Structured Generation of Technical Reading Lists

Jonathan Gordon
USC Information Sciences Institute
Marina del Rey, CA, USA
jgordon@isi.edu

Stephen Aguilar
USC Rossier School of Education
Los Angeles, CA, USA
aguilars@usc.edu

Emily Sheng
USC Information Sciences Institute
Marina del Rey, CA, USA
ewsheng@isi.edu

Gully Burns
USC Information Sciences Institute
Marina del Rey, CA, USA
burns@isi.edu

Abstract

Learners need to find suitable documents to read and prioritize them in an appropriate order. We present a method of automatically generating reading lists, selecting documents based on their pedagogical value to the learner and ordering them using the structure of concepts in the domain. Resulting reading lists related to computational linguistics were evaluated by advanced learners and judged to be near the quality of those generated by domain experts. We provide an open-source implementation of our method to enable future work on reading list generation.

1 Introduction

More scientific and technical literature is instantly accessible than ever before, but this means that it can also be harder than ever to determine what sequence of documents would be most helpful for a learner to read. Standard information retrieval tools, e.g., a search engine, will find documents that are highly relevant, but they will not return documents about concepts that must be learned first, and they will not identify which documents are appropriate for a particular user. Learners would greatly benefit from an automated approximation of the sort of personalized reading list an expert tutor would create for them. We have developed TechKnAcq – short for Technical Knowledge Acquisition – to automatically construct this kind of pedagogically useful reading list for technical subjects.

Presented with only a "core corpus" of technical material that represents the subject under study, without any additional semantic annotation, TechKnAcq generates a reading list in response to a simple query. For instance, given a corpus of documents related to natural language processing, a reading list can be generated for the query "machine translation." The reading list should be similar to what a PhD student might be given by her advisor: it should include prerequisite subjects that need to be understood before attempting to learn material about the query, and it should be tailored to the individual needs of the student.

To generate such a reading list, we first infer the conceptual structure of the domain from the core corpus. We then expand this corpus to include a greater amount of relevant, pedagogically useful documents, and we relate concepts to one another and to the individual documents in a *concept graph* structure. Using this graph and a model of the learner's expertise, we generate personalized reading lists for the user's queries. In the following sections, we describe these steps and then evaluate the resulting reading lists for several concepts in computational linguistics, compared to reading lists generated by domain experts.

2 Generating a Concept Graph

A *concept graph* (Gordon et al., 2016) is a model of a knowledge domain and related documents. To generate a concept graph, we start with a *core corpus*, consisting of technical documents, e.g., the archives of an academic journal. We identify technical phrases in the core corpus and use these to find additional, potentially pedagogically valuable documents, such as reference works or tutorials. For each document in the resulting *expanded corpus*, we infer a distribution over a set of pedagogical roles. We model the concepts in the domain using topic modeling techniques and apply information-theoretic measures to predict *concept dependency* (roughly, prerequisite) relations among them. Associating the documents of the expanded corpus with these concepts results in a rich graph representation that enables structured reading list generation.

2.1 Pedagogical Corpus Expansion

Most technical corpora are directed at experts, so they typically focus on presenting new methods and results. They often lack more introductory or instructional documents, and those covering fundamental concepts. Therefore, before generating a reading list, we want to automatically expand a core technical corpus to include relevant documents that are directed at learners at different levels.

Identifying terms Given a collection of documents, our first step is to identify a list of technical terms that can be used as queries. We adapt the lightweight, corpus-independent method presented by Jardine (2014):

1. Generate a list of n-grams that occur two or more times in the titles of papers in the corpus.
2. Filter unigrams that appear in a Scrabble dictionary (e.g., common nouns).
3. Filter n-grams that begin or end with stop words, such as conjunctions or prepositions. (Remove "part of" but not "part of speech".)
4. Filter any n-gram whose number of occurrences is within 25% of the occurrences of a subsuming n+1-gram. E.g., remove "statistical machine" because "statistical machine translation" is nearly as frequent.

Based on manual inspection of the results, we increased the threshold for subsumption to 30% and added two steps:

5. Filter regular plurals if the list includes the singular.
6. Order technical terms based on the density of the citation graph for documents containing them (Jo et al., 2007).

Jardine (2014) removes the bottom 75% of unigrams and bigrams by frequency (but keeps all longer n-grams). The Jo et al. (2007) method is better for comparing terms than simple frequency, but most technical terms we discover are also of high quality, making aggressive filtering of unigrams and bigrams unnecessary. Jardine also adds acronyms (uppercase words in mixed-case titles), regardless of frequency. We find acronyms from the initial collection of terms and do not consider it necessary to add singleton acronyms to our results – or those that are also a common noun, e.g., TRIPS, since we cannot assure case sensitivity in our searches.

Wikipedia and ScienceDirect We retrieve book chapters from Elsevier's ScienceDirect full-text document service and encyclopedia articles from Wikipedia. For Wikipedia, each term is queried individually, but only the top two results are included. For ScienceDirect, terms are used to retrieve batches of 50 results for each disjunction of 100 technical terms. This identifies documents that are central to the set of query terms rather than those with minimal shared content, and it reduces the number of API requests required. These documents are filtered based on heuristic relevance criteria: For Wikipedia, we keep documents if they contain at least 15 occurrences of at least five unique technical terms. For ScienceDirect, we require at least 20 occurrences of at least 10 unique technical terms since these documents tend to be longer.

Given this initial set of matching documents, we can then exploit their natural groupings: For Wikipedia, these are the categories that articles belong to, while for ScienceDirect, they are the books the chapters are from. For each grouping of the matched documents, ordered by size, we add the most relevant 75% of the documents that belong to the grouping and pass a weaker threshold of relevance to the query terms (four occurrences of two unique technical terms). This adds back in documents that would not pass the more stringent filters above but are likely to be relevant based on these groupings. These thresholds were manually tuned to balance the accuracy and coverage of expansion documents for these sources, but a full consideration of the parameter space is left for future work.

Tutorials Tutorials are often written by researchers for use within their own groups or for teaching a course and are then made available to the broader community online. For developing scientists in the field, these serve as valuable training resources, but they are not indexed or collected in any centralized way. Our approach for downloading tutorials from the Web is as follows:

1. Search Google or Bing for each of the top-200 technical terms and for randomized disjunctions of 10 technical terms for the full list.
2. Filter the results with the ".pdf" file extension and containing the phrase "this tutorial."
3. For each result found for more than one query, perform OCR and export the document.

2.2 Computing Pedagogical Roles

Given an expanded corpus of pedagogically diverse documents, we would like to infer a distribution for each document of how well it fulfills different ped-

agogical roles. Sheng et al. (2017) have created an annotated corpus and trained a classifier to predict these roles:

- **Survey**: A survey examines or compares across a broad concept.
- **Tutorial**: Tutorials describe a coherent process about how to use tools or understand a concept, and teach by example.
- **Resource**: Does this document describe the authors' implementation of a tool, corpus, or other resource that has been distributed?
- **Reference work**: Is this document a collection of authoritative facts intended for others to refer to? Reports of novel, experimental results are not considered authoritative facts.
- **Empirical results**: Does this document describe results of the authors' experiments?
- **Software manual**: Is this document a manual describing how to use different components of a piece of software?
- **Other**: This includes theoretical papers, papers that present a rebuttal for a claim, thought experiments, etc.

For the training corpus – a subset of the pedagogically expanded corpus – annotators were instructed to select all applicable pedagogical roles for each document. In the experiments we report, we use a combination of the predicted roles and manually set prior probabilities for the different document sources (e.g., an article from Wikipedia is most likely to be a *Reference work*).

2.3 Computing Concepts and Dependencies

To infer conceptual structure in a collection of documents, TechKnAcq must first identify the *concepts* that are important in the document domain. We model concepts as probability distributions over words or phrases, known as *topics* (Griffiths and Steyvers, 2004). Specifically, we use latent Dirichlet allocation (LDA) (Blei et al., 2003), implemented in MALLET (McCallum, 2002), to discover topics in the core corpus.[1]

Many relations can hold between concepts, but for reading list generation we are most interested in *concept dependency*, which holds whenever one concept would help you to understand another. This is strongest in the case of prerequisites (e.g., *First-order logic* is a prerequisite for understanding *Markov logic networks*). Gordon et al. (2016) propose and evaluate approaches to predict concept dependency relations between LDA topics, and we adopt the average of their best-performing methods:

Word-similarity method The strength of dependency between two topics is the Jaccard similarity coefficient $J(t_1, t_2) = \frac{t_1 \cap t_2}{t_1 \cup t_2}$, using the top 20 words in the associated topic distributions. A limitation of this method is that it is symmetric, while dependency relations can be asymmetric.

Cross-entropy method Topic t_1 depends on topic t_2 if the distribution (e.g., of top-k associated words) for t_1 is better approximated by that of t_2 than vice versa – for cross entropy H function, $H(t_1, t_2) > H(t_2, t_1)$ – and their joint entropy is lower than a chosen threshold, namely, the average joint entropy of topics known not to be dependent.

2.4 Concept Graphs

In a concept graph, concepts are nodes, which may be connected by weighted, directed edges for relations including concept dependency. These concepts have associated features, most importantly their distribution over words or phrases, which will be used to match learners' queries. Documents are also represented as nodes, which have as their features basic bibliographic information and their pedagogical role distributions. Documents are connected to concepts by weighted edges indicating their relevance.

A natural basis for identifying the most relevant documents for a concept is the distribution over topics that LDA produces for each document. However, high relevance of a topic to a document does not entail that the document is highly relevant to the topic. In particular, the LDA document–topic composition gives anomalous results for documents that are not well aligned with the topic model. Therefore, we also compute scores for a document's relevance to a topic based on the importance of each word in the document to the topic. For each document, we sum the weight of each word or phrase for the topic (i.e., the number of times LDA assigned the word to that topic in the entire corpus). This score is then normalized by dividing by the length of the document and then by the maximum score of any document for that topic. The algorithm is given in Figure 1. In the concept graph, we use the average of the original document–topic composition weight and this alternative measure.

[1] Concepts are not tied to standard topic modeling, e.g., they can also come from running Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007) using Wikipedia pages.

```
Input: topic model T, corpus C, document d
scores ← nested hash table
foreach topic t ∈ T do
    scores[t][d] ← 0
    max_score ← 0
    foreach document d ∈ C do
        foreach word w ∈ d do
            scores[t][d] ← scores[t][d] +
              topic_weight(w, t)
        scores[t][d] ← scores[t][d] / length(d)
        if scores[t][d] > max_score then
            max_score ← scores[t][d]
    foreach document d ∈ C do
        scores[t][d] ← scores[t][d] / max_score
return scores
```

Figure 1: Algorithm to score the relevance of documents to concepts.

3 Generating a Reading List

Given a concept graph linking each concept to the concepts it depends upon and to the documents that describe it, we generate a reading list by

1. computing the relevance of each concept to the user's query string,
2. performing a depth-first traversal of the dependencies, starting from the best match, and
3. selecting documents for each concept based on our model of the user's expertise and the documents' pedagogical roles.

Learner models The *learner model* gives the user's level of familiarity with each concept in the concept graph for the domain. By modeling the user's familiarity with concepts when we generate personalized reading lists, we can prefer introductory material for new concepts and more advanced documents for the user's areas of expertise, omitting them when they would be included only as dependencies for another concept. Such a model can be built from an initial questionnaire or inferred from other inputs, such as documents the user has marked as read. In the absence of a model of the specific user, we fall back to generic "beginner," "intermediate," and "advanced" preferences, where all concepts are assigned the same level of familiarity.

Concept relevance Given a query, we match concepts based on lexical overlap with their associated word distribution. For each concept with a match score over a threshold, if the learner model indicates that the user is a beginner at that concept, we traverse concept dependencies until the relevance score drops below a threshold. If concept d is a prerequisite of the matched topic m with weight $P(d, m)$, the relevance $R(d) = M(d) + M(m) \cdot P(d, m)$, where M is the function giving the lexical overlap strength.

Document selection When we include concept dependencies, we bookend their presentation on the reading list by presenting one or more introductory or overview documents, presenting documents about the dependencies, and then proceeding to more advanced documents about the original concept. So, for instance, a reading list might include an overview about *Markov logic networks*, then present documents about the prerequisite concepts *First-order logic* and *Markov network*, and end with more advanced documents about *Markov logic networks*. This avoids the confusion of presenting documents in strict concept dependency order, where the learner may not have the basic understanding of a subject to recognize why the prerequisites are in the reading list and how they relate to the query concept.

If the user already has advanced knowledge of a concept, we do not follow dependencies. Instead, we present three papers for that concept: a survey and two empirical results papers. We keep track of the concepts and documents that have been covered by the reading list generation so that, for instance, a matching topic that is also a dependency of a stronger match will be included as a dependency but not repeated later.

4 Evaluation

To enable comparison to an existing gold standard, we evaluated TechKnAcq on the domain of computational linguistics and natural language processing. Our evaluation covers 16 topics: For eight topics, we evaluate the expert-generated Jardine (2014) gold standard (JGS) reading lists and reading lists generated by TechKnAcq for the same topics. We additionally evaluated reading lists generated by TechKnAcq for eight topics of central importance in the domain, sampled from the list of "Major evaluations and tasks" on the Wikipedia article on natural language processing.[2] In this section, we describe the generation of a concept graph for the evaluation domain, the evaluation methodology and participants, and the results.

4.1 Evaluation Domain

As our core corpus, we used the ACL Anthology, which consists of PDFs – many of them scanned –

[2] https://en.wikipedia.org/wiki/Natural_language_processing#Major_evaluations_and_tasks

of conference and workshop papers and journal articles. There have been multiple attempts to produce machine-readable versions of the corpus, but all suffer from problems of text quality and extraction coverage. We used the December 2016 release of the ACL Anthology Network corpus (Radev et al., 2009), which includes papers published through 2014. We automatically and manually enhanced this corpus by adding missing text, removing documents not primarily written in English and ones with only abstracts, and joining words split across lines. After running the corpus expansion method described in Section 2.1, the corpus includes:

- 22,084 papers from the ACL Anthology
- 1,949 encyclopedia articles from Wikipedia
- 1,172 book chapters from ScienceDirect
- 114 tutorials retrieved from the Web

The concept graph was generated using a 300-topic LDA model, defined over bigrams. Names were manually assigned to 238 topics, and 62 topics that could not be assigned a name were excluded from the concept graph.

4.2 Evaluation Method

We recruited 33 NLP researchers to take part in the evaluation, primarily from an online mailing list for the computational linguistics community. Participants were required to have institutional affiliations and expertise in NLP. In the evaluation, participants were presented with the reading lists[3] and asked to change the order of documents to the order they would recommend a novice in NLP to read, i.e., ensuring that the first documents require limited knowledge and the documents that follow are predicated on the ones that came before. The participants could also remove documents from the reading list and suggest new documents be added in any position. By tracking changes in the reading lists, we can measure how many entries had to be changed for the list to be satisfactory.

Three sets of reading lists were evaluated. The first two were comparable lists, consisting of expert-generated lists, and their TechKnAcq counterparts. Together, these constitute the "comparison" set. The third set consisted of additional TechKnAcq-generated reading lists; this constitutes the "stand-alone" set. In addition to this edit-based evaluation, for the stand-alone set participants were asked to rate their agreement with statements about reading lists generated by TechKnAcq for a qualitative measure of a reading list's pedagogical value.

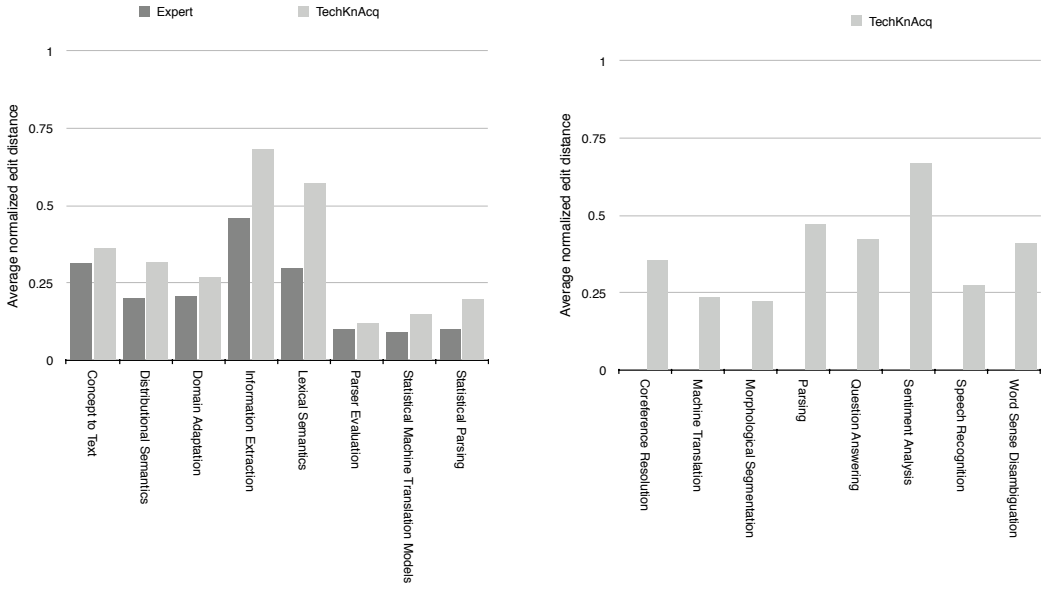
[3] The order in which TechKnAcq and JGS reading lists were presented was randomized and counterbalanced to control for order effects.

4.3 Evaluation Results

The similarity of TechKnAcq reading lists to expert-generated ones in terms of pedagogical value was assessed based on the changes participants made to the lists – the fewer documents that were moved, deleted, or added, the better the participant considered the reading list. The total number of changes to a reading list was measured using edit distance, but we are also interested specifically in the stability of document positions, the number of documents deleted, and the number of documents added to the reading lists.

Edit distance One of the most natural ways to compute how much a participant modified a given reading list overall is to use Levenshtein (1966) edit distance. This is a method of computing the fewest edit operations necessary to turn one sequence into another, classically applied to spell-checking. The operations are insertion, deletion, and substitution of an item. So, for instance, if the participant removes a paper and adds another in the same location in the reading list, she has performed a substitution, with an edit distance of one. If she then moves a paper from the end of the reading list to the beginning, that is a deletion from the old location followed by an insertion. A limitation of edit distance is that it does not take into account the length of the sequence being modified. E.g., a long reading list that is mostly considered to be good may have the same number of edits as a shorter reading list that is much worse. As such, we also normalized the edit distance scores by dividing by the length of the original reading list. For the comparable set, the average edit distance was 0.22 for an expert reading list and 0.33 for a TechKnAcq-generated one. The edit distance for TechKnAcq reading lists for the stand-alone set was 0.38. These results are shown in Figure 2.

List stability One indicator of reading list quality is how stable a list is, i.e., whether a document changes position within a list. This is computed as the number of documents whose absolute position in the reading list has changed, not including documents that were added (written in) by the participants. The mean level of stability for reading lists is given in Table 1. Smaller means, paired with smaller standard deviations indicate more stability within the reading list for a query. Minimums and



(a) Comparison set

(b) Stand-alone set

Figure 2: Average Levenshtein edit distances for reading lists produced by domain experts and by TechKnAcq, normalized by dividing by the original length of each reading list.

	TechKnAcq-generated reading lists						*Expert-generated reading lists*					
Domain	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*
Concept to Text	0.63	3.75	2.01	0	6	6	0.80	12.83	4.26	0	16	16
Distributional Semantics	0.66	4.62	2.93	0	7	7	0.71	10.00	4.73	0	14	14
Domain Adaptation	0.69	4.13	1.55	2	6	6	0.80	8.78	1.64	6	10	11
Information Extraction	0.64	7.04	3.25	0	10	11	0.65	5.85	3.72	0	9	9
Lexical Semantics	0.60	5.95	3.95	0	10	10	0.56	7.90	5.04	0	14	14
Parser Evaluation	0.77	10.00	1.41	9	12	13	0.75	3.00	1.41	1	4	4
Stat. Machine Trans. Models	0.84	15.88	3.40	10	19	19	0.55	2.75	2.05	0	5	5
Statistical Parsing	0.79	10.29	4.64	0	13	13	0.66	14.57	10.03	0	22	22
Average	0.70						0.69					
Coreference Resolution	0.60	3.58	1.98	0	6	6						
Machine Translation	0.55	8.25	4.74	0	13	15						
Morphological Segmentation	0.67	6.00	2.78	0	9	9						
Parsing	0.54	5.40	4.81	0	10	10						
Question Answering	0.56	3.36	2.17	0	6	6						
Sentiment Analysis	0.68	4.05	1.86	0	6	6						
Speech Recognition	0.73	8.00	4.18	0	11	11						
Word Sense Disambiguation	0.64	3.81	2.04	0	6	6						
Average	0.62											

Table 1: Changes to document positions in expert and TechKnAcq reading lists, for the comparison and stand-alone sets. Lower numbers indicate greater list stability. *Norm* is the mean number of changes normalized by dividing by the reading list length to allow comparison across lists.

maximums are also reported, with TechKnAcq scoring a minimum of zero more often, indicating that participants left these lists unchanged more often than the expert (JGS) lists. Note that, unlike for edit distance, some changes to reading lists, such as moving the first document to the end, have an outsize effect on the stability score compared with others, like swapping the first and last documents. This indicator is also sensitive to list length – the longer the list, the more potential there is for changes within the list. For the comparison set, the average stability for TechKnAcq reading lists, normalized by length, is 0.70 vs 0.69 for expert-generated reading lists, indicating a similar level of document movement.

Deletions Fewer deletions signals a judgment that the reading list contents are appropriate. Table 2 presents the mean number of deletions. When deletions are normalized by reading list length, there are fewer (0.16) for expert-generated reading lists than for for TechKnAcq (0.23) on the comparison set. While the stability scores were similar for the comparison set, the deletions suggest that TechKnAcq does worse at selecting documents than experts do. This may be a limitation of computing relevance using a coarse-grained topic model or it may reflect that TechKnAcq includes more documents for concept dependencies than the participants felt necessary.

Additions Participants were encouraged to add any documents they felt belonged in the reading list that were not present. However, this was relatively labor-intensive, requiring the participant to either remember or look up relevant papers and then enter information about them. As such, relatively few documents were added. Statistics for additions are given in Table 3, but the rate with which documents were added is similar for TechKnAcq and expert-generated reading lists.

Qualitative For reading lists generated for the stand-alone set, participants qualitatively evaluated whether they were appropriate to use in a pedagogical setting. They were asked to rate their agreement with these statements on a scale from 1 (strongly disagree) to 7 (strongly agree):

1. This reading list is complete.
2. This is a good reading list for a PhD student.
3. I would use this reading list in one of my classes.
4. I would send this reading list to a colleague of mine.
5. This is a good reading list for a master's student.
6. I could come up with a more complete reading list than the one provided.
7. If a PhD read the articles in this reading list in order, they would master the concepts.

Cronbach's α was calculated for each set of questions; high values ($\alpha > .8$) indicate that each set of items were internally consistent, and closely related as a set (Santos, 1999). Thus, we averaged these ratings (with responses to Statement 6 inverted) for a composite measure of the pedagogical value of each reading list. Results indicate that, on average, the reading lists have moderate-to-high potential. These results are in Table 4.

5 Related Work

Research on information retrieval provides a historically sizable literature describing methods to catalog, index, and query document collections, but it focuses on the task of finding the most relevant documents for a given query (Witten et al., 1999). Wang et al. (2007) build a repository of learning objects characterized by metadata and then personalize recommendations based on a user's preferences. Tang (2008) introduces the problem of reading list generation and addresses it using collaborative filtering techniques. Ekstrand et al. (2010) provide a good run-through of possible competition based on collaborative filtering.

The doctoral work of Jardine (2014) addresses the question of building reading lists over corpora of technical papers. Given an input word, phrase, or entire document, Jardine identifies a weighted set of relevant topics using an LDA model trained on a corpus and then selects the most relevant papers for each topic using his ThemedPageRank metric. This is an unstructured method for reading list generation, while TechKnAcq uses concept dependency relations to order the presentation of topics. Jardine's method selects documents based on their importance to a topic but without consideration of the pedagogical roles the documents serve for different learner models.

Jardine's work provides a set of expert-generated gold-standard reading lists, which we have reused in our evaluation. Jardine asked experts to compose gold standard reading lists and compared these to the reading lists generated by his system, using a citation substitution coefficient to judge how similar a paper in his output is to that chosen by an expert.

	TechKnAcq-generated reading lists						*Expert-generated reading lists*					
Domain	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*
Concept to Text	0.17	1.00	1.28	0	4	6	0.31	4.92	4.27	0	14	16
Distributional Semantics	0.73	5.08	1.04	3	7	7	0.09	1.23	1.59	0	4	14
Domain Adaptation	0.11	0.78	1.09	0	3	7	0.19	2.11	1.05	0	4	11
Information Extraction	0.12	1.37	1.71	0	5	11	0.50	4.48	1.19	2	7	9
Lexical Semantics	0.26	2.55	3.43	0	10	10	0.07	1.00	1.62	0	5	14
Parser Evaluation	0.08	1.00	1.41	0	3	13	0.00	0.00	0.00	0	0	4
Stat. Machine Trans. Models	0.18	3.50	2.33	1	7	19	0.05	0.25	0.71	0	2	5
Statistical Parsing	0.19	2.43	1.81	0	4	13	0.11	2.43	2.37	0	7	22
Average	0.23						0.16					
Coreference Resolution	0.05	0.27	0.47	0	1	6						
Machine Translation	0.08	1.13	2.80	0	8	15						
Morphological Segmentation	0.23	2.11	2.32	0	6	9						
Parsing	0.08	0.83	1.60	0	4	10						
Question Answering	0.06	0.36	0.63	0	2	6						
Sentiment Analysis	0.07	0.41	0.80	0	3	6						
Speech Recognition	0.18	2.00	1.80	0	5	11						
Word Sense Disambiguation	0.07	0.44	0.89	0	3	6						
Average	0.10											

Table 2: Number of documents participants deleted from expert and TechKnAcq reading lists, for the comparison and stand-alone sets. Lower numbers indicate better document selection. *Norm* is the mean number of deletions normalized by dividing by the reading list length to allow comparison across lists.

	TechKnAcq-generated reading lists						*Expert-generated reading lists*					
Domain	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*	*Norm*	*Mean*	*SD*	*Min*	*Max*	*Len.*
Concept to Text	0.00	0.00	0.00	0	0	6	0.00	0.00	0.00	0	0	16
Distributional Semantics	0.11	0.77	1.17	0	3	7	0.04	0.54	0.78	0	2	14
Domain Adaptation	0.06	0.44	1.01	0	3	7	0.04	0.44	1.01	0	3	11
Information Extraction	0.04	0.48	0.98	0	4	11	0.02	0.19	0.48	0	2	9
Lexical Semantics	0.09	0.85	1.69	0	5	10	0.00	0.00	0.00	0	0	14
Parser Evaluation	0.08	1.00	1.41	0	3	13	0.13	0.50	0.58	0	1	4
Stat. Machine Trans. Models	0.07	1.38	1.41	0	4	19	0.08	0.38	0.74	0	2	5
Statistical Parsing	0.07	0.86	1.46	0	3	13	0.02	0.43	1.13	0	3	22
Average	0.06						0.04					
Coreference Resolution	0.10	0.58	1.24	0	4	6						
Machine Translation	0.01	0.13	0.35	0	1	15						
Morphological Segmentation	0.06	0.56	1.01	0	3	9						
Parsing	0.04	0.40	0.97	0	3	10						
Question Answering	0.02	0.14	0.53	0	2	6						
Sentiment Analysis	0.08	0.48	0.93	0	3	6						
Speech Recognition	0.05	0.56	1.33	0	4	11						
Word Sense Disambiguation	0.05	0.31	0.70	0	2	6						
Average	0.05											

Table 3: Number of documents participants added to expert and TechKnAcq reading lists, for the comparison and stand-alone sets. Lower numbers indicate better original reading lists. *Norm* is the mean number of additions normalized by dividing by the reading list length to allow comparison across lists.

	N	*Mean*	*SD*	*Min*	*Max*	α
Coreference Resolution	12	4.22	1.18	1.33	5.50	0.89
Machine Translation	8	4.04	1.95	1.00	5.83	0.97
Morphological Segmentation	9	3.41	1.73	1.00	5.50	0.96
Parsing	10	4.32	1.63	1.17	6.83	0.97
Question Answering	14	3.80	1.51	1.33	5.50	0.93
Sentiment Analysis	22	4.18	1.25	1.00	6.67	0.93
Speech Recognition	9	4.41	1.35	2.00	6.83	0.91
Word Sense Disambiguation	16	4.40	1.23	1.17	6.50	0.92

Table 4: Descriptive statistics for the pedagogical value of each TechKnAcq reading list, with 1 = weak pedagogical potential and 7 = strong pedagogical potential. *N* is the number of participants who rated the reading list for each query.

He also performed user satisfaction evaluations, where thousands of users of the Qiqqa document management system evaluated the quality of the technical terms and documents generated from their libraries.

In Section 2.1, we use a variant of Jardine's method for identifying technical terms in a set of documents, in order to run queries for expanding a core technical corpus to include more pedagogically helpful documents. There is significant prior work on identifying key phrases or technical terminology, e.g., Justeson and Katz (1995). We could also select phrases based on TF–IDF weighting of n-grams or using the highest weighted phrases in the LDA topic model. However, since the technical terms are only used to find additional documents, whose relevance is then determined by the LDA topic model and the document–topic relevance algorithm (Figure 1), the accuracy of technical term identification is not critical to our results. As this was not a focus of our research, Jardine's method was chosen largely for its simplicity.

6 Conclusions

We have presented the first system for generating reading lists based on inferred domain structure and models of learners. Our method builds a topic-based index for a technical corpus, expands that corpus with relevant pedagogically oriented documents, provides a preliminary encoding of the pedagogical roles played by individual documents, and builds a personalized, structured reading list for use by learners.

We predict that the greatest performance gains to be generated in future work are likely to come from more detailed and complete studies of the pedagogical value of specific documents (and types of documents) for individual learners. Thus, an important direction for future investigation may be to characterize a learner's knowledge in order to be able to score the pedagogical value of reading material for that person rather than for the generic learner models used in our evaluation.

We have demonstrated that the quality of reading lists generated in this way may be quantitatively compared to existing expert-generated lists and that our system approaches the performance of human experts. We are releasing our implementation[4] to support future efforts and serve as a basis for comparison.

Acknowledgments

The authors thank Yigal Arens, Aram Galstyan, Vishnu Karthik, Prem Natarajan, and Linhong Zhu for their contributions and feedback on this work.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Air Force Research Laboratory (AFRL). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

David M. Blei, Andew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Michael D. Ekstrand, Praveen Kannan, James A. Stemper, John T. Butler, Joseph A. Konstan, and John T.

[4] http://techknacq.isi.edu

Riedl. 2010. Automatically building research reading lists. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 159–66.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–11, San Francisco, CA, USA. Morgan Kaufmann.

Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. 2016. Modeling concept dependencies in a scientific corpus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (*ACL 2016*), pages 866–75.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the USA*, volume 101, pages 5228–35. Supplement 1.

James G. Jardine. 2014. Automatically generating reading lists. Technical Report UCAM-CL-TR-848, University of Cambridge Computer Laboratory.

Yookyung Jo, Carl Lagoze, and C. Lee Giles. 2007. Detecting research topics via the correlation between graphs and texts. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 370–9, New York, NY, USA. ACM.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. 1:9–27.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–10.

Andrew McCallum. 2002. MALLET: A machine learning for language toolkit.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 54–61.

J. Reynaldo A. Santos. 1999. Cronbach's alpha: A tool for assessing the reliability of scales. *Journal of Extension*, 37:1–5.

Emily Sheng, Prem Natarajan, Jonathan Gordon, and Gully Burns. 2017. An investigation into the pedagogical features of documents. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

Tiffany Ya Tang. 2008. *The Design and Study of Pedagogical Paper Recommendation*. Ph.D. thesis, University of Saskatchewan.

Tzone I. Wang, Kun Hua Tsai, Ming Che Lee, and Ti Kai Chiu. 2007. Personalized learning objects recommendation based on the semantic-aware discovery and the learner preference pattern. *Educational Technology & Society*, 10:84–105.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann.