each user, we split their posts into training and testing sets. We trained the system on a set of documents such that each user owned 10 documents. All the posts were used as a test set.

The vocabulary was initialized using the tags that the user assigned to each question, and expanded using terms from the bodies and titles of the posts. The final vocabulary was chosen by taking the top 500 terms with the highest *idf* value (i.e., the terms used in the fewest documents). The same vocabulary was used for both training and testing.

Once the vocabulary was selected, we learned the weights from the training set and generated topic recommendations based on the methods described in Chapter 4. To test the accuracy of these recommendations, we compared the set of recommended topics that were generated from the training set to the set of topics of interest as learned from the test set. We were essentially predicting the future interest, or user-topic weight W_{TU}.

We used a test set that included all posts from the training set plus additional posts authored by the users. Weights were learned using this expanded test set. We then compared the recommended topics with the actual topics that the user was interested in once the additional documents were included. To do this, we used a standard metric of success from information retrieval called average precision [10].

Our system returns a ranked list of terms that the user might be interested in. For each of these terms, we calculate precision, which is the ratio of correct predictions seen up to this point, over the total number of predictions. We then average over the precision values for correct predictions to get average precision. For example, say we recommend "HTML", "Algorithms", "Django", "CPU", "CSS", where "CPU" and "Algorithms" are incorrect. We get precision values of $\frac{1}{1}, 0, \frac{2}{3}, 0, \frac{3}{5}$ for each prediction, respectively. We then average these over only correct predictions. The final AP value for this user is 0.75. One way to interpret this is the frequency of correct answers. An AP value of 0.5 means every other prediction was correct, an AP of 0.33 means every third prediction was correct.

We are attempting to label users with multiple terms they might be interested in. Essentially the problem reduces to a multi-label classification problem, where

the order of the labels matters. This score measures how well our recommendations match the true sequence of labels. To determine the true labels for each user as a binary vector, we learned weights from the test set to see what users were interested in once more data was added. We then used the weight W_{TU} learned from the test set to get the true labels. If the weight W_{TU} is more than two standard deviations above the mean, we consider that a true label. In other words, if the weight to a particular term is higher than most of the other weights, that user is considered interested in the topic. We then use the average precision score to determine how many of these term labels the algorithm labelled correctly.

In the algorithm described in Algorithm 1, we rank the candidate topics by the relevance score defined in Equation 4.2.1, which depends on the similarity of the candidate topic to the current topic, W_{TT}, and the interest of the user in the candidate topic, W_{TU}. As a baseline we remove the W_{TU} parameter so that the relevance is based only on the similarity of the two topics, and not the user preferences. We also compare to a second baseline where we recommend random topics.

5.2.2 Results

We tested our algorithm with a random sample of 100 users on Stack Overflow who authored more than 20 posts. We ran our evaluation and measured the average precision of each user's term predictions.

Figure 5.2.1 shows the results of this evaluation in a box plot, where the y-axis measures average precision. The mean average precision (MAP) across all the users is shown by the small square in the middle of each box. Random predictions, yielded

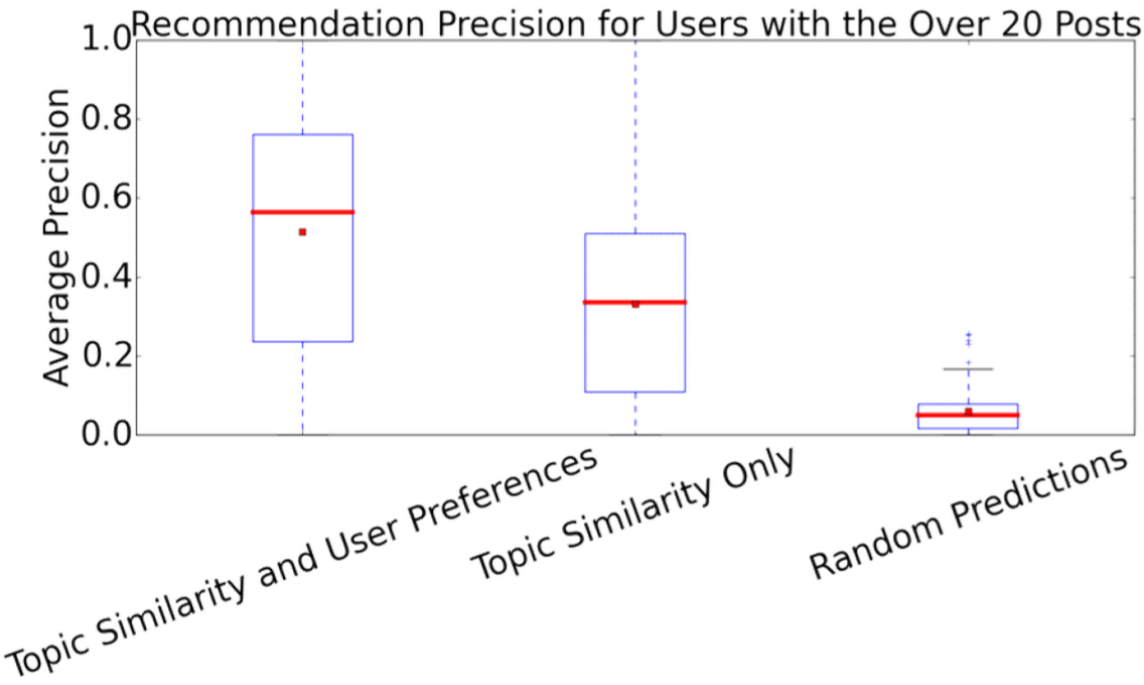

FIGURE 5.2.1: Evaluation Results with Random Sample

a mean of 0.06. Using only term similarity, the mean average precision goes to 0.33. Our algorithm achieved a mean average precision of 0.51. This means every other recommendation was something that the user showed interested in. An AP value of 1 is all correct answers, so with a value of 0.51 there is still room for improvement.

We used a paired sample t-test to determine if these differences were statistically significant, assuming a normal distribution due to the sample size of 100. This test shows that including user preferences in the relevance score performed significantly better ($p = 8.29 \times 10^{-15}$) than the topic similarity method alone and random recommendations ($p = 1.03 \times 10^{-23}$).

5.2.3 Threats to Validity and Next Steps

One threat to validity is that this dataset does not exactly match the format and use cases of Knowd. For example, Knowd allows many users to tag documents, but in Stack Overflow, documents can only be tagged by the author. We use these tag actions as feedback according to Equation 4.1.5, however, because only one user is tagging, this does not evaluate the effectiveness of aggregating multiple user opinions into the recommendations.

Also, because the only user activity we consider for this dataset is authoring a post, we could not use this dataset to test the document recommendations. We cannot attempt to predict documents that users might interact with because all the correct posts (ones authored by the user) do not yet exist in the training set. With actual user data from Knowd in the future, we can use interaction data such as bookmarking and viewing so that users can interact with existing documents.

A logical next step would be to compare this algorithm with standard recommendation algorithms such as collaborative filtering. In the future, we would like to find a better way to test the core features of Knowd such as aggregating human opinion by tagging, and recommending topics in context of what is currently being viewed.

Chapter 6

Conclusions and Future Work

In this thesis, we presented Knowd: an application that allows users to create, organize, and consume content related to their learning. We have designed, implemented and tested Knowd. In addition, we have presented a model to learn and recommend learning resources to users based on their collective and individual activity.

Knowd has laid the foundation necessary to evolve into an LMS that allows both expert-curated and crowdsourced courses built from freely available content. Existing learning systems only distribute content to a closed group of students about a closed group of manually created topics. In contrast, the end goal of Knowd is to present courses openly to the public about any topic. These courses could be manually created by experts or dynamically generated from many sources. For example, a professor can organize course content on Knowd and get recommendations for content from other sources to fill curriculum gaps or provide an alternative view on a topic. With

this system, students can learn from varying sources, get a personalized perspective of the topic and contribute their own resources to the course.

In the future, we would like to improve the user experience by making the process of saving and organizing resources more streamlined. Since the goal is for this system to become a full learning management system for independent learners, the process of creating courses needs to be clear and simple. Feedback received from students indicate that the UI has room for considerable improvement. In line with this goal, we also want to include the ability to add files and images.

6.1 Planned Updates

In response to the feedback received in the user evaluation, several updates to the application are planned. There are two main areas for improvement. The first is the ability to collaborate on a shared repository of resources. Students expressed interest in having a place not only to store resources for themselves, but also an easier way to share those resources with their classmates. Concrete features include direct recommendations to friends, creating and viewing different user-created "courses" on a given topic.

The second area is the communication of the data model. We would like to make it easier to understand the difference between topic to topic relationships and topic to resource relationships, and make it more clear to the user what is a topic and what is a resource. Exploring the topic graph without being bombarded with resource recommendations is important for students to quickly get an overview of their areas

of interest. Once they have an idea of what topics they should learn, they can explore them in more detail.

We expect to add these features and conduct another user study in the future to better gauge users' needs and expectations of this problem. In addition to improving the user interface, improvements will be made to the algorithm for learning topic relationships and personalizing recommendations.

6.2 Improving Knowledge Model

There are several possible directions to take this research. We want to make the ontology and recommendation system more focused on the overall goal of exploring knowledge. One way to move the implementation forward is to use graph centrality measures in the relevance rankings to suggest topics that help connect that user to more knowledge at a faster rate. The challenge will be balancing the tendency to make high level topic recommendations with the need for deep learning in one topic. Another area to explore is more explicitly measuring prerequisite topics and trying to predict the order in which students should learn. Using semantic relationships in the ontology, rather than just numeric weights, will help capture that information as well as other types of relationships. As the system matures, it would also make sense to migrate the data to use a graph database like neo4j.[1] Graph databases are designed for this kind of modeling, and will make storage and retrieval more efficient as well as allow for operations like finding the shortest path between two nodes.

[1] `http://neo4j.com/`

The research goal of this project is to find a way to design an application that is useful to learners and invites them to contribute data to help the system learn relationships between concepts. The main challenge in moving forward is to keep these two goals in sync with each other as new features are added. The overall goal of automatically generating personalized course content necessarily involves educators as well as students. To accomplish that goal, the knowledge graph must integrate data about both the learning process and the teaching process. Getting that data will require a focus on acquiring users and better evaluating their needs.

6.3 Limitations

A fundamental assumption we make is that current learning is dependent on past learning, and that past bookmarking activity is predictive of current knowledge. For example, suppose a user bookmarks many resources about "HTML" in the past. When they are attempting to learn "PHP" (a scripting language to embed dynamic content in HTML), Knowd interprets the past activity as current knowledge, and might adapt its recommendations to show how the new topic relates to the old topic. When learning about disparate concepts that are unrelated, this assumption may not hold, and a different approach might be needed.

One limitation of this approach is that when expanding knowledge, this system might shy away from some content in an attempt to relate something new to something you already know. For example, if a student is learning about iOS development, and has a high weight to a related topic "Front-End Web Development", that student

might get recommendations that combine the two topics such as building an iOS app with web technologies. However, the student might need to learn native iOS. In other words, recommendations from this approach have a bias towards new topics which build on prerequisite knowledge that the user already has.

We also run into a similar problem as collaborative filtering techniques. The ontology provides a starting point, but to get meaningful data on user preferences, we need users to interact with the content on the site.

6.4 Conclusion

In the process of self-education, there are limited tools available to organize learning materials. Without the guiding hand of a professor or curriculum, it is often difficult to get an overview of a topic. With Knowd, students can explore related concepts to get a sense of the learning environment for a particular topic. In addition, users can find new topics to learn about that are related to their interests and find resources that similar students have found helpful. We are excited to continue developing this idea into a product that students everywhere can use to facilitate their learning.

Appendix A. Full Survey Questions

1. How much did you enjoy using Knowd?
2. How easy or complicated was it to use Knowd?
3. How easy or complicated was it to use the pin functionality?
4. How easy or complicated was it to add a new resource?
5. How likely are you to return to these resources at a later time?
6. How likely are you to save resources to Knowd in the future?
7. How likely are you to search for resources on Knowd in the future?
8. How likely are you to learn independently outside of class work?
9. What tools do you currently use to find good learning resources or tutorials?
10. What do you wish you could change or improve about them?
11. What features do you like best about Knowd?
12. How do they compare to existing tools?
13. What would you change or improve about Knowd?
14. How do you see yourself using Knowd in the future, if at all?
15. What do you think about the quality of the recommendations?
16. Do you have any other comments or concerns about Knowd?

Appendix B. Variables used

T	set of all tags
U	set of all users
tf-idf	Term Frequency * Inverse Document Frequency
$freq(t_j, d_i)$	Term Frequency of term t_j in document d_i
$idf(t_j, D)$	Inverse Document Frequency of term t_j in document set D
d_i	one document
t_j	one term
D	set of all documents
n_j	number of documents in which term t_j appears
W_{TD}	matrix of weights between all terms and all documents
W_{TU}	matrix of weights between all terms and all users
u_k	one user
$U(t_j, d_i)$	set of all users who tagged document d_i as topic t_j
α_U	parameter defining the relative strength of user tags in Equation 4.1.5
$A(u_k)$	set of documents added by user u_k
$V(u_k)$	set of documents viewed by user u_k
α_A	parameter defining the relative strength of the documents the user added in Equation 4.1.5
α_V	parameter defining the relative strength of the user's viewing history in Equation 4.1.6
W_{TT}	matrix of weights (similarity) between all pairs of terms
θ_T	threshold weight for candidate topics
θ_D	threshold weight for candidate documents
$\beta_{diversity}$	parameter defining the balance between recommending documents related to the current topic vs documents related to related topics

Bibliography

[1] Robert M Bell and Yehuda Koren, *Lessons from the Netflix prize challenge*, ACM SIGKDD Explorations Newsletter **9** (2007), no. 2, 75–79.

[2] Robin Burke, *Hybrid web recommender systems*, The adaptive web, Springer, 2007, pp. 377–408.

[3] Cristian Cechinel, Miguel-ÁNgel Sicilia, Salvador SáNchez-Alonso, and Elena GarcíA-Barriocanal, *Evaluating collaborative filtering recommendations inside large learning object repositories*, Inf. Process. Manage. **49** (2013), no. 1, 34–50.

[4] Gayle Christensen, Andrew Steinmetz, Brandon Alcorn, Amy Bennett, Deirdre Woods, and Ezekiel J Emanuel, *The MOOC phenomenon: who takes massive open online courses and why?*, Available at SSRN 2350964 (2013).

[5] Dragan Gašević and Marek Hatala, *Searching context relevant learning resource using ontology mappings*, 3rd International Workshop on Applications of Semantic Web Technologies for E-Learning, Citeseer, 2005, pp. 45–52.

[6] Xing Jiang and Ah-Hwee Tan, *Learning and inferencing in user ontology for personalized semantic web search*, Information sciences **179** (2009), no. 16, 2794–2808.

[7] Raymond YK Lau, Dawei Song, Yuefeng Li, Terence CH Cheung, and Jin-Xing Hao, *Toward a fuzzy domain ontology extraction method for adaptive e-learning*, Knowledge and Data Engineering, IEEE Transactions on **21** (2009), no. 6, 800–813.

[8] Julie M Little-Wiles, Stephen Hundley, Wanda L Worley, and Erich J Bauer, *Faculty perceptions and use of a learning management system at an urban, research institution*, American Society for Engineering Education, American Society for Engineering Education, 2012.

[9] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro, *Content-based recommender systems: State of the art and trends*, Recommender systems handbook, Springer, 2011, pp. 73–105.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[11] Jonathan M Mortensen, *Crowdsourcing ontology verification*, The Semantic Web–ISWC 2013, Springer, 2013, pp. 448–455.

[12] G. Salton, A. Wong, and C. S. Yang, *A vector space model for automatic indexing*, Commun. ACM **18** (1975), no. 11, 613–620.

[13] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock, *Methods and metrics for cold-start recommendations*, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (New York, NY, USA), SIGIR '02, ACM, 2002, pp. 253–260.

[14] Sergey Sosnovsky, I-Han Hsiao, and Peter Brusilovsky, *Adaptation "in the wild": ontology-based personalization of open-corpus learning material*, 21st Century learning for 21st Century skills, Springer, 2012, pp. 425–431.

[15] Xiaoyuan Su and Taghi M. Khoshgoftaar, *A survey of collaborative filtering techniques*, Adv. in Artif. Intell. **2009** (2009), 4:2–4:2.

[16] William R Watson and Sunnie Lee Watson, *What are learning management systems, what are they not, and what should they become?*, TechTrends **51** (2007), no. 2, 29.

[17] Wilson Wong, Wei Liu, and Mohammed Bennamoun, *Ontology learning from text: A look back and into the future*, ACM Comput. Surv. **44** (2012), no. 4, 20:1–20:36.

[18] Yao-Tang Yu and Chien-Chang Hsu, *A structured ontology construction by using data clustering and pattern tree mining*, Machine Learning and Cybernetics (ICMLC), 2011 International Conference on, vol. 1, July 2011, pp. 45–50.

[19] Li Yuan, Stephen Powell, and JISC CETIS, *MOOCs and open education: Implications for higher education*, Cetis White Paper (2013).

[20] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang, *Solving the apparent diversity-accuracy dilemma of recommender systems*, Proceedings of the National Academy of Sciences **107** (2010), no. 10, 4511–4515.

[21] Leyla Zhuhadar, Olfa Nasraoui, Robert Wyatt, and Elizabeth Romero, *Multi-model ontology-based hybrid recommender system in e-learning domain*, Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03, IEEE Computer Society, 2009, pp. 91–95.

[22] Hanane Zitouni, Lamia Berkani, and Omar Nouali, *Recommendation of learning resources and users using an aggregation-based approach*, Advanced Information Systems for Enterprises (IWAISE), 2012 Second International Workshop on, IEEE, 2012, pp. 57–63.