About *SIGKDD Explorations*

Explorations is published twice yearly, in June/July and December/January each year. After the first two volumes, frequency may increase to quarterly. The newsletter is distributed in hardcopy form to all members of the ACM SIGKDD. It is also sent to ACM's network of libraries. Additionally, issues are published on the web and are free to the general public (http://www.acm.org/sigkdd/explorations/).

Our goal is to make *SIGKDD Explorations* an informative, rapid means of publication and a dynamic forum for communication with the Knowledge Discovery and Data Mining community. SIGKDD membership is growing at a very fast pace, and with KDD being a multi-disciplinary field, we hope that *Explorations* will facilitate its fusion and enhance the sense of community. Submissions will be reviewed by the editor and/or associate and guest editors as appropriate. We are particularly interested in short research and survey articles on various aspects of data mining and KDD. *Explorations* is also a forum for publishing position papers, controversial positions, challenges to the community, product reviews, book reviews, news items and other items of interest to the field. Please see:

http://www.acm.org/sigkdd/explorations/instructions.htm

Advertiser Information:

Explorations accepts advertisements related to data mining and KDD, including company, book, vendor, and service advertisements. For rates and instructions on submitting an ad, please see:

http://www.acm.org/sigkdd/explorations/instructions.htm#advertise

Notice to Contributing Authors of SIGKDD Explorations:

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library
- to allow users to copy and distribute the article for noncommercial, educational or research purposes.

However, as a contributing author, you retain the copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Notice to Past Authors of ACM-Published Articles:

ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written work that was previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform you respective editors and permissions@acm.org, stating the title of the work, the author(s), and where and when published.

On the Measurement and Prediction of Web Content Utility: A Review

Yuan Yao
Department of Computer Science and Technology
Nanjing University, China
y.yao@nju.edu.cn

Hanghang Tong
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, USA
hanghang.tong@asu.edu

Feng Xu, Jian Lu
Department of Computer Science and Technology
Nanjing University, China
{xf,lj}@nju.edu.cn

ABSTRACT

Nowadays, various types and large amount of content are available on the Web. Characterizing the Web content and predicting its inherent usefulness become important problems that may benefit many applications such as information filtering and content recommendation. In this article, we present a brief review of the existing measurements and the corresponding prediction methods for Web content utility. Specially, we focus on three close and widely studied tasks, i.e., content popularity prediction, content quality prediction, and scientific article impact prediction. While reviewing the existing work in each of the above three tasks, we mainly aim to answer the following two fundamental questions: how to measure the Web content utility, and how to make the predictions under the measurement. We find that while the three tasks are closely related, they bear subtle differences in terms of prediction urgency, feature extraction, and algorithm design. After that, we discuss some future directions in measuring and predicting Web content utility.

1. INTRODUCTION

In recent years, measuring and predicting Web content utility (e.g., popularity, quality, impact, etc.) have attracted much attention and have potential usefulness in many applications. For example, by measuring and predicting the popularity of Web content, we can design more profitable advertising strategies over the social media; by measuring and predicting the quality of Web content, we can more easily identify informative messages from the huge amount of noisy information; by measuring and predicting the future impact of research articles, we can better envision the research trends and plan our research roadmaps.

In this article, we briefly review the existing measurements and the corresponding prediction methods for Web content utility. Specially, we focus on the following three widely studied tasks.

- *Content popularity prediction.* Content popularity prediction is based on the current state of the content, and the goal is to predict its future popularity after a relatively long period. Popularity metrics (e.g., the number of views) are usually available in the social platforms that host the content. For this task, we summarize the existing methods on general Web content including text, videos, and images.
- *Content quality prediction.* The input of content quality prediction is similar to that of the content popularity prediction. For the output, content quality itself is a vague concept and different people may have different definitions on it over different social platforms. Therefore, most existing work either adopts human labeling or uses approximate metrics to define quality. Another difference from content popularity prediction (which estimates the future popularity) is that content quality prediction estimates the current quality of the content. In this article, we mainly summarize several mainstream content quality prediction problems including the helpfulness of product reviews, the credibility of microblog messages, and the quality of question/answers in question answering sites.
- *Scientific article impact prediction.* In addition the above to two general tasks, we review the impact prediction task for a specific type of content, i.e., the scientific articles. Scientific impact is of special interest to researchers. Similar to content popularity prediction, the impact prediction of scientific articles also aims to predict the future impact based on the current state, although scientific articles usually have a longer lifecycle than the other types of Web content.

While reviewing the existing work in each of the above three tasks, we mainly aim at answering two fundamental questions: how to measure the Web content utility, and how to make the predictions under the measurement. For the prediction of Web content utility, we focus on the following three aspects.

- *Prediction time.* The first aspect is about when to make the prediction. Based on the prediction time, existing work can be divided into three classes, i.e., before publication, at publication, and after publication. Before publication means that the prediction is made even before the content has been created and published; at prediction means that the prediction is made at the moment when the content is published; after publication means that the prediction is made after a short period of the publication time.
- *Features.* Depending on the prediction time, several types of features/factors may be indicative for the prediction task. For example, when the prediction time

is before publication, the content creator is the main factor; when the prediction time is at publication, the content itself is an indicative factor; when the prediction time is after publication, more context factors may play important roles.

- *Algorithms.* The third aspect is about the detailed algorithms used to finish the prediction. Many off-the-shell data mining algorithms have been used for the prediction tasks. Additionally, some researchers propose designed algorithms to better leverage the characteristics of the underlying social platforms.

The rest of this article is organized as follows. Section 2, Section 3, and Section 4 present the brief review of content popularity prediction, content quality prediction, and scientific article impact prediction, respectively. Section 5 summarizes the main findings and insights, and discusses some future directions in measuring and predicting Web content utility. Section 6 concludes the article.

2. CONTENT POPULARITY PREDICTION

In this section, we briefly review the existing literature for the popularity measurement and prediction of Web content.

2.1 Content Popularity Measurement

For content popularity, the measurement is relatively agreed. Most literature uses the view count (i.e., the number of views by the community users on the content) to measure popularity [74]. Since the view count may exhibit a power-law distribution in many social platforms, it is usually normalized by existing methods (e.g., [43; 19]). Compared to predicting view count, some researchers argue that identifying highly popular content is more meaningful. They find that a small amount of highly popular content usually dominates the major popularity views, and thus formulate a classification problem to identify such content (e.g., [45; 26]).

In addition to view count, some other metrics such as the number of votes/comments are also used to measure popularity with the assumption that the view count information is not always available. In this article, we treat popularity as view count as it is largely agreed by existing literature, and leave the discussions about other metrics in Section 3.

2.2 Content Popularity Prediction

Popularity prediction has been studied for many different types of Web content including videos [26], news [47], microblogs [39], reviews [79], images [19], codes [84], mobile apps [96], etc. Based on literature, popularity prediction is a challenging task due to the following two reasons. First, many factors are intuitively known to influence content popularity, and some of these factors are difficult to quantify (such as the quality of the content, the complex interactions between users and content, etc.). Second, content popularity can be affected by many external factors such as a shocking event in the physical world.

Next, we summarize the existing efforts for content popularity prediction in terms of prediction time, factors/features, and algorithms.

2.2.1 *Prediction Time*

For the prediction time, most existing work makes the prediction after a period of the publication time. The basic assumption is that the early popularity is a strong indicator for the future popularity. For example, an early practice finds a strong correlation between the logarithmically transformed past popularity and current popularity, and proposes to use the number of views that the Web content receives early after publication to predict its future popularity [73]. Within these after publication methods, some of them require fast predictions as users' interest may quickly fade (e.g., news), and some may afford several days to make predictions (e.g., videos).

In addition to most existing methods that make the prediction after publication, some methods focus on a harder problem to predict the popularity at publication time [6]. For example, Khosla et al. [43] use social features (e.g., the number of friends) and content features both of which are available at publication time to predict the popularity of images.

An even harder problem is to predict the popularity before publication, i.e., when the content is not published yet. This problem is seldom studied by existing researchers due to the fact that complex factors (e.g., content features) may affect the content popularity.

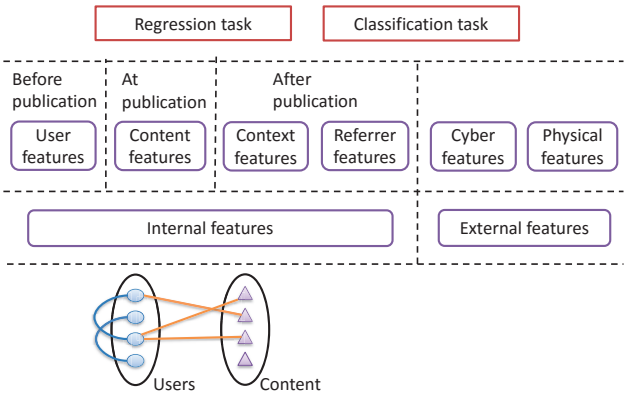


Figure 1: The features for content popularity prediction.

2.2.2 *Features*

The commonly used features for content popularity prediction are summarized in Fig. 1. Nowadays, Web content is typically published in social platforms where users can interact with the content and with each other. As shown in Fig. 1, the popularity of such content can be affected by both **internal features** and **external features**. Here, internal features indicate those features that can be extracted from the social platform where the content is published. These features are based on the interactions between users and content as well as the interactions between users. As shown in Fig. 1, we summarize the following four types of internal features.

- **User features**. User features indicate the impact of the content creator. Node-based user features (e.g., the account age) are relatively less visited as they are intuitively less indicative. In contrast, link-based user features (a.k.a. social features) have been widely explored [26; 27; 43]. Such features include the number of friends/followers/subscribers, social influence, etc.

- **Content features**. Content features are directly extracted from the content, and they are content-specific.

For example, language models can be used for text, representations from deep neural networks can be used for images [43], and abstract syntax tree features can be used for codes [84]. Some content features are easy to extract (e.g., the related tags, the time when the content is published, the location relevance in the content [10]), while others need careful design. For example, existing research has shown that emotion is one of the most important drivers for online audience [7; 33]; however, analyzing the emotion in Web content is a non-trivial task. Other non-trivial content features include quality, structure, readability, etc.

- **Context features**. Context features refer to the community users' early responses to the content after it is published. The early popularity is a typical context feature. In fact, Borghol et al. [9] have studied the popularity differences among videos that have essentially the same content, and find that the early popularity is one of the most indicative factor for videos. Actually, as we will later show in the next subsection, many existing methods use only this type of features to predict content popularity. Other used context features include the number of views on each monitored day, the popularity evolution patterns, the number of comments, the number of favorites, etc.

- **Referrer features**. Referrer features are related to the internal recommender systems, the internal search engine, or the internal social sharing tools of the given social platform. For example, the popularity can be dramatically increased if the content is recommended in the front page [27].

As to external features, they are extracted from outside the content's social platform. We divide these features into two classes.

- **Cyber features**. Cyber features are from cyber world, such as the number of incoming links from other social platforms, the recommendation from general search engines, etc. Castillo et al. [12] have shown that the features about a news article on social networks (e.g., the number of Facebook shares) are effective in predicting the popularity of the same article on a news site.

- **Physical features**. Physical features are real-world features with the assumption that the popularity of some Web content is strongly correlated with real-world events. However, it is challenging to transfer information from the physical world to the cyber world. An existing attempt is by Tsagkias et al. [77] who use the weather conditions to predict the number of comments for news articles.

For the above features, user features can be extracted before publication, content features can be extracted at publication time, and context features and referrer features can be extracted after publication. For external features (cyber features and physical features), they could be extracted at any time point. For example, a news article reporting a hot event may be popular, and a hot event may also make an old content popular.

2.2.3 *Algorithms*

Depending on the target measurements, many off-the-shell regression (e.g., SVR, linear regression, etc.) and classification (e.g., SVM, random forest, etc.) methods have been used for popularity prediction. Some other algorithms such as the stochastic process [38; 47], the extremely randomized ensemble trees [31; 26], and the two-step methods [4; 92] are also considered by existing literature. Overall, existing algorithms for content popularity prediction are not designed to fit the underlying social platforms.

2.3 Representatives

Here, we describe some representative popularity prediction methods.

As a pioneering practice, Szabo and Huberman [73] find a strong correlation between the logarithmically transformed past popularity and current popularity, and propose to use only the popularity value at prediction time as input. Following this work, Pinto et al. [59] find that different videos (and Web content in general) may have different temporal patterns in terms of popularity evolution, and they propose to use the view count in each monitored day before prediction time as features. Further, Ahmed et al. [4] propose to identify the temporal patterns of popularity evolution, and use these patterns to predict the future popularity. Specially, they take a two-step method, where the first step is to cluster the patterns based on the similarities of features, and the second step is to model the temporal evolution of popularity over the clusters. Finally, Yu et al. [92] define the concept of phase as a continuous time period in which a video's popularity has a salient rising or falling trend. Adding these phase features into the method by Pinto et al. [59], they achieve further accuracy improvement.

The above four pieces of work use only the context features (more specifically, the view count from publication time to prediction time) as input, and they all take a regression setting to predict the future view count of the content. However, as indicated by Castillo et al. [12], social features and other context features received in the first ten minutes after publication can achieve the same performance with the view count in the first three hours. Therefore, various other types of features that are useful for fast and accurate popularity prediction are considered by existing work. For example, Figueiredo [26] considers social features and content features in addition to context features, and formulates a classification problem to predict different types of popularity trends. Similarly, Vasconcelos et al. [79] consider social features, content features, and context features to predict the popularity levels[1] of micro-reviews (i.e., Foursquare tips); they formulate both regression and classification problems.

Speaking of social features, existing methods have some interesting findings that reveal their importance. For example, Khosla et al. [43] use both social features and image content features to predict the normalized view count. One of their findings is that popularity is difficult to precisely predict based on image content alone, and social features have a great influence on the popularity of images. Their methods do not use context features and referrer features, and thus can be used at publication time.

Intuitively, there is a tradeoff between the prediction time

[1]This work actually predicts the number of likes of Foursquare tips.

and prediction accuracy. Recently, Figueiredo et al. [28] explicitly address this tradeoff, and argue that accurate predictions should be made as early as possible (e.g., before users' interest has exhausted). They take a two-step method of trend extraction and trend prediction, and formulate multi-class classification problem for popularity trend prediction.

2.4 Summary

Content popularity may play important roles in many applications such as media advertising, content caching, and traffic management. Therefore, understanding what makes a piece of content popular and being able to predict its future popularity have attracted a lot of research interest in the past few years. Typically, the view count is used as the popularity measurement, and the main focus of existing work is on the feature extraction aspect. Overall, we summarize the existing features into internal features and external features, where internal features are more commonly used. We further divide internal features into four types, in which user features, content features, and context features are mostly used.

Although not covered in this article, many other methods are related to the popularity prediction of Web content. For example, Anderson et al. [5] propose to predict the long-lasting value which is defined as the pageviews of a question-answer thread in community question answering sites; Roy et al. [65] use transfer learning from the Twitter domain to the video domain to detect popularity burst; Cheng et al. [17; 18] propose to predict the phenomenon of large re-sharing behaviors in social network services; Cha et al. [15] find that the presence of multiple versions of the same content tends to limit the popularity of each individual copy; Lerman et al. [47] propose to predict which articles will get on the front page. A more thorough survey on these topics can be found in [74].

3. CONTENT QUALITY PREDICTION

In this section, we briefly review the existing work on the measurement and prediction of Web content quality.

3.1 Content Quality Measurement

In general, Web content quality is a vague concept, and it is related to a wide range of concepts like popularity, readability, conciseness, trustworthiness, helpfulness, etc. Additionally, content quality is a subjective concept and it depends on the factors that can be different among different social platforms. In other words, quality measurement tends to differ based on the underlying social platforms. In this article, to address the vagueness issue, we put our focus on the quality measurements that either are labeled by human annotators or can reflect the community users' overall opinions; to address the subjectiveness issue, we summarize the quality measurements on several typical types of social platforms as follows.

- **Collaborative content quality**. Collaborative content is the content that can be collaboratively edited by the registered users. Wikipedia article is a typical example. In Wikipedia, the quality of collaborative content is also collaboratively defined. That is, users can evaluate an article to several quality levels (e.g., Featured Article, A-Class, and B-Class) provided by Wikipedia, and these quality levels serve as the quality measurement [20].
- **Microblog credibility**. Rumors or non-credible messages may spread over microblogs like Twitter and Weibo, and evaluating the credibility of microblog messages becomes an important problem. To this end, human efforts are usually involved to judge if a microblog message corresponds to a newsworthy event [13; 14]. The official services for identifying rumors may also be used [88].
- **Question and answer quality**. Community question answering sites such as Yahoo! Answers and Stack Overflow contain rich knowledge, and thus have attracted much recent attention. There are several types of measurements for question/answer quality in these sites. The first one resorts to human annotators to manually label the question/answer quality [42; 35; 72]; the second one is the so-called questioner satisfaction, indicating whether an answer is chosen as the accepted/best answer by the questioner [53; 82; 66; 76]; the third one is based on the votes on the questions/answers [63; 89; 90; 85].
- **Review/comment helpfulness**. Reviews in many product review sites may contain noisy and misleading information. Therefore, identifying the helpfulness of the reviews becomes an important task in these sites. Similarly, it is also important to identify useful comments in social platforms like YouTube, Flickr, and Digg. To measure the review/comment helpfulness, the voting information is usually employed. Typically, the helpfulness is defined as the number of positive votes divided by the number of total votes [44; 69; 32; 40].

3.2 Content Quality Prediction

As we can see from the above four typical types of content quality problems, the quality is mainly reflected by either the community users' responses or the manual labeling. In the former case, it usually needs a relatively long time to accumulate sufficient user responses to accurately assess the quality. However, it would be more helpful if we can evaluate the content quality as soon as it is published. In the latter case, manual labeling is costly and it cannot scale to large data sets. Consequently, predicting the content quality as soon as it is published, and predicting with the input of a small amount of labeling information become necessary problems.

3.2.1 Prediction Time

The prediction time of quality prediction usually happens at the publication time. Methods in this category believe that many quality factors are inherent in the content itself. Also, these methods usually use human annotators to label quality. When the community's overall response is used as quality measurement, the prediction is usually made after publication. This is due to the fact that social platforms that allow users to evaluate the content quality may also provide other mechanisms where the content quality can be implicitly implied. For after publication methods in this category, the prediction should be made as soon as possible [90]. Although predicting content quality before publication is also feasible, seldom researchers consider this setting.

3.2.2 Features

Basically, the features used for content quality prediction resemble the features for popularity prediction as shown in Fig. 1. For example, Kim et al. [44] explore content features including structural features (e.g., the length of reviews), lexical features (e.g., unigrams), syntactic features (e.g., percentage of verbs and nouns), and semantic features (e.g., product feature mentions) for review helpfulness prediction. Based on these content features, Lu et al. [54] further incorporate the features from the reviewers. They exploit user features like the number of past reviews and social features like the out-degree of the reviewer in the social network.

In addition to the common features, some platform-specific features have been considered by existing content quality prediction methods. For example, when evaluating the quality of collaborative content, review features which are extracted from the review history of each article can be considered [20]. These features (e.g., the number of revisions) are indicative for the maturity and stability of an article. Additionally, since collaborative content involves many citations between each other, features can also be extracted from the citation network. These two types of features are actually complementary. That is, a mature article should be stable; however, it may be stable because no one is interested in it (e.g., in-degree) due to its poor quality. For microblogs, since there are extensive re-sharing behaviors, the propagation features about these re-sharing paths can be considered. Examples include the depth of the re-tweet tree, or the number of initial tweets of a topic [14]. For community question answering sites, complex relationships among questioners, answerers, questions, and answers have been considered [3]. The features like whether the service is free [35] and authors' offline reputation [75] are also considered.

3.2.3 Algorithms

Similar to popularity prediction, many off-the-shell regression and classification methods (e.g., SVR [41], SVM [69], neural networks [46]) are used by existing quality prediction methods.

Based on the characteristics of the underlying social platforms, some advanced algorithms are designed to finish the prediction task. For example, to predict microblog message credibility, Gupta et al. [34] propose to propagate credibility on a constructed network consisting of events, tweets, and users. One of their basic assumptions is that credible users tend to provide credible tweets. Similarly, to predict the quality of questions/answers, Bian et al. [8] propose to propagate the labels through user-question-answer graph with the assumption that reputable users tend to give high-quality questions/answers. These methods are also designed to tackle the sparsity problem where only a small number of messages/questions/answers are labeled. In review helpfulness prediction, Lu et al. [54] study two methods for incorporating social context into the prediction process: either as features, or as regularization constraints. The proposed regularization framework may also be used when a small amount of data is labeled.

3.3 Representatives

Here, we present some representatives.

Collaborative content quality. Dalip et al. [20] use SVR to estimate the quality of Wikipedia articles. They use content features and revision features. The content features include the length, style, readability, and structure of the article, and the citations between articles. Take structure as an example. The basic intuition is that a good article must be organized clearly, and it should provide necessary references to additional material. Therefore, features like section count and reference count are extracted. For revision features which are extracted from the revision history, they consider features like the discussion count, the review count, and the number of revisions in the last three months. The quality measurement is based on users' ratings, and user ratings are based on the quality criteria defined by Wikipedia which considers completeness, neutral point-of-view, good organization, factual accuracy, and provision of references to relevant sources of information.

Microblog credibility. Castillo et al. [13; 14] propose to assess the credibility of tweets. They first find some tweets that are about current hot topics, and then classify them as credible or non-credible by a group of human annotators. Next, they formulate a classification problem by considering a set of features. The considered features include content features (e.g., twitter-independent features like the length of a message and the number of positive/negative sentiment words, and Twitter-dependent features like if the tweet contains a hashtag and if the message is a re-tweet), user features (e.g., the accout age, the number of followers, and the number of followees), as well as propagation features as we previously mentioned. They show that the way microblog messages propagate (i.e., propagation features) is indicative for message credibility.

Question and answer quality. Jeon et al. [42] propose to use non-textual features to predict answer quality and incorporate it to improve retrieval performance. The used features include answerer's acceptance ratio, answerer's activity level, and answerer's category specialty. Kernel density estimation and the maximum entropy approach are used to handle different types of features, and a stochastic process is built to make the prediction. The quality labels are annotated by humans.

Agichtein et al. [3] propose to predict the quality of both questions and answers. They use some common features including content features and user features. They also extract several platform-specific features from the user-question-answer graph. Example platform-specific features include the average number of answers with references (URLs) given by the asker of the current question, the average number of positive votes received by answers written by the asker of the current question, etc. Finally, they lable the quality of questions and answers by human annotators, and use stochastic gradient boosted trees [29] to classify high-quality questions/answers.

Review/Comment helpfulness. Ghose and Ipeirotis [32] use content features and user features to predict the review helpfulness. The content features include subjectivity aspects and readability aspects. They find that reviews that have a mixture of objective and subjective sentences are rated more helpful. The user features consider the users' performance in previous reviews. They first use positive votes and negative votes to define quality, and then formulate a classification problem by choosing a threshold based on two human annotators. They adopt the random forest-based classifiers.

As to comment quality/helpfulness, Momeni et al. [56] propose to classify useful comments on YouTube and Flickr

with logistic regression and Naive Bayes. They use content features and user features. User features describe the users' posting and social behaviors (e.g., the size of the user's network). Content features include syntactic, semantic, and topical features. By human labeling, they show that semantic and topical features are very important for accurate classification for both Flickr and YouTube.

3.4 Summary

Measuring and predicting the content quality can help to identify some valuable information from the large amount of content in social platforms. In this review, we mainly focus on four types of quality measurement and prediction settings, i.e., the quality of collaborative articles, the credibility of microblog messages, the quality of questions and answers, and the helpfulness of reviews and comments. Different from popularity prediction in the previous section, content quality is a vague concept and many existing methods use human annotators to label the quality. However, human labeling is costly and not scalable. Therefore, community users' responses are also widely used as quality. As to the prediction, some platform-specific features and algorithms are designed by existing methods.

In this section, we only summarize some typical and mainstream social platforms. Within the covered social platforms, other types of measurements that are related to content quality have also been studied. In community question answering sites, examples include the usefulness of questions (i.e., the possibility that a question would be repeated by other people) [71], the arrival speed of the best answer [48], whether a question will be answered [25], site searcher satisfaction (i.e., whether or not the answer satisfies the information searcher) [52], and the subjectiveness of questions and answers [49; 95].

4. SCIENTIFIC IMPACT PREDICTION

In this section, we put our focus on the measurement and prediction of the impact of scientific articles.

4.1 Impact Measurement for Scientific Articles

Typically, the impact of a research article is closely related to the citations of this article, and the study of research article citations can date back to the 1960s and 1970s [21; 60]. Recently, it attracts attention after the citation count prediction competition in KDD Cup 2003 [30].

In literature, there are several ways to measure the impact of scientific articles, which are summarized as follows.

The first measurement is the number of articles that cite the given article (i.e., citation count). Citation count is a most straightforward measurement for article impact, and it is adopted by most existing work (e.g., [86; 81; 16; 70]). However, as noticed by several researchers, citation count has several disadvantages. For example, it tends to favor those aging articles, and it may be quite different in different fields [57].

The second measurement neglects the actual number of citations, but focuses on whether the citation count of a given scientific article is above the average (or above a certain percentile). For example, Newman [58; 57] proposes to predict highly cited articles, and argues that pure citation count may not be a good indicator for article impact because it is highly related to publication dates. Compared to citation count which can be viewed as a regression problem, this measurement can be viewed as a classification problem.

In addition to the above two measurements, Dong et al. recently propose to measure the impact of an article based on whether this article will increase its authors' h-indices in the future [22; 23]. This measurement can be seen as a personalized article impact as it depends on the authors of the article.

In addition to the above measurements in literature, other measurements for article impact may also be further studied. For example, we can consider the downloads of an article, and ratio between the citations and the downloads, the citation differences among different fields, etc.

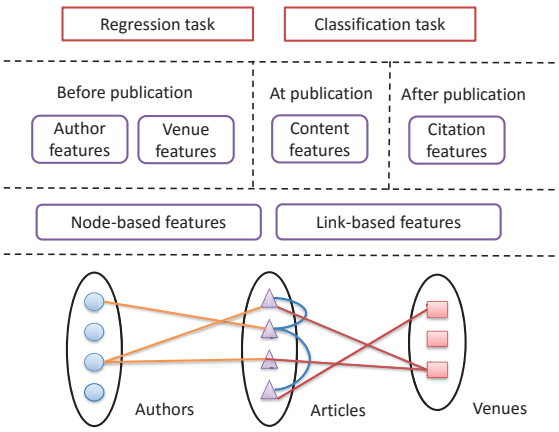


Figure 2: The features for scientific article impact prediction.

4.2 Impact Prediction for Scientific Articles

Next, we discuss the prediction time, features, and algorithms for scientific article impact prediction.

4.2.1 *Prediction Time*

Most existing work makes the prediction after publication time. For scientific articles, predicting their impact is not as urgent as the popularity/quality prediction of Web content, as scientific articles have a longer lifecycle than the general content on social media. For example, Wang et al. [81] make the citation prediction for an article five or even ten years after publication.

Next, since the research community is more stable and controlled, the research impact is more predictable than the content popularity. For example, some existing methods find that accurate predictions can be made when the prediction time is at publication or even before publication (e.g. [87; 86]).

4.2.2 *Features*

The features for article impact prediction are summarized in Fig. 2. First, the features are extracted from the heterogeneous network among authors, articles, and venues. In this heterogeneous network, authors write articles, articles are published on venues, and articles can cite each other. Although not explicitly drawn in the figure, authors are also related to authors (by collaboration or citation) and venues

(by publishing on the venues).
Rich features can be extracted from the heterogeneous network. We divide these features into node-based features and link-based features. Authors, articles, and venues all have some attributes from which node-based features can be extracted. For example, we can extract the total number of citations of authors, the topics of articles, and the h-indices of venues. For link-based features, we can apply PageRank-like algorithms on the network to achieve the authority of authors, centrality of venues, etc.
Based on the two types of node-based and link-based features, we further divide these features into four classes, i.e., author features, venue features, content features, and citation features.

- **Author features**. Author features reflect how well the authors have performed before the publication of the current article. Intuitively, these features may indicate the future performance of the authors. Typical example features include author productivity (e.g., the number of authors' previous publications), author influence (e.g., the number of authors' citations/h-indices), author sociality (e.g., the number of co-authors), author authority (e.g., PageRank value on author-author citation network), and authors diversity (e.g., the research fields that the author publishes).

- **Venue features**. Venue features indicate the impact of venues. High-impact venues may attract much attention from related scholars, and thus have a better chance to be cited. Similar to author features, venue features can be extracted before publication. Typical example features include venue influence (e.g., its h-index and citations), venue centrality (e.g., the PageRank value on venue-venue citation network), and venue diversity (e.g., the diversity of its articles).

- **Content features**. Intuitively, the content of the scientific article is a major factor that can affect its impact. To extract content features, topic modeling is widely used in existing literature. Typical content features include content novelty (e.g., the similarity between the article and its references), content diversity (e.g., the topical breadth of the article), and content popularity (e.g., the popularity of content topic).

- **Citation features**. Finally, when the prediction time can be lagged after publication, citation features may be very indicative for the impact prediction of scientific articles. The intuition behind is the richer-get-richer phenomenon. Here, the citation features are extracted based on the citations in a time window after publication. In fact, several researchers propose to predict the future citations of an article by fitting a function on the citations during the time window [81; 67]. Other citation features such as the total citations in the time window and the number of countries citing the current article in the time window have also been considered [93].

In addition to the above four classes of features, some researchers claim that they have considered the temporal features [87; 86]. Basically, these temporal features are applied on a recent short-term publication data, and the extracted features can also be categorized into the above four classes.

4.2.3 Algorithms

Based on the extracted features and the prediction target (i.e., impact measurement), the next component is the algorithm used to finish the prediction. Depending on the impact measurement, regression task and classification task can be formulated. For the regression task, methods such as linear regression, Gaussian Process regression [62], and SVR have been considered by existing literature. For the classification task, existing methods have considered logistic regression, decision trees, random forest, SVM, etc.
Unlike content quality prediction where platform-specific algorithms are designed, the characteristics of the underlying author-article-venue network is relatively ignored in terms of the algorithm level. We will discuss this more in Section 5.

4.3 Representatives

Here, we describe some representative methods for scientific impact prediction.
As an early practice, Castillo et al. [11] propose to extract both node-based and link-based author features as well as the number of citations after publication to predict the future citations. They state that their prediction can be done both before and after publication depending on whether to include the citation features. They formulate both regression and classification (top 10% cited articles as successful articles) problems, and use linear regression and C4.5 decision trees for them, respectively.
Although Castillo et al. [11] have considered a wide range of aspects, the extracted features are relatively simple. Yan et al. [87; 86] create a short-term data and extract content features, author features, and venue features from this short-term data as well as the full data. Since they do not use citation features, their prediction time is at publication or even before publication (if the content features are not used). As for prediction algorithm, they formulate a regression problem, and use both CART (classification and regression tree) and Gaussian Process regression to predict the citations after five and ten years.
The citation features are also widely used. For example, Chakraborty et al. [16] add first-year citation as features, and consider content features, author features, and venue features. The vary the time when the ground truth value of impact is computed, and observe the evolution of feature importance. One of their findings is that content features seem to have less significance in the initial few years, and become more effective in the later years. This result indicates that, to some extent, a high-quality article would eventually gain high impact irrespective of the reputation of the authors and the publication venue. Following Chakraborty et al. [16], Singh et al. [70] consider more citation features such as the number of times an article is cited within the same article and the number of words within the citation context. Both Chakraborty et al. [16] and Singh et al. [70] adopt a two-step regression formulation. That is, they first use SVM to divide articles into several predefined categories (based on citation patterns) and then use SVR to predict citations.
Taking only the citation information after publication as input is also studied. For example, Wang et al. [81] and Shen et al. [67] propose to mine the citation patterns of each article to predict its future citations. To fit the pattern function, these methods consider three factors, i.e., preferential

attachment (highly-cited articles are more visible and are more likely to be cited again), aging (article's novelty fades), and fitness (the inherent competitiveness against other articles).

Recently, Dong et al. [22; 23] propose to infer whether a given article will increase its authors' h-indices in the future. They use content features, author features, and venue features, and the prediction time is at publication. In content features, they further extract some reference features such as the average citations of the reference articles in the current article. They find that authors' authority on the topic and the publication venue are key factors for citations.

4.4 Summary

Measuring and predicting the impact of scientific articles have gained attention in recent years. Overall, we find three types of measurements for scientific article impact, and categorize four types of features for two types of tasks. A key difference from general content popularity/quality prediction is that article's scientific impact is more predictable and it can be accurately predicted at or even before publication. There are some interesting findings from existing literature. For example, authors' authority and publication venue are found to be the key factors for citations, while content features become more effective as time goes by.

In addition to scientific articles, some researchers propose to measure and predict the impact of research scholars [2; 24]. For example, h-index is proposed to measure both the productivity and impact of a research scholar [37]; later, the differences of disciplines and years are incorporated into h-index [61]; finally, the number of top-venue publications and the citations of big-hit publications are also used [24]. Some other researchers propose to measure the quality of conference/journal venues based on the overall citations of the venue [55] or the program committee characteristics [97]. Other related directions include recommending research articles for potential readers [80], recommending previous articles for citation [64], predicting citation relationships [94], ranking existing research articles [83], summarizing the research articles from multiple scholars viewpoints [1], mining the citation patterns for different articles [68], revealing the relationships between impact and combination of prior work [78], etc.

5. DISCUSSIONS AND FUTURE DIRECTIONS

In this section, we discuss some insights and present some future directions.

Prediction time v.s. prediction accuracy. Intuitively, there is a tradeoff between prediction time and prediction accuracy. An illustrative example is shown in Fig. 3. Time T1 in the figure means before publication. The accuracy is relatively low as only a small set of features are available before publication. Time T2 means at publication. At publication time, some useful features may be extracted from the published content. The curve means accuracy trend after publication. Intuitively, the prediction accuracy at time T4 would be higher than that at time T3, as more information is available. However, in many prediction tasks, the prediction should be made as early as possible, resulting a tradeoff between prediction accuracy and prediction time.

Based on our review of existing literature, seldom work has explicitly addressed this tradeoff. There are two reasons.

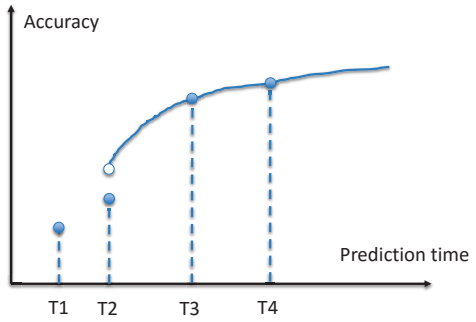


Figure 3: Prediction time v.s. prediction accuracy.

First, it is difficult to determine the time when the users' interest is exhausted. Second, experimenting over many time points would be costly. A possible future direction would be determining a suitable prediction time for different tasks. Take popularity prediction as an example. We may find the time when at least a certain (significant) amount of attention or a certain percentage of attention has not been received, especially for those highly popular content.

Learn from each other. The reviewed three tasks (i.e., content popularity prediction, content quality prediction, and scientific article impact prediction) bear some subtle differences, and they can borrow experiences and ideas from each other.

First, referrer features and external features have been considered by popularity prediction. Such features can also be used for the other prediction tasks. Take the impact prediction of scientific articles as an example. Whether the article is freely available on the Web and whether the authors have released their codes may also be helpful for the impact prediction. Second, some platform-specific features and algorithms are designed for content quality prediction. These ideas may also work for the other two tasks. Third, in scientific article impact prediction, the importance of features over time has been studied. Based on this study, some interesting findings have been observed. Similar study can be applied on the other tasks.

New measurements. For scientific impact, new measurements can be defined. The first direction is to make use of the popularity information. As mentioned above, we can consider the ratio between the number of citations and the number of downloads. The second direction is to add weights on the citation articles (i.e., the articles that cite the given article). Instead of pure citation count, we may define impact as weighted sum/average of the citation articles where the weight is based on the impact of the citation article itself. The intuition behind is that a citation from a high-impact article is more meaningful than that from a low-impact article. The third direction is to consider pathway measurements [51]. That is, instead of predicting the citation count at a given future time point, we can predict the citation numbers over several years. Such prediction is capable of visioning the impact trends of articles.

New algorithms. In the algorithm aspect, many existing methods apply off-the-shell machine learning algorithms to make the predictions. Here, some advanced algorithms that take the characteristics of the underlying social platforms into account can be designed. Existing examples in-

clude Agichtein et al. [3] who construct user-question-answer graph for the question/answer quality prediction task. However, their goal is to facilitate the label sparsity problem. More general methods for improving the prediction accuracy when we have enough labels are still needed. Recently, some efforts have been made towards this direction. For example, in question answering sites, the question-answer correlation has been considered to improve prediction accuracy [91]; in scientific research filed, the differences between different research domains have been considered [50].

Another future direction is to view the problem as a prediction problem on a data stream. Web content can be generated in a rapid speed. Therefore, a future direction is to design online algorithms that can efficiently make the predictions.

Additionally, some authors argue that the relative ranking rather than the actual value is more important [36]. However, they still use actual value as the prediction target. Here, pair-wised prediction target such as the learning to rank methods can be considered.

Beyond Web content. In this article, we focus on the measurement and prediction of Web content utility, where we define Web content as any individual item (in the form of text, image, or video) available on a web site in which a certain level of interest can be reflects by the community users. In addition to measuring and predicting on an individual item, we may also consider to predict the popularity or the research impact of a certain topic. Considering the fact the advertising usually choose a set of nodes, predicting the popularity of a given set of content nodes is also potentially useful.

6. CONCLUSIONS

In this article, we have reviewed the existing literature for the measurement and prediction of Web content. Specially, we focus on three tasks, i.e., content popularity prediction, content quality prediction, and scientific article impact prediction. We analyze the measurement definition, the prediction time, the extracted features, and the employed algorithms for each task. Finally, we identify the connections and differences between these three tasks, and discuss some future possible directions.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2016YFB1000802), National 863 Program of China (No. 2015AA01A203), National Natural Science Foundation of China (No. 61702252, 61672274, 61321491), the Fundamental Research Funds for the Central Universities (No. 020214380033), and the Collaborative Innovation Center of Novel Software Technology and Industrialization. Hanghang Tong is partially supported by DTRA under the grant number HDTRA1-16-0017, by Army Research Office under the contract number W911NF-16-1-0168, by National Institutes of Health under the grant number R01LM011986, Region II University Transportation Center under the project number 49997-33 25 and a Baidu gift.

7. REFERENCES

[1] A. Abu-Jbara and D. Radev. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 500–509. Association for Computational Linguistics, 2011.

[2] D. E. Acuna, S. Allesina, and K. P. Kording. Future impact: Predicting scientific success. *Nature*, 489(7415):201–202, 2012.

[3] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, pages 183–194. ACM, 2008.

[4] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini. A peek into the future: predicting the evolution of popularity in user generated content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 607–616. ACM, 2013.

[5] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *KDD*, pages 850–858. ACM, 2012.

[6] I. Arapakis, B. B. Cambazoglu, and M. Lalmas. On the feasibility of predicting news popularity at cold start. In *International Conference on Social Informatics*, pages 290–299. Springer, 2014.

[7] J. Berger and K. L. Milkman. What makes online content viral? *Journal of marketing research*, 49(2):192–205, 2012.

[8] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *WWW*, pages 51–60. ACM, 2009.

[9] Y. Borghol, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti. The untold story of the clones: content-agnostic factors that impact youtube video popularity. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1186–1194. ACM, 2012.

[10] A. Brodersen, S. Scellato, and M. Wattenhofer. Youtube around the world: geographic popularity of videos. In *Proceedings of the 21st international conference on World Wide Web*, pages 241–250. ACM, 2012.

[11] C. Castillo, D. Donato, and A. Gionis. Estimating number of citations using author reputation. In *International Symposium on String Processing and Information Retrieval*, pages 107–117. Springer, 2007.

[12] C. Castillo, M. El-Haddad, J. Pfeffer, and M. Stempeck. Characterizing the life cycle of online news stories using social media reactions. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 211–223. ACM, 2014.

[13] C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.

[14] C. Castillo, M. Mendoza, and B. Poblete. Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588, 2013.

[15] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking (TON)*, 17(5):1357–1370, 2009.

[16] T. Chakraborty, S. Kumar, P. Goyal, N. Ganguly, and A. Mukherjee. Towards a stratified learning approach to predict future citation counts. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 351–360. IEEE Press, 2014.

[17] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936. ACM, 2014.

[18] J. Cheng, L. A. Adamic, J. M. Kleinberg, and J. Leskovec. Do cascades recur? In *Proceedings of the 25th International Conference on World Wide Web*, pages 671–681. International World Wide Web Conferences Steering Committee, 2016.

[19] A. Choudhury and S. Nagaswamy. A bayesian approach to identify photos likely to be more popular in social media. *Journal of Computer and Communications*, 3(11):198, 2015.

[20] D. H. Dalip, M. A. Gonçalves, M. Cristo, and P. Calado. Automatic assessment of document quality in web collaborative digital libraries. *Journal of Data and Information Quality (JDIQ)*, 2(3):14, 2011.

[21] D. J. de Solla Price. Networks of scientific papers. *Science*, 149(3683):510–515, 1965.

[22] Y. Dong, R. A. Johnson, and N. V. Chawla. Will this paper increase your h-index?: Scientific impact prediction. In *WSDM*, pages 149–158. ACM, 2015.

[23] Y. Dong, R. A. Johnson, and N. V. Chawla. Can scientific impact be predicted? *IEEE Transactions on Big Data*, 2(1):18–30, 2016.

[24] Y. Dong, R. A. Johnson, Y. Yang, and N. V. Chawla. Collaboration signatures reveal scientific impact. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 480–487. IEEE, 2015.

[25] G. Dror, Y. Maarek, and I. Szpektor. Will my question be answered? predicting question answerability in community question-answering sites. In *ECML/PKDD*, pages 499–514, 2013.

[26] F. Figueiredo. On the prediction of popularity of trends and hits for user generated videos. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 741–746. ACM, 2013.

[27] F. Figueiredo, J. M. Almeida, M. A. Gonçalves, and F. Benevenuto. On the dynamics of social media popularity: A youtube case study. *ACM Transactions on Internet Technology (TOIT)*, 14(4):24, 2014.

[28] F. Figueiredo, J. M. Almeida, M. A. Gonçalves, and F. Benevenuto. Trendlearner: Early prediction of popularity trends of user generated content. *Information Sciences*, 349:172–187, 2016.

[29] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[30] J. Gehrke, P. Ginsparg, and J. Kleinberg. Overview of the 2003 kdd cup. *ACM SIGKDD Explorations Newsletter*, 5(2):149–151, 2003.

[31] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[32] A. Ghose and P. G. Ipeirotis. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1498–1512, 2011.

[33] R. E. Guadagno, D. M. Rempala, S. Murphy, and B. M. Okdie. What makes a video go viral? an analysis of emotional contagion and internet memes. *Computers in Human Behavior*, 29(6):2312–2319, 2013.

[34] M. Gupta, P. Zhao, and J. Han. Evaluating event credibility on twitter. In *SDM*, pages 153–164. SIAM, 2012.

[35] F. Harper, D. Raban, S. Rafaeli, and J. Konstan. Predictors of answer quality in online q&a sites. In *CHI*, pages 865–874. ACM, 2008.

[36] X. He, M. Gao, M.-Y. Kan, Y. Liu, and K. Sugiyama. Predicting the popularity of web 2.0 items based on user comments. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 233–242. ACM, 2014.

[37] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences of the United States of America*, pages 16569–16572, 2005.

[38] T. Hogg and K. Lerman. Stochastic models of user-contributory web sites. In *Third International AAAI Conference on Weblogs and Social Media*, 2009.

[39] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 57–58. ACM, 2011.

[40] Y. Hong, J. Lu, J. Yao, Q. Zhu, and G. Zhou. What reviews are satisfactory: novel features for automatic helpfulness voting. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 495–504. ACM, 2012.

[41] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *International Conference on Computational Science and Engineering*, volume 4, pages 90–97. IEEE, 2009.

[42] J. Jeon, W. Croft, J. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *SIGIR*, pages 228–235. ACM, 2006.

[43] A. Khosla, A. Das Sarma, and R. Hamid. What makes an image popular? In *Proceedings of the 23rd international conference on World wide web*, pages 867–876. ACM, 2014.

[44] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on empirical methods in natural language processing*, pages 423–430. Association for Computational Linguistics, 2006.

[45] J. G. Lee, S. Moon, and K. Salamatian. Modeling and predicting the popularity of online contents with cox proportional hazard regression model. *Neurocomputing*, 76(1):134–145, 2012.

[46] S. Lee and J. Y. Choeh. Predicting the helpfulness of online reviews using multilayer perceptron neural networks. *Expert Systems with Applications*, 41(6):3041–3046, 2014.

[47] K. Lerman and T. Hogg. Using a model of social dynamics to predict popularity of news. In *Proceedings of the 19th international conference on World wide web*, pages 621–630. ACM, 2010.

[48] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak. Analyzing and predicting question quality in community question answering services. In *WWW*, pages 775–782. ACM, 2012.

[49] B. Li, Y. Liu, and E. Agichtein. Cocqa: co-training over questions and answers with an application to predicting question subjectivity orientation. In *EMNLP*, pages 937–946, 2008.

[50] L. Li and H. Tong. The child is father of the man: Foresee the success at the early stage. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 655–664. ACM, 2015.

[51] L. Li, H. Tong, J. Tang, and W. Fan. *iPath: Forecasting the Pathway to Impact*, pages 468–476. 2016.

[52] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *SIGIR*, pages 415–424, 2011.

[53] Y. Liu, J. Bian, and E. Agichtein. Predicting information seeker satisfaction in community question answering. In *SIGIR*, pages 483–490. ACM, 2008.

[54] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th international conference on World wide web*, pages 691–700. ACM, 2010.

[55] W. S. Martins, M. A. Gonçalves, A. H. Laender, and N. Ziviani. Assessing the quality of scientific conferences based on bibliographic citations. *Scientometrics*, 83(1):133–155, 2010.

[56] E. Momeni, C. Cardie, and M. Ott. Properties, prediction, and prevalence of useful user-generated comments for descriptive annotation of social media objects. In *ICWSM*, 2013.

[57] M. Newman. Prediction of highly cited papers. *EPL (Europhysics Letters)*, 105(2):28002, 2014.

[58] M. E. Newman. The first-mover advantage in scientific publication. *EPL (Europhysics Letters)*, 86(6):68001, 2009.

[59] H. Pinto, J. M. Almeida, and M. A. Gonçalves. Using early view patterns to predict the popularity of youtube videos. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 365–374. ACM, 2013.

[60] D. d. S. Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American society for Information science*, 27(5):292–306, 1976.

[61] F. Radicchi, S. Fortunato, and C. Castellano. Universality of citation distributions: Toward an objective measure of scientific impact. *Proceedings of the National Academy of Sciences*, 105(45):17268–17272, 2008.

[62] C. E. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[63] S. Ravi, B. Pang, V. Rastogi, and R. Kumar. Great question! question quality in community q&a. In *ICWSM*, pages 426–435, 2014.

[64] X. Ren, J. Liu, X. Yu, U. Khandelwal, Q. Gu, L. Wang, and J. Han. Cluscite: Effective citation recommendation by information network-based clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–830. ACM, 2014.

[65] S. D. Roy, T. Mei, W. Zeng, and S. Li. Towards cross-domain learning for social video popularity prediction. *IEEE Transactions on multimedia*, 15(6):1255–1267, 2013.

[66] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community qa. In *SIGIR*, pages 411–418, 2010.

[67] H. Shen, D. Wang, C. Song, and A.-L. Barabási. Modeling and predicting popularity dynamics via reinforced poisson processes. In *AAAI*, pages 291–297. AAAI Press, 2014.

[68] X. Shi, J. Leskovec, and D. A. McFarland. Citing for high impact. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 49–58. ACM, 2010.

[69] S. Siersdorfer, S. Chelaru, W. Nejdl, and J. San Pedro. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *Proceedings of the 19th international conference on World wide web*, pages 891–900. ACM, 2010.

[70] M. Singh, V. Patidar, S. Kumar, T. Chakraborty, A. Mukherjee, and P. Goyal. The role of citation context in predicting long-term citation profiles: An experimental study based on a massive bibliographic text dataset. In *CIKM*, pages 1271–1280. ACM, 2015.

[71] Y. Song, C. Lin, Y. Cao, and H. Rim. Question utility: A novel static ranking of question search. In *AAAI*, pages 1231–1236, 2008.

[72] M. Suryanto, E. Lim, A. Sun, and R. Chiang. Quality-aware collaborative question answering: methods and evaluation. In *WSDM*, pages 142–151. ACM, 2009.

[73] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8):80–88, 2010.

[74] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis. A survey on predicting the popularity of web content. *Journal of Internet Services and Applications*, 5(1):1, 2014.

[75] Y. Tausczik and J. Pennebaker. Predicting the perceived quality of online mathematics contributions from users' reputations. In *CHI*, pages 1885–1888. ACM, 2011.

[76] Q. Tian, P. Zhang, and B. Li. Towards predicting the best answers in community-based question-answering services. In *ICWSM*, 2013.

[77] M. Tsagkias, W. Weerkamp, and M. De Rijke. Predicting the volume of comments on online news stories. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1765–1768. ACM, 2009.

[78] B. Uzzi, S. Mukherjee, M. Stringer, and B. Jones. Atypical combinations and scientific impact. *Science*, 342(6157):468–472, 2013.

[79] M. Vasconcelos, J. M. Almeida, and M. A. Gonçalves. Predicting the popularity of micro-reviews: A foursquare case study. *Information Sciences*, 325:355–374, 2015.

[80] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[81] D. Wang, C. Song, and A.-L. Barabási. Quantifying long-term scientific impact. *Science*, 342(6154):127–132, 2013.

[82] X.-J. Wang, X. Tu, D. Feng, and L. Zhang. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 179–186. ACM, 2009.

[83] Y. Wang, Y. Tong, and M. Zeng. Ranking scientific articles by exploiting citations, authors, journals, and time information. In *AAAI*, 2013.

[84] S. Weber and J. Luo. What makes an open source code popular on git hub? In *2014 IEEE International Conference on Data Mining Workshop*, pages 851–855. IEEE, 2014.

[85] X. Wei, H. Huang, C.-Y. Lin, X. Xin, X. Mao, and S. Wang. Re-ranking voting-based answers by discarding user behavior biases. In *AAAI*, pages 2380–2386, 2015.

[86] R. Yan, C. Huang, J. Tang, Y. Zhang, and X. Li. To better stand on the shoulder of giants. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 51–60. ACM, 2012.

[87] R. Yan, J. Tang, X. Liu, D. Shan, and X. Li. Citation count prediction: learning to estimate future citations for literature. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1247–1252. ACM, 2011.

[88] F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.

[89] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu. Joint voting prediction for questions and answers in cqa. In *ASONAM*, 2014.

[90] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu. Detecting high-quality posts in community question answering sites. *Information Sciences*, 302:70–82, 2015.

[91] Y. Yao, H. Tong, F. Xu, and J. Lu. Predicting long-term impact of cqa posts: a comprehensive viewpoint. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1496–1505. ACM, 2014.

[92] H. Yu, L. Xie, and S. Sanner. The lifecyle of a youtube video: Phases, content and popularity. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[93] T. Yu, G. Yu, P.-Y. Li, and L. Wang. Citation impact prediction for scientific papers using stepwise regression analysis. *Scientometrics*, 101(2):1233–1252, 2014.

[94] X. Yu, Q. Gu, M. Zhou, and J. Han. Citation prediction in heterogeneous bibliographic networks. In *SDM*, volume 12, pages 1119–1130. SIAM, 2012.

[95] T. Zhou, X. Si, E. Chang, I. King, and M. Lyu. A data-driven approach to question subjectivity identification in community question answering. In *AAAI*, 2012.

[96] H. Zhu, C. Liu, Y. Ge, H. Xiong, and E. Chen. Popularity modeling for mobile apps: A sequential approach. *IEEE transactions on cybernetics*, 45(7):1303–1314, 2015.

[97] Z. Zhuang, E. Elmacioglu, D. Lee, and C. L. Giles. Measuring conference quality by mining program committee characteristics. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 225–234. ACM, 2007.

Toward a Progress Indicator for Machine Learning Model Building and Data Mining Algorithm Execution: A Position Paper

Gang Luo
Department of Biomedical Informatics and Medical Education, University of Washington
UW Medicine South Lake Union, 850 Republican Street, Building C, Box 358047
Seattle, WA 98195, USA
luogang@uw.edu

ABSTRACT

For user-friendliness, many software systems offer progress indicators for long-duration tasks. A typical progress indicator continuously estimates the remaining task execution time as well as the portion of the task that has been finished. Building a machine learning model often takes a long time, but no existing machine learning software supplies a non-trivial progress indicator. Similarly, running a data mining algorithm often takes a long time, but no existing data mining software provides a non-trivial progress indicator. In this article, we consider the problem of offering progress indicators for machine learning model building and data mining algorithm execution. We discuss the goals and challenges intrinsic to this problem. Then we describe an initial framework for implementing such progress indicators and two advanced, potential uses of them, with the goal of inspiring future research on this topic.

Keywords

Machine learning, data mining, progress indicator, load management, automatic administration

1. INTRODUCTION

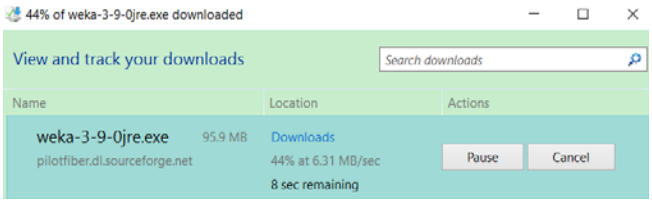


Figure 1. A determinate progress indicator for file downloading.

Progress indicators (see Fig. 1) are desired for long-duration tasks and widely used in software systems [49]. There are two types of progress indicators: indeterminate and determinate [5]. An indeterminate progress indicator like looped animation shows that the task is running, but gives no hint on when the task will finish. In contrast, a determinate progress indicator continuously estimates the remaining task execution time and/or the portion of the task that has been finished. This helps users better use their time. Frequently, even a rough estimate of the remaining task execution time can benefit users [8]. In human-computer interaction, a standard rule of thumb is that each task taking more than 10 seconds needs a determinate progress indicator [50, Chapter 5.5]. Also ideally, each task taking between two to 10 seconds should have an indeterminate progress indicator [5]. So far, sophisticated, determinate progress indicators have been designed for automatic machine learning model selection [41, 46] and built for database queries [11, 37, 43-45], program compilation [42], static program analysis [38], subgraph queries [77], and MapReduce jobs [47, 48]. Besides making the software more user friendly, determinate progress indicators can also help with workload management [42, 45, 59]. In the rest of the paper, we focus on determinate progress indicators, unless indicated otherwise.

1.1 Determinate progress indicators are needed for machine learning model building

As evidenced by multiple user requests, determinate progress indicators are desired for machine learning model building [1, 35] that typically takes a long time. On a modern computer, two days are needed to train the Practice Fusion diabetes classification competition's champion model [58] on 9,948 patients with 133 features. The average model building time is often several orders of magnitude longer than 10 seconds. It is unlikely to drop significantly in the foreseeable near future, although computers keep becoming faster. In fact, in many applications, model building time is increasing for two reasons. First, to obtain higher model accuracy, people keep developing more and more complex models such as deep neural networks with many hidden layers and ensembles of many base models. Model building time tends to grow with model complexity. Second, in the big data era, large data sets are commonly seen. Model building time often grows superlinearly with the training set size.

To reduce the entry bar to using machine learning, computer scientists and statisticians have developed multiple open source software tools integrating many machine learning algorithms. These tools include Weka [75], scikit-learn [55], RapidMiner, R, and KNIME [32]. With these tools in hand, more and more domain experts like healthcare researchers with limited machine learning knowledge start to build machine learning models by themselves. This reflects the industry trend of citizen data scientists, where an organization equips its talent with tools to perform deep analytics [26]. Frequently, these lay users significantly underestimate the time needed for building a model. If no determinate progress indicator is offered during the long period when a model is being built, they could mistakenly think that the tool has stopped working and become frustrated and/or discouraged from using it again. Just providing an indeterminate progress indicator like the one in Weka is insufficient for solving this problem.

In areas like healthcare, model generalization needs to be done frequently. It is a major reason causing domain experts with limited machine learning knowledge to build machine learning models by themselves, often for the first time in their life. In model generalization, a previously developed model needs to be rebuilt on a new data set (e.g., from a different healthcare system), possibly with new features. Frequently, the model rebuilding cost differs considerably from the original model building cost, as the data set's content affects model building cost. In this case, the user may know the original model building time, if it was reported by the people who developed the original model. But, this knowledge provides limited help for mitigating the problem mentioned above.

The situation becomes worse when lay users try to select machine learning algorithms and hyper-parameter values via multiple iterations, rather than using the default ones provided by a machine learning software tool. Many algorithms exist. Each algorithm has one or more hyper-parameters that a tool user needs to manually set before building a machine learning model. An example hyper-parameter is the number of hidden layers in a deep neural network. Different combinations of algorithms and hyper-parameter values often impact model accuracy by 40% or more [71] and model building cost by several orders of magnitude [80]. According to the "no free lunch" theorem [76], no single combination performs well on model accuracy for every modeling problem. Instead, the good choice of the combination varies by the specific modeling problem. To find a combination that can produce good model accuracy, a tool user typically needs to try many combinations and build one model per combination. A lay user often conducts this trial via random search, which is effective for selecting algorithms and hyper-parameter values [7]. Yet, if no determinate progress indicator is provided to give some hint on model building time, a lay user could be puzzled when experiencing vastly varying model building time across different combinations. This could cause the user to have low confidence in the tool and be unwilling to use it again.

Besides being useful for lay users, progress indicators for machine learning model building are also useful for computing experts. Progress indicators can help users better plan their time. In addition, progress indicators have other advanced, potential uses, such as load management and automatic administration described in Section 5.

1.2 No current machine learning software supplies a non-trivial progress indicator

Despite the need for determinate progress indicators for machine learning model building, to the best of our knowledge, no existing machine learning software supplies a non-trivial progress indicator and no technique has been published for supporting it. Similarly, running a data mining algorithm frequently takes a long time. But, no existing data mining software provides a non-trivial progress indicator and no technique has been published for supporting it, although it is desired.

For certain machine learning algorithms, some software offers trivial progress indicators for model building, e.g., by showing the number of decision trees that have been built in a random forest [62], the number of training epochs completed for a neural network [35], or the objective function's value reached [1] over time. Although such progress indicators are better than nothing, they cover limited algorithms and are too coarse for many purposes. During model training, the number of rounds needed for going through the training set is frequently unknown beforehand, and each round can take a long time.

Another way to offer a trivial progress indicator is to use an existing method to predict a machine learning model's building time before model building starts [15, 59-61, 63, 68]. If the model building task is predicted to take t seconds and has run for t' seconds, the remaining time is projected to be $t-t'$ seconds. Although better than nothing, such a trivial progress indicator tends to be rather inaccurate for two reasons. First, the predicted model building time is usually inaccurate. Second, the computer load may vary greatly due to other concurrently running tasks. Even if accurate on an unloaded computer, the predicted time may differ considerably from the actual model building time on a loaded computer.

1.3 Our contributions

In this paper, we consider the problem of offering non-trivial, determinate progress indicators for machine learning model building and data mining algorithm execution. We present the challenges intrinsic to this problem, an initial framework for implementing such progress indicators, two advanced, potential uses of them, and an initial framework for implementing their use. We hope this paper will stimulate future research on this topic.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 mentions the goals of progress indicators for machine learning model building and data mining algorithm execution. Section 4 presents an initial framework for implementing such progress indicators. Section 5 describes two advanced, potential uses of them and an initial framework for implementing their use. We conclude in Section 6.

2. RELATED WORK

Progress indicators

The human-computer interaction community has done much work on progress indicators [5, 8, 49, 50]. As mentioned in the introduction, sophisticated progress indicators have been designed or built for several types of tasks [11, 37, 38, 41-45, 47, 48, 77]. Yet, none of the work has addressed machine learning model building or data mining algorithm execution. Since different types of tasks have differing properties, the techniques developed in prior work [11, 38, 41-45, 47, 48, 77] cannot be used directly for machine learning model building or data mining algorithm execution.

Predicting job runtime

Researchers have done much work on predicting machine learning model building time [15, 59-61, 63, 68] and the runtime of a computer program [20, 25, 29, 65, 74], an iterative algorithm on graph data [57], and an algorithm for solving a combinatorial optimization problem [30]. The predicted model building time is often inaccurate and may differ considerably from the actual model building time on a loaded computer. Continuous refinement of predicted model building time is unavailable, but is required by progress indicators. Also, compared to general program execution, machine learning model building and data mining algorithm execution have their own characteristics, and hence more accurate cost models can be built for them. This will boost the precision of the progress indicator's estimates.

Researchers have developed techniques for predicting the runtime of a database query [22] and a program [39, 64, 66, 67] on a loaded computer cluster in a high-performance computing

center-like environment. These techniques, such as making prediction based on the user id, cannot well handle machine learning model building and data mining algorithm execution. The load on such a computer cluster has different characteristics from that in a typical environment (e.g., on a computer in the user's office), where a machine learning model is built or data mining algorithm is executed. Also, these techniques usually ignore hyper-parameters, whose values can greatly impact machine learning model accuracy and building cost [63, 75], as well as data mining results and algorithm execution cost. Like the case with machine learning, many data mining algorithms have hyper-parameters. Two example hyper-parameters for association rule mining are the minimum thresholds on support and confidence. For the same reason as hyper-parameters, the advanced, potential uses of progress indicators for machine learning model building and data mining algorithm execution are more involved and require more complex supporting techniques than those of progress indicators for database queries [45].

Complexity analysis

For building a machine learning model or running a data mining algorithm, much work has been done on providing the time complexity and theoretical bounds on the number of rounds needed for going through the training/data set [2, 70]. This information is insufficient for offering progress indicators and gives no estimate of model building or algorithm execution time on a loaded computer. Time complexity often excludes lower order terms and coefficients, which are needed for estimating model building or algorithm execution cost. The number of rounds needed depends on data properties. The theoretical bounds on it ignore data properties and are usually loose [54]. To support progress indicators, the estimated number of rounds needs to be periodically refined as model building or algorithm execution progresses.

Providing job runtime guarantee

In Section 5, we use progress indicators to facilitate automatic administration so that a machine learning model building or data mining algorithm execution job is likely to finish by a user-specified deadline. Real-time systems provide runtime guarantees for short jobs that usually finish within sub-seconds [34]. The techniques used there do not apply to model building and algorithm execution jobs, which take much longer to run.

Hu *et al*. [27] developed an approach that returns partial or approximate query results to provide runtime guarantees for database queries. Unlike our automatic administration approach in Section 5, that approach makes no attempt to adjust computing resources allocated to the query to meet the deadline.

In a cloud computing environment, many techniques have been developed for providing runtime guarantees for database queries [12, 52, 53, 78] and MapReduce-like jobs [14, 16, 21, 24, 31, 56, 72, 73], e.g., by estimating the number of computers that need to be allocated to a query/job for it to be likely to finish by a user-specified deadline. For the query/job, these techniques typically require using statistics collected from either its prior executions or first running it on many sample instances of the input data, do not continuously refine its estimated execution cost, and are not specifically designed and suitable for machine learning model building and data mining algorithm execution. In particular, none of these techniques considers changing the machine learning or data mining algorithm's hyper-parameter values to meet the deadline. Frequently, a model building or algorithm execution job is run for the first time, with no statistics available from its prior executions [72]. Running the job on many sample instances of the input data takes much time, causing an undesirable, long delay before any arrangement for job execution (e.g., on resource allocation) can be suggested. The initially estimated model building or algorithm execution cost is usually inaccurate and may differ considerably from the actual one. Properly changing hyper-parameter values often greatly reduces model building cost without much degradation of model accuracy. Sometimes, properly changing hyper-parameter values both reduces model building cost and boosts model accuracy. Given a fixed amount of computing resources, a model building job may not finish by the deadline even if all resources are given to the job. In this case, reallocating resources cannot help meet the deadline, but changing hyper-parameter values often can. Our automatic administration approach in Section 5 is specifically designed for machine learning model building and data mining algorithm execution jobs, requires no statistics from prior executions of the job, can suggest arrangements for job execution immediately after the job starts running, continuously refines the estimated job execution cost, and considers changing hyper-parameter values to meet the deadline.

Minimizing machine learning model building time

On a computer cluster, Huang *et al*. [28] developed an approach to find a near-optimal memory configuration to finish building a machine learning model as soon as possible. The approach offers no guarantee on model building time.

3. PROGRESS INDICATORS' GOALS

In this section, we discuss the goals of progress indicators for machine learning model building. The case with data mining algorithm execution is similar. Fig. 2 provides an example of the kind of progress indicator desired for model building. The interface is continuously updated. It displays the elapsed time, estimated remaining model building time, estimated percentage of the model building task that has been done, estimated model building cost, and current model building speed. Both the estimated model building cost and current model building speed are quantified by *U*. *U* is an abstract quantity depicting one unit of work and defined in Section 4.

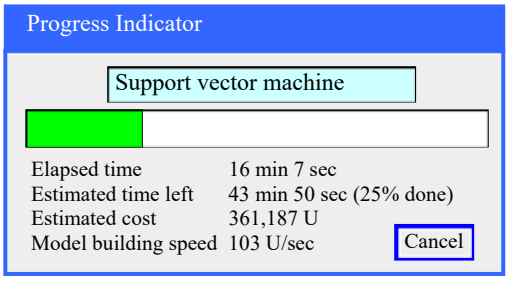


Figure 2. A progress indicator for machine learning model building.

An ideal progress indicator should fulfill the following four goals [42, 43]:

(1) **Continuously revised estimates**: At any moment, for any information displayed to users, the progress indicator should provide an estimate based on all information available about the computer and model building task at that moment. The estimate should be continuously revised based on more

accurate information about the task and changes in model building speed.

(2) **Acceptable pacing**: The progress indicator's update rate should be sufficiently high for users to see a smooth display, but not be too high to overburden the user interface or users.

(3) **Minimal overhead**: The progress indicator should have little impact on model building efficiency.

(4) **Maximal functionality**: Many machine learning algorithms exist, each with one or more hyper-parameters. The progress indicator should support many algorithms and give useful information for each hyper-parameter value combination of a supported algorithm.

4. A FRAMEWORK FOR IMPLEMENTING PROGRESS INDICATORS

In this section, we present an initial framework for implementing progress indicators for machine learning model building and data mining algorithm execution. Our key idea is as follows. During machine learning model building, we collect various statistics like the number of data instances and number of model building iterations that have been processed. As a model is being built, more accurate information of the model building task becomes available. We use this information to continuously revise the estimated model building cost. We keep monitoring the model building speed defined as the number of *U*s processed per second. At any moment, the remaining model building time is estimated as the estimated remaining model building cost divided by the observed current model building speed. Periodically, the progress indicator displays to users the latest estimates. The case with data mining algorithm execution is similar.

Section 4.1 discusses how to select the work unit *U* and convert it to time. Section 4.2 describes how to initially estimate the costs of building a machine learning model and running a data mining algorithm, and continuously refine the estimates. Section 4.3 presents how to monitor machine learning model building and data mining algorithm execution speeds. Section 4.4 discusses how to handle distributed and parallel computing.

4.1 Selecting *U* and converting to time

The following discussion focuses on machine learning model building. The case with data mining algorithm execution is similar. As described in Section 3, both the estimated model building cost and current model building speed are quantified by the abstract unit *U*. Each *U* depicts one unit of work. We intentionally make the statement vague and general, as several viable ways exist for defining *U*. The main requirements on the definition are: (1) we can estimate how many *U*s it will take to build the model, and (2) we can go from *U*s to estimated time, about which users tend to care most. CPU (central processing unit) cycles, I/Os (inputs/outputs), and a weighted combination of them are all potential candidates for *U*.

Similar to the approach Luo *et al*. [43] used and for simplicity, we set *U* to be one data instance when data are in the form of a set of data instances. As a rough proxy for CPU cycles and I/Os, this is easy to measure. The cost of building a machine learning model is the total number of data instances that need to be processed counting repeated processing. For instance, if we need to go through the training set *r* rounds to build the model, each training instance is counted *r* times. When data are in the form of sequences of observations, such as in the case of a hidden Markov model, we set *U* to be one observation. The cost of building a model is the total number of observations that need to be processed counting repeated processing. How to refine *U*'s definition to improve the estimates without incurring excessive overhead is an interesting area for future work.

Before model building starts, we compute an initial estimate of the model building cost. Before offering its first estimate of the remaining model building time, the progress indicator watches for some time to see how quickly *U*s are processed. As model building progresses, the conversion factor from *U* to time will change based on the observed current model building speed.

4.2 Initially estimating the costs of building a machine learning model and running a data mining algorithm, and continuously refining the estimates

Many machine learning and data mining algorithms exist. Often, multiple implementation methods exist for an algorithm [2, 75]. This paper's goal is not to cover all algorithms and implementation methods. Instead, our goal is to provide a framework and illustrate at a high level that it is feasible to support progress indicators for machine learning model building and data mining algorithm execution. In this section, we select several commonly used algorithms. For each algorithm, we choose one typical implementation method described in Alpaydin *et al*. [2, 4] and present the high-level idea of how to initially estimate the model building or algorithm execution cost and continuously refine the estimate. Before discussing how to handle each individual algorithm, we first describe a general technique useful for handling multiple algorithms.

4.2.1 A general technique useful for handling multiple algorithms

The following discussion focuses on machine learning model building. The case with data mining algorithm execution is similar. For many machine learning algorithms, building a model takes multiple iterations. An example of an iteration is to go through the training set once. To estimate the model building cost, we need to estimate both the number of needed iterations and each iteration's cost. Before model building starts, we obtain an initial estimate of each iteration's cost based on the training set size. If the initial estimate tends to be inaccurate, as model building progresses, we regularly use the observed, previous iterations' costs to refine each subsequent iteration's estimated cost. For instance, starting from the second iteration, we estimate each subsequent iteration's cost as the observed average cost of all previous iterations.

Frequently, the goal of doing iterations is to minimize the value of an objective function like negative log-likelihood and the number of misclassifications. We keep iterating until the objective function's value meets a pre-defined convergence condition. The number of needed iterations is unknown beforehand. In this case, before model building starts, we can conduct meta-learning to obtain an initial estimate of the number of needed iterations. For a machine learning algorithm, meta-learning uses historical data from model building on previous data sets to train a predictive model. The predictive model forecasts the number of needed iterations based on a data set's feature values and the algorithm's hyper-parameter values. Meta-learning was used before to predict model building time [15, 59-61, 63, 68].

The objective function's value decreases, whereas the rate of decrease typically drops over iterations (Fig. 3). Starting from when a fixed number (e.g., 4) of iterations are done, we have accumulated some information on the objective function's value over time. If the initial estimate of the number of needed iterations tends to be inaccurate, we regularly use this information to refine the estimate. More specifically, we use an inverse power law function of the form $f(n)=a+b\times n^{-c}$ [19] to model the objective function's value over iterations. Here, n denotes the number of iterations, $b>0$, and $c>0$. The inverse power law function was used before to model the relationship between model accuracy and training set size [19]. In our case, the inverse power law function is re-fitted periodically to project the objective function's value over iterations. The projection is used to estimate when the objective function's value will meet the pre-defined convergence condition, based on which we refine the estimated number of needed iterations. For a machine learning algorithm, if the inverse power law function cannot fit the objective function well, we perform modeling using another monotonically decreasing function like $g(n)=a+b\times n^{-c}+d\times n^{-e}$, where $b>0$, $c>0$, $d>0$, $e>0$, and $c\neq e$ [54].

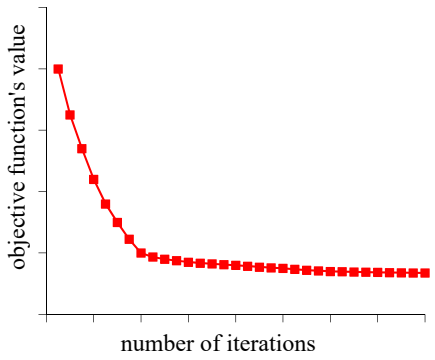


Figure 3. An objective function's value over iterations.

Next, we discuss how to handle individual machine learning and data mining algorithms one by one.

4.2.2 Handling supervised machine learning algorithms

Naive Bayes, linear discriminant analysis, linear regression

Training a naive Bayes classifier requires going through the training set once. The model building cost in U = the number of training instances. The cases with linear discriminant analysis and linear regression are similar.

Logistic regression, lasso regression

Building a logistic regression model requires multiple iterations, each going through the training set once. Each iteration's cost in U = the number of training instances. We use the technique in Section 4.2.1 to estimate the number of needed iterations. The case with lasso (least absolute shrinkage and selection operator) regression is similar.

Decision tree

We consider the common case of using a univariate, binary decision tree. During tree building, we track the number of training instances reaching each internal node of the tree. When we reach the node, if this number is larger than a pre-determined threshold (e.g., 2,000), we use this number to refine the estimated cost of building the sub-tree rooted at the node. If this number is no larger than the pre-determined threshold, we can build the sub-tree and know its actual building cost quickly without incurring any estimation overhead.

A decision tree's building cost depends on how balanced the tree is. A decision tree is often reasonably, even if not perfectly, balanced [30]. We use this as a heuristic in estimating tree building cost. Initially before tree building starts, we estimate the tree building cost by assuming that the tree is perfectly balanced. When reaching an internal node of the tree during tree building, if needed, we refine the estimated cost of building the sub-tree rooted at the node by assuming that the sub-tree is perfectly balanced.

The tree building cost has three components: one for inspecting every feature at each internal node of the tree to decide the test function to be used at the node, one for splitting all training instances reaching each internal node into two partitions based on the test function used at the node, and one for pruning the tree after it is formed. Based on the tree's hierarchical structure, the first two components can be estimated in a recursive way.

We estimate the cost of inspecting each feature at an internal node of the tree as follows. To inspect a categorical feature, we need to go through all training instances reaching the node once. The corresponding cost in U = the number of these training instances. To inspect a numerical feature, we need to sort all training instances reaching the node. The sorting cost is estimated using the method described in our prior work [44].

To split all training instances reaching an internal node of the tree into two partitions, we need to go through these training instances once. The corresponding cost in U = the number of these training instances.

The cost of pruning the tree is estimated as the number of data instances in the pruning set multiplied by a factor. The factor is projected via meta-learning.

Decision stump

The case with decision stump is similar to that with decision tree. A decision stump is a one-level decision tree. Its building cost has two components: one for inspecting every feature at the root node of the tree to decide the test function to be used at the node, and another for splitting all training instances into two partitions based on the test function used at the node.

Random forest

A random forest is an ensemble of decision trees. The cost of building a random forest = the sum of the cost of building each tree in the random forest. During the process of building a tree, we use the method described above to continuously refine the estimated cost of building the tree, considering the factors that (1) a fixed fraction of all training instances are used to build the tree, (2) a fixed fraction of all features are inspected at each internal node of the tree, and (3) no pruning is needed for building the tree. When building the i-th ($i\geq 2$) tree, we estimate the cost of building each subsequent tree as the observed average cost of building each of the previous i-1 trees.

Neural network

A neural network is trained in epochs. Each epoch requires going through all training instances once, with a cost in U = the number of training instances.

Support vector machine

Training a support vector machine requires solving a convex optimization problem. The training cost depends on which optimization algorithm is used to solve the problem [9]. The optimization algorithm runs in multiple iterations. To estimate the optimization algorithm's execution cost, we need to estimate both the number of needed iterations and each iteration's cost, possibly using a technique similar to that in Section 4.2.1. Due to such optimization algorithms' complex nature [9], figuring out how to do this exactly needs non-trivial future work.

Voting, bagging, and Adaboost

For voting, the final model is an ensemble of multiple models. The cost of building the final model = the sum of the cost of building each individual model. The cases with bagging and Adaboost are similar. For bagging, we need to consider that each individual model is built using a fixed fraction of all training instances.

Expectation-maximization, hidden Markov model, Kalman filter, and chain-structured graphical model

Using the expectation-maximization algorithm to train a hidden Markov model takes multiple iterations. Each iteration goes through every sequence of observations in the training set for F passes (either forward or backward), where F is a known constant. We use the technique in Section 4.2.1 to estimate the number of needed iterations. Each iteration's cost in $U = F \times$ the total length of all sequences of observations in the training set. Alternatively, we consider the factor that the average CPU cost of going through an observation varies across passes. During model building, we collect statistics on and continuously revise our estimate of the average CPU cost of going through an observation for every pass. In estimating the model building cost, we give differing weights to an observation in different passes accordingly. The cases with Kalman filter and chain-structured graphical models like linear-chain conditional random field are similar.

4.2.3 Handling unsupervised machine learning algorithms

k-means clustering

Building a k-means clustering model takes multiple iterations, each going through the training set once. We use the technique in Section 4.2.1 to estimate the number of needed iterations. Each iteration's cost in U = the number of training instances.

4.2.4 Handling data mining algorithms

Association rule mining

We first consider using the Apriori algorithm [3] to mine frequent itemsets and subsequently association rules. The Apriori algorithm runs in multiple iterations, each going through the data set once. The j-th iteration identifies all frequent j-itemsets, each with j items. By setting U to be one transaction, each iteration's cost in U = the number of transactions in the data set.

Before running the Apriori algorithm, we conduct meta-learning to obtain an initial estimate of the number of needed iterations. As j increases, the support of a frequent j-itemset tends to decrease. During algorithm execution, one possible way to refine the estimated number of needed iterations is to use a monotonically decreasing function like that used in Section 4.2.1 to model the dependency of the median support of all frequent j-itemsets on j. The number of needed iterations is estimated based on when the projected median support drops below the required minimum support threshold.

Next, we consider using the frequent pattern (FP)-growth algorithm [23] to mine frequent itemsets and subsequently association rules. The FP-growth algorithm runs in three steps. In step 1, we go through the data set once to identify all frequent items. In step 2, we go through the data set once to build an FP-tree T for the whole data set. In step 3, for each frequent item I in T, we construct I's conditional pattern base B_I from T and build a conditional FP-tree T_I for B_I in a way similar to that of steps 1 and 2. B_I corresponds to all of the transactions in the data set containing I. After T_I is built, we form a conditional pattern base and a conditional FP-tree for each frequent item in T_I recursively, until the formed conditional FP-tree contains a single path.

Both step 1's cost in U and step 2's cost in U = the number of transactions in the data set. Initially, we estimate step 3's cost in U as the number of transactions in the data set multiplied by a factor. The factor is projected via meta-learning. After finishing step 2, we know every frequent item I in the FP-tree T and the size of I's conditional pattern base B_I. We use this information to refine the estimated cost of recursively processing B_I in step 3, and re-compute step 3's estimated cost as the sum of the estimated costs of recursively processing B_I over every I. During recursive processing of B_I in step 3, we refine the processing's estimated cost if B_I's size is larger than a pre-determined threshold (e.g., 500). If B_I's size is no larger than the pre-determined threshold, we can finish recursively processing B_I and know the processing's actual cost quickly without incurring any estimation overhead.

4.3 Monitoring machine learning model building and data mining algorithm execution speeds

The following discussion focuses on machine learning model building. The case with data mining algorithm execution is similar. The progress indicator converts U to time based on what is observed as model building progresses. At all times, we track the amount of model building work in Us that has been completed in the past K seconds. K is a pre-determined number like 10. The current model building speed is estimated as the average model building speed in the past K seconds. K should be reasonably large to reduce the impact of temporary fluctuations. Also, K should not be too large to cause the computed model building speed to differ considerably from the actual one.

4.4 Handling distributed and parallel computing

The discussion so far focuses on sequential execution of machine learning and data mining algorithms. In the big data era, large data sets are common. To build a machine learning model or run a data mining algorithm on a large data set in reasonable time, we often need to do distributed or parallel computing, e.g., on a computer cluster [6, 79]. There, multiple processing elements are used to do the work concurrently, each handling a portion of the work. Two examples of a processing element are a computer and a CPU core. To support progress indicators in this case, we

designate one processing element as the master one. Each processing element periodically sends the latest statistics, including its progress, on its portion of the work to the master processing element. The master processing element aggregates the statistics from all processing elements and computes the overall estimates to be displayed to users. In particular, the remaining model building or algorithm execution time is estimated based on the processing element that will finish its portion of the work last among all processing elements. When estimating model building or algorithm execution costs, we consider both computation and communication costs and model the objective function's value's dependency on the number of processing elements in a way similar to that in Pan *et al* [54, 69, 72]. If a processing element fails, we re-compute the model building or algorithm execution cost by factoring in the cost of redoing the part of the job originally scheduled on this processing element.

5. ADVANCED, POTENTIAL USES OF PROGRESS INDICATORS

In this section, we describe two advanced, potential uses of progress indicators for machine learning model building and data mining algorithm execution, as well as an initial framework for implementing these two uses. We focus on machine learning model building. The case with data mining algorithm execution is similar.

5.1 Load management

Suppose that multiple machine learning models are being built on the same computer or computer cluster. For some reason, the execution of some model building tasks needs to be blocked so that other models can be built faster. Using the estimates provided by progress indicators in a way similar to that in Luo *et al*. [45], we can add a load management function into the machine learning software to help decide which tasks should be blocked.

5.2 Automatic administration

Sometimes, the user of a machine learning software tool needs or would like to finish building a machine learning model within a certain amount of time, such as 30 minutes or two hours. Two examples of such a scenario are as follows:

(1) Model building is iterative and involves explorations. The user starts from an initial set of features. If it yields low model accuracy, the user considers adding new features to boost accuracy. Each iteration requires building one or more models. To expedite the iteration process, the user often wants to quickly build a model on a given set of features and have a rough estimate of its potential. This helps the user decide the most effective way to spend his/her time. If the current set of features looks promising, the user may switch to fine-tuning the machine learning algorithm and/or its hyper-parameter values without further changes of the features. Otherwise, if the current set of features looks unpromising, the user may keep checking new features, increase the amount of historical data used (e.g., from one to two years), change the dependent variable to make it easier to predict (e.g., from making predictions six months in advance to making predictions two months in advance) [33], or conclude that the data set contains insufficient information for reaching high model accuracy. In the last case, the user can move on to another modeling problem rather than keep wasting time, effort, and resources on the current one [36].

(2) In an Internet company, new problems keep coming up on a daily basis, e.g., because of customer behavior change or a new event's occurrence. When a new problem arises, e.g., in fraud detection, an engineer of the company often needs to figure out and deploy a solution to it as quickly as possible to minimize potential business loss. If the solution involves using a model, the engineer needs to build the model quickly, possibly on a large data set. Even if the engineer is a machine learning expert and has built models on the same large data set for similar problems before, he/she may have no good estimate of model building time for the new problem himself/herself. To handle the new problem, the engineer often needs to use a new machine learning algorithm and/or new hyper-parameter values. This can change model building cost by one or more orders of magnitude. Even if the deadline is not tight, there is still a limit on when the engineer should finish model building. Moreover, building a model on a large data set consumes a lot of computing resources and incurs a high cost correlated with model building time. Even in a rich Internet company, the engineer often has a limited budget for model building and would appreciate a tool that can facilitate budget control.

To the best of our knowledge, no existing machine learning software provides guarantees for machine learning model building time. Using the estimates provided by progress indicators, we can add an automatic administration function into the machine learning software. The user inputs his/her desired, soft deadline. Then the function lists one or more representative arrangements, if any, as its suggestions. Adopting any of these suggestions will make model building likely to finish by the user-specified deadline. To help the user know his/her options, multiple suggestions are provided whenever possible to illustrate the range of feasible arrangements. Each suggestion is accompanied by an estimated financial cost of task execution as well as an estimated remaining model building time by adopting the suggestion, is of one of two types (allocating more resources and changing hyper-parameter values), and is adjusted from time to time based on the latest estimates provided by the progress indicator. The financial cost of task execution is estimated based on the allocated computing resources and projected model building time. Each suggestion on adjusting hyper-parameter values is accompanied by an estimated change in the resulting model's accuracy by adopting the suggestion. Suggestions on allocating more resources have no impact on model accuracy. Since changing hyper-parameter values can affect model accuracy, we let the user decide which suggestion to adopt rather than automatically adopt a suggestion for him/her without obtaining his/her consent first.

In the following discussion, unless indicated otherwise, we assume that when the automatic administration function is invoked, the user of the machine learning software tool has already specified an arrangement, including the resources allocated to the machine learning model building task, the machine learning algorithm, and its hyper-parameter values. Also, the progress indicator shows that the task is not going to finish by the deadline desired by the user. The case that the user wants to directly obtain suggestions from the automatic administration function without first fully inputting his/her own arrangement can be handled similarly.

We use two types of suggestions.

5.2.1 The first type of suggestion

The first type of suggestion is on allocating more resources to the remaining part of the machine learning model building task. To increase the degree of parallel processing, more cores of a CPU or more computers in a computer cluster are allocated to the task. For example, the decision trees remaining to be built in a random forest are constructed concurrently via multi-threading rather than sequentially in a single thread. When invoking the automatic administration function, the user specifies the maximum resources, such as the maximum number of computers, that can be used [53, 72]. This poses an upper bound on the resources that a suggestion of the first type can recommend using.

5.2.2 The second type of suggestion

Overview

The second type of suggestion is on changing the machine learning algorithm's hyper-parameter values. This type of suggestion is particularly useful for lay users with limited machine learning knowledge. Many lay users are unaware that different hyper-parameter values can impact the cost of building a machine learning model by several orders of magnitude. These users frequently specify, e.g., in random search [7], inappropriate hyper-parameter values that lead to excessively long model building time. To make the machine learning software more user friendly, we can let the software's automatic administration function be proactive. When the function detects that a model's estimated building cost is excessively large both in absolute value and in relationship to the data set size, the function automatically prompts the user to reconsider the hyper-parameter values, without the user having to explicitly invoke the function himself/herself. We use a pre-determined constant as the threshold for deciding whether a model's estimated building cost is excessively large in absolute value. For each algorithm, using historical data from model building on previous data sets, we fit a function describing the relationship between reasonable model building costs and the data set size. We multiply the value computed by the function by a constant factor as the threshold for deciding whether a model's estimated building cost is excessively large in relationship to the current data set's size.

The automatic administration function will suggest value changes only for the hyper-parameters whose values can potentially have a large impact on model building cost. Based on its value's potential impact on model building cost, we categorize each hyper-parameter into one of the following two types:

(i) For a hyper-parameter of the first type, its value typically has little or no impact on model building cost. An example of such a hyper-parameter is the seed of the random number generator in a neural network.

(ii) For a hyper-parameter of the second type, its value can potentially have a large impact on model building cost. The second type includes two classes:

  (a) In the first class, for each possible direction of value change like value increase, changing the hyper-parameter's value along the direction will always increase or decrease model building cost, depending on the specific direction. An example of such a hyper-parameter is the number of decision trees in a random forest. Increasing the number of decision trees will always increase the cost of building the random forest.

  (b) In the second class, changing the hyper-parameter's value along a particular direction can either increase or decrease model building cost, depending on the specific data set used. An example of such a hyper-parameter is the γ in a support vector machine with a Gaussian radial basis function kernel. As shown in Reif *et al*. [63], increasing γ's value can either increase or decrease the cost of training the support vector machine, depending on the specific data set used.

The automatic administration function will suggest value changes only for hyper-parameters of the second type. For a hyper-parameter in the first class of the second type, the suggested change is always along the specific, known direction that will decrease model building cost.

Our desired goals of suggesting changes to hyper-parameter values

When suggesting changes to the machine learning algorithm's hyper-parameter values, we strive to make every suggestion fulfill the following four goals whenever possible:

(1) **Affecting minimal number of hyper-parameters**: The suggested changes should affect the values of as few hyper-parameters as possible. Ideally, only one hyper-parameter is impacted.

(2) **Minimal change to the hyper-parameter value**: When a hyper-parameter is affected, the suggested change to its value should be as small as possible. The first two goals try to preserve the hyper-parameter values specified by the user as much as possible. This helps increase the chance that the user will adopt the suggestion.

(3) **Minimal decrease of model accuracy**: Hyper-parameter values impact both model accuracy and building cost. The suggested changes to hyper-parameter values should not significantly degrade the resulting model's accuracy.

(4) **Considering reusing the results that have already been computed**: Changing hyper-parameter values does not necessarily mean that the model has to be re-built from scratch. In estimating the remaining model building time by adopting the suggested changes to hyper-parameter values, we should consider the possibility of reusing the results that have been computed so far and continuing the model building process.

Implementation techniques to fulfill the first three goals

When computing the list of suggestions on changing the machine learning algorithm's hyper-parameter values, we initially consider that each suggestion changes a single hyper-parameter's value. We conduct meta-learning to project both the resulting model's accuracy and building cost by adopting a suggestion. For each hyper-parameter whose value can potentially have a large impact on model building cost, we consider making the smallest change to its value, which causes the model building task's estimated completion time to be no later than the user-specified deadline. If such a change in value exists, we keep it as a suggestion if the model accuracy projected using it is not lower than that projected using the current hyper-parameter values by more than a pre-determined threshold, such as 2%.

A modeling problem includes both the problem specification and the data set used. Our meta-learning approach uses historical data from building models for other modeling problems, as well as data from the user's prior trials of building models for the current modeling problem. The latter provides information on the current modeling problem and is given a higher weight than the former. To check whether a prior trial of the user and the current

model building task handle the same modeling problem, we compare the file names, numbers of data instances, and numbers of features of the data sets, as well as the names of the dependent variables used in them. Such a comparison will not ensure 100% accuracy for the check, but can be quickly done with reasonably good accuracy. In this way, we avoid detailed comparisons of data set contents, which can be costly to do on large data sets.

After going through every hyper-parameter whose value can potentially have a large impact on model building cost, if we have obtained at least one feasible suggestion on changing a hyper-parameter's value, we stop the process of computing suggestions on changing hyper-parameter values and display the obtained suggestions. Otherwise, if changing a single hyper-parameter's value is insufficient for finishing model building by the user-specified deadline, we continue to consider that each suggestion changes two hyper-parameters' values simultaneously. If that is still insufficient for finishing model building by the deadline, we continue to consider that each suggestion changes three hyper-parameters' values simultaneously, and so on.

For tasks such as scheduling machine learning model building jobs [59] and automatic machine learning model selection [10, 17, 18, 40, 68], meta-learning has been used before to predict model accuracy and building time. There, no constraint is put on the hyper-parameter value combinations selected for testing. In comparison, in our case of suggesting changes to hyper-parameter values, we prefer the suggested combinations to be as close to the one specified by the user as possible.

Implementation technique to fulfill the fourth goal

Recall that only the hyper-parameters whose values can potentially have a large impact on model building cost are of interest to the automatic administration function. To fulfill the fourth goal mentioned above, we categorize each hyper-parameter of interest into one of the following three types based on the degree to which, after changing the hyper-parameter's value to cut model building cost, we can reuse the results that have been computed so far and continue the model building process:

(i) For a hyper-parameter of the first type, if we change its value to cut model building cost, we can reuse all of the results that have been computed so far and continue the model building process. For example, if we decrease the number of decision trees in a random forest, we can reuse all of the decision trees that have been formed so far and continue to build the random forest. As another example, if we decrease the number of training epochs in a neural network, we can reuse all of the results computed in the prior epochs and continue to train the neural network.

(ii) For a hyper-parameter of the second type, if we change its value to cut model building cost, we can reuse possibly only part of the results that have been computed so far and continue the model building process. One example of such a hyper-parameter is the minimal number of data instances at every leaf node in a decision tree. When we increase this number from n_{old} to n_{new}, if a specific leaf node has already been formed and contains fewer than n_{new} data instances, we can keep merging this leaf node with its sibling nodes until the merged node includes at least n_{new} data instances. Alternatively, we can keep this leaf node as it is and only require the leaf nodes remaining to be formed to each include at least n_{new} data instances.

(iii) For a hyper-parameter of the third type, if we change its value to cut model building cost, we would need to re-build the model from scratch. Two examples of such a hyper-parameter are the kernel and regularization constant C used in a support vector machine.

In estimating the remaining model building time by adopting the suggested changes to hyper-parameter values, if all suggested changes happen to hyper-parameters of the first two types, we consider the factor of reusing all or part of the results that have been computed so far and continuing the model building process. If all suggested changes happen to hyper-parameters of the first type, we can reuse all of the results that have been computed so far. If at least one of the suggested changes happens to a hyper-parameter of the second type, we can reuse part of the results that have been computed so far.

5.2.3 Additional details

Besides the two types of suggested changes mentioned above, we can also make suggestions that each include changes of both types. Moreover, the two types of suggested changes mentioned above do not touch the machine learning algorithm chosen by the user. If just making such changes is insufficient for finishing model building by the user-specified deadline, we could also suggest the user to change the algorithm. Since this is a large change that may make the user feel uncomfortable, we should make such suggestions with great caution, possibly as a last resort.

Usually, our estimated cost of building a machine learning model is imprecise. Each time we change the arrangement based on which the model is built, we incur an overhead that is often non-trivial. Thus, we may not always be able to repeatedly change the arrangement to make up the imprecision in cost estimation. To increase the chance that model building can finish by the deadline despite the imprecision, we use a slack factor like s=1.2 and the approach in Ferguson *et al* [16, 31]. When computing potential arrangements, we divide the amount of time available for building the model by the deadline by s. This gives some buffer to address the imprecision.

Typically, CPU is the bottleneck for machine learning model building. If the machine learning software is run on a computer cluster with more than one type of computers, we need to consider the computer heterogeneity in suggesting resource allocation. One way to do this is to pre-construct a mapping between the computing speeds of different types of computers via profiling. Alternatively, we can use more advanced techniques similar to those in Delimitrou *et al* [13].

6. CONCLUSIONS

Machine learning model building and data mining algorithm execution are often time consuming, particularly on large data sets. This paper presents an initial framework for supporting progress indicators for the two tasks, as well as two advanced, potential uses of such progress indicators. Much research is needed to refine the framework, implement progress indicators for various machine learning and data mining algorithms, work out the details of the supporting techniques for the two advanced uses of progress indicators, and investigate additional advanced uses of progress indicators. We hope this paper can provoke future research on this topic. Many techniques described in this paper can be extended to implement progress indicators for long-duration tasks of data pre-processing, feature selection, and running iterative analytics [57], artificial intelligence, natural language processing, combinatorial optimization [30], and numerical optimization [51] algorithms, as well as to support advanced uses of those progress indicators.

Ten years after we initially proposed them [43-45], progress indicators for database queries were put into the commercial product of Microsoft SQL Server database management system [37]. Machine learning model building and data mining algorithm execution have a recursive nature. Consequently, on the same amount of data, they often run several orders of magnitude more slowly and have a greater need for progress indicators than database query execution. Once implementation techniques are fully worked out, we would expect progress indicators for machine learning model building and data mining algorithm execution to provide more value and have faster commercial adoption than progress indicators for database queries.

7. ACKNOWLEDGMENTS

We thank Bev Kerlin for helpful discussions. GL was partially supported by the National Heart, Lung, and Blood Institute of the National Institutes of Health under Award Number R21HL128875. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

8. REFERENCES

[1] A progress bar for scikit-learn? https://stackoverflow.com/questions/34251980/a-progress-bar-for-scikit-learn.

[2] Aggarwal, C.C. Data Mining: The Textbook. New York, NY: Springer 2015.

[3] Agrawal, R., Srikant, R. Fast algorithms for mining association rules in large databases. In: Proc. VLDB, 1994, pp. 487-99.

[4] Alpaydin, E. Introduction to Machine Learning. Cambridge, MA: The MIT Press 2014.

[5] Babich, N. Best practices for animated progress indicators. https://www.smashingmagazine.com/2016/12/best-practices-for-animated-progress-indicators/.

[6] Bekkerman, R., Bilenko, M., Langford, J. Scaling up Machine Learning: Parallel and Distributed Approaches. New York, NY: Cambridge University Press 2011.

[7] Bergstra, J., Bengio, Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research 2012;13:281-305.

[8] Berque, D.A., Goldberg, M.K. Monitoring an algorithm's execution. Computational Support for Discrete Mathematics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 15, 1992:153-63.

[9] Bottou, L., Chapelle, O., DeCoste, D., Weston, J. Large Scale Kernel Machines. Cambridge, MA: MIT Press 2007.

[10] Brazdil, P., Soares, C., da Costa, J.P. Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. Machine Learning 2003;50(3):251-77.

[11] Chaudhuri, S., Narasayya, V.R., Ramamurthy, R. Estimating progress of long running SQL queries. In: Proc. SIGMOD, 2004, pp. 803-14.

[12] Chi, Y., Moon, H.J., Hacigümüs, H., Tatemura, J. SLA-tree: a framework for efficiently supporting SLA-based decisions in cloud computing. In: Proc. EDBT, 2011, pp. 129-40.

[13] Delimitrou, C., Kozyrakis, C. QoS-aware scheduling in heterogeneous datacenters with Paragon. ACM Trans Comput Syst 2013;31(4):12.

[14] Delimitrou, C., Kozyrakis, C. Quasar: resource-efficient and QoS-aware cluster management. In: Proc. ASPLOS, 2014, pp. 127-44.

[15] Doan, T., Kalita, J. Predicting run time of classification algorithms using meta-learning approach. International Journal of Machine Learning and Cybernetics, 2016.

[16] Ferguson, A.D., Bodík, P., Kandula, S., Boutin, E., Fonseca, R. Jockey: guaranteed job latency in data parallel clusters. In: Proc. EuroSys, 2012, pp. 99-112.

[17] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F. Efficient and robust automated machine learning. In: Proc. NIPS, 2015, pp. 2944-52.

[18] Feurer, M., Springenberg, T., Hutter, F. Initializing Bayesian hyperparameter optimization via meta-learning. In: Proc. AAAI, 2015, pp. 1128-35.

[19] Figueroa, R.L., Zeng-Treitler, Q., Kandula, S., Ngo, L.H. Predicting sample size required for classification performance. BMC Med Inform Decis Mak 2012;12:8.

[20] Flajolet, P., Steyaert, J. A complexity calculus for recursive tree algorithms. Mathematical Systems Theory 1987;19(4):301-31.

[21] Gandhi, A., Thota, S., Dube, P., Kochut, A., Zhang, L. Autoscaling for Hadoop clusters. In: Proc. IC2E, 2016, pp. 109-18.

[22] Gupta, C., Mehta, A., Dayal, U. PQR: predicting query execution times for autonomous workload management. In: Proc. ICAC, 2008, pp. 13-22.

[23] Han, J., Pei, J., Yin, Y. Mining frequent patterns without candidate generation. In: Proc. SIGMOD, 2000, pp. 1-12.

[24] Herodotou, H., Dong, F., Babu, S. No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics. In: Proc. SoCC, 2011, 18.

[25] Hickey, T.J., Cohen, J. Automating program analysis. Journal of the ACM 1988;35(1):185-220.

[26] Hickins, M. Citizen data scientists unite! http://www.forbes.com/sites/oracle/2016/10/03/citizen-data-scientists-unite.

[27] Hu, Y., Sundara, S., Srinivasan, J. Supporting time-constrained SQL queries in Oracle. In: Proc. VLDB, 2007, pp. 1207-18.

[28] Huang, B., Boehm, M., Tian, Y., Reinwald, B., Tatikonda, S., Reiss, F.R. Resource elasticity for large-scale machine learning. In: Proc. SIGMOD, 2015, pp. 137-52.

[29] Huang, L., Jia, J., Yu, B., Chun, B., Maniatis, P., Naik, M. Predicting execution time of computer programs using sparse polynomial regression. In: Proc. NIPS, 2010, pp. 883-91.

[30] Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K. Algorithm runtime prediction: methods & evaluation. Artif Intell 2014;206:79-111.

[31] Jalaparti, V., Ballani, H., Costa, P., Karagiannis, T., Rowstron, A. Bridging the tenant-provider gap in cloud services. In: Proc. SoCC, 2012, 10.

[32] Jovic, A., Brkic, K., Bogunovic, N. An overview of free software tools for general data mining. In: Proc. MIPRO, 2014, pp. 1112-7.

[33] Kanter, J.M., Gillespie, O., Veeramachaneni, K. Label, segment, featurize: a cross domain framework for prediction engineering. In: Proc. DSAA, 2016, pp. 430-9.

[34] Kao, B., García-Molina, H. An overview of real-time database systems. In: Proc. NATO ASI RTC, 1992, pp. 261-82.

[35] Keras integration with TQDM progress bars. https://github.com/bstriner/keras-tqdm.
[36] Lam, H.T., Thiebaut, J., Sinn, M., Chen, B., Mai, T., Alkan, O. One button machine for automating feature engineering in relational databases. CoRR abs/1706.00327, 2017.
[37] Lee, K., König, A.C., Narasayya, V.R., Ding, B., Chaudhuri, S., Ellwein, B., Eksarevskiy, A., Kohli, M., Wyant, J., Prakash, P., Nehme, R.V., Li, J., Naughton, J.F. Operator and query progress estimation in Microsoft SQL Server Live Query Statistics. In: Proc. SIGMOD, 2016, pp. 1753-64.
[38] Lee, W., Oh, H., Yi, K. A progress bar for static analyzers. In: Proc. SAS, 2014, pp. 184-200.
[39] Lee, B., Schopf, J.M. Run-time prediction of parallel applications on shared environments. In: Proc. CLUSTER, 2003, pp. 487-91.
[40] Luo, G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw Model Anal Health Inform Bioinform 2016;5:18.
[41] Luo, G. PredicT-ML: a tool for automating machine learning model building with big clinical data. Health Inf Sci Syst 2016;4:5.
[42] Luo, G., Chen, T., Yu, H. Toward a progress indicator for program compilation. Software: Practice and Experience 2007;37(9):909-33.
[43] Luo, G., Naughton, J.F., Ellmann, C.J., Watzke, M. Toward a progress indicator for database queries. In: Proc. SIGMOD, 2004, pp. 791-802.
[44] Luo, G., Naughton, J.F., Ellmann, C.J., Watzke, M. Increasing the accuracy and coverage of SQL progress indicators. In: Proc. ICDE, 2005, pp. 853-64.
[45] Luo, G., Naughton, J.F., Yu, P.S. Multi-query SQL progress indicators. In: Proc. EDBT, 2006, pp. 921-41.
[46] Luo, G., Stone, B.L., Johnson, M.D., Tarczy-Hornoch, P., Wilcox, A.B., Mooney, S.D., Sheng, X., Haug, P.J., Nkoy, F.L. Automating construction of machine learning models with clinical big data: proposal rationale and methods. JMIR Res Protoc 2017;6(8):e175.
[47] Morton, K., Balazinska, M., Grossman, D. ParaTimer: a progress indicator for MapReduce DAGs. In: Proc. SIGMOD, 2010, pp. 507-18.
[48] Morton, K., Friesen, A.L., Balazinska, M., Grossman, D. Estimating the progress of MapReduce pipelines. In: Proc. ICDE, 2010, pp. 681-4.
[49] Myers, B.A. The importance of percent-done progress indicators for computer-human interfaces. In: Proc. SIGCHI, 1985, pp. 11-7.
[50] Nielsen, J. Usability Engineering. San Francisco, CA: Morgan Kaufmann 1993.
[51] Nocedal, J., Wright, S. Numerical Optimization, 2nd ed. New York, NY: Springer 2006.
[52] Ortiz, J., de Almeida, V.T., Balazinska, M. Changing the face of database cloud services with personalized service level agreements. In: Proc. CIDR, 2015.
[53] Ortiz, J., Lee, B., Balazinska, M., Hellerstein, J.L. PerfEnforce: a dynamic scaling engine for analytics with performance guarantees. CoRR abs/1605.09753, 2016.
[54] Pan, X., Venkataraman, S., Tai, Z., Gonzalez, J. Hemingway: modeling distributed optimization algorithms. In: Proc. NIPS Workshop on Machine Learning Systems, 2016.
[55] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É. Scikit-learn: machine learning in Python. Journal of Machine Learning Research 2011;12:2825-30.
[56] Polo, J., Carrera, D., Becerra, Y., Steinder, M., Whalley, I. Performance-driven task co-scheduling for MapReduce environments. In: Proc. NOMS, 2010, pp. 373-80.
[57] Popescu, A.D., Balmin, A., Ercegovac, V., Ailamaki, A. PREDIcT: towards predicting the runtime of large scale iterative analytics. PVLDB 2013;6(14):1678-89.
[58] Practice Fusion diabetes classification homepage. https://www.kaggle.com/c/pf2012-diabetes, 2017.
[59] Priya, R., de Souza, B.F., Rossi, A.L.D., de Carvalho André, C.P.L.F. Predicting execution time of machine learning tasks for scheduling. Int J Hybrid Intell Syst 2013;10(1):23-32.
[60] Priya, R., de Souza, B.F., Rossi, A.L.D., de Carvalho André, C.P.L.F. Using genetic algorithms to improve prediction of execution times of ML tasks. In: Proc. HAIS (1), 2012, pp. 196-207.
[61] Priya, R., de Souza, B.F., Rossi, A.L.D., de Carvalho André, C.P.L.F. Predicting execution time of machine learning tasks using metalearning. In: Proc. WICT, 2011, pp. 1193-8.
[62] Progress bar in random forest model in R. https://stackoverflow.com/questions/32791701/progress-bar-in-random-forest-model-in-r.
[63] Reif, M., Shafait, F., Dengel, A. Prediction of classifier training time including parameter optimization. In: Proc. KI, 2011, pp. 260-71.
[64] Reiner-Benaim, A., Grabarnick, A., Shmueli, E. Highly accurate prediction of jobs runtime classes. International Journal of Advanced Research in Artificial Intelligence 2016;5(6):28-34.
[65] Sarkar, V. Determining average program execution times and their variance. In: Proc. PLDI, 1989, pp. 298-312.
[66] Senger, L.J., Santana, M.J., Santana, R.H.C. An instance-based learning approach for predicting execution times of parallel applications. In: Proc. I2T2S, 2004, pp. 9-15.
[67] Smith, W., Foster, I.T., Taylor, V.E. Predicting application run times with historical information. J Parallel Distrib Comput 2004;64(9):1007-16.
[68] Snoek, J., Larochelle, H., Adams, R.P. Practical Bayesian optimization of machine learning algorithms. In: Proc. NIPS, 2012, pp. 2960-8.
[69] Sparks, E.R., Venkataraman, S., Kaftan, T., Franklin, M.J., Recht, B. KeystoneML: optimizing pipelines for large-scale advanced analytics. In: Proc. ICDE, 2017, pp. 535-46.
[70] Sra, S., Nowozin, S., Wright, S.J. Optimization for Machine Learning. Cambridge, MA: The MIT Press 2011.
[71] Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proc. KDD, 2013, pp. 847-55.
[72] Venkataraman, S., Yang, Z., Franklin, M.J., Recht, B., Stoica, I. Ernest: efficient performance prediction for large-scale advanced analytics. In: Proc. NSDI, 2016, pp. 363-78.
[73] Verma, A., Cherkasova, L., Campbell, R.H. ARIA: automatic resource inference and allocation for MapReduce environments. In: Proc. ICAC, 2011, pp. 235-44.
[74] Wegbreit, B. Mechanical program analysis. Communications of the ACM 1975;18(9):528-39.
[75] Witten, I.H., Frank, E., Hall, M.A., Pal, C.J. Data Mining: Practical Machine Learning Tools and Techniques, 4th ed. Burlington, MA: Morgan Kaufmann 2016.

[76] Wolpert, D.H. The lack of a priori distinctions between learning algorithms. Neural Computation 1996;8(7):1341-90.
[77] Xie, X., Fan, Z., Choi, B., Yi, P., Bhowmick, S.S., Zhou, S. PIGEON: Progress indicator for subgraph queries. In: Proc. ICDE, 2015, pp. 1492-5.
[78] Xiong, P., Chi, Y., Zhu, S., Tatemura, J., Pu, C., Hacigümüs, H. ActiveSLA: a profit-oriented admission control framework for database-as-a-service providers. In: Proc. SoCC, 2011, 15.
[79] Zaki, M.J., Ho, C. Large-Scale Parallel Data Mining. New York, NY: Springer 2000.
[80] Zeng, X., Luo, G. Progressive sampling-based Bayesian optimization for efficient and automatic machine learning model selection. Health Inf Sci Syst 2017;5(1):2.

A Survey on Dialogue Systems: Recent Advances and New Frontiers

Hongshen Chen†, Xiaorui Liu‡, Dawei Yin†, and Jiliang Tang‡
†Data Science Lab, JD.com
‡Data Science and Engineering Lab, Michigan State University
chenhongshen@jd.com, yindawei@acm.org,{xiaorui,tangjili}@msu.edu

ABSTRACT

Dialogue systems have attracted more and more attention. Recent advances on dialogue systems are overwhelmingly contributed by deep learning techniques, which have been employed to enhance a wide range of big data applications such as computer vision, natural language processing, and recommender systems. For dialogue systems, deep learning can leverage a massive amount of data to learn meaningful feature representations and response generation strategies, while requiring a minimum amount of hand-crafting. In this article, we give an overview to these recent advances on dialogue systems from various perspectives and discuss some possible research directions. In particular, we generally divide existing dialogue systems into task-oriented and non-task-oriented models, then detail how deep learning techniques help them with representative algorithms and finally discuss some appealing research directions that can bring the dialogue system research into a new frontier.

1. INTRODUCTION

To have a virtual assistant or a chat companion system with adequate intelligence has seemed illusive, and might only exist in Sci-Fi movies for a long time. Recently, human-computer conversation has attracted increasing attention due to its promising potentials and alluring commercial values. With the development of big data and deep learning techniques, the goal of creating an automatic human-computer conversation system, as our personal assistant or chat companion, is no longer an illusion. On the one hand, nowadays we can easily access "big data" for conversations on the Web and we might be able to learn how to respond and what to respond given (almost) any inputs, which greatly allows us to build data-driven, open-domain conversation systems between humans and computers. On the other hand, deep learning techniques have been proven to be effective in capturing complex patterns in big data and have powered numerous research fields such as computer vision, natural language processing and recommender systems. Hence, a large body of literature has emerged to leverage a massive amount of data via deep learning to advance dialogue systems in many perspectives.

According to the applications, dialogue systems can be roughly categorized into two groups – (1) task-oriented systems and (2) non-task-oriented systems (also known as chat bots). Task-oriented systems aim to assist the user to complete certain tasks (e.g. finding products, and booking accommodations and restaurants). The widely applied approaches to task-oriented systems are to treat the dialogue response as a pipeline as shown in Figure 1. The systems first understand the message given by human, represent it as a internal state, then take some actions according to the policy with respect to the dialogue state, and finally the action is transformed to its surface form as a natural language. Though language understanding is processed by statistical models, most deployed dialogue systems still use manual features or hand crafted rules for the state and action space representations, intent detection, and slot filling. This not only makes it expensive and time-consuming to deploy a real dialogue system, but also limits its usage to other domains. Recently, many algorithms based on deep learning have been developed to alleviate those problems by learning feature representations in a high dimensional distributed fashion and achieve remarkable improvements in these aspects. In addition, there are attempts to build end-to-end task-oriented dialogue systems, which can expand the state space representation in the traditional pipeline systems and help generalize dialogues outside the annotated task-specific corpora.

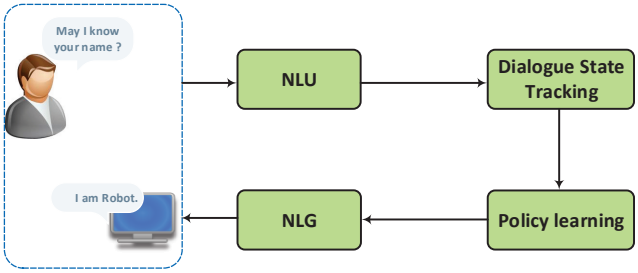


Figure 1: Traditional Pipeline for Task-oriented Systems.

Non-task-oriented systems interact with human to provide reasonable responses and entertainment. Typically, they focus on conversing with human on open domains. Though non-task-oriented systems seem to perform chit-chat, it dominates in many real word applications. As revealed in [98], nearly 80% utterances are chi-chat messages in the online shopping scenario and handling those queries is closely related to user experiences. In general, two major approaches have been developed for non-task-oriented systems – (1) generative methods such as sequence-to-sequence models, which generate proper responses during the conversation; and (2) retrieval-based methods, which learn to select responses from the current conversation from a repository.

Sentence	show	restaurant	at	New	York	tomorrow
Slots	O	O	O	B-desti	I-desti	B-date
Intent	Find Restaurant					
Domain	Order					

Table 1: An Illustrative Example of Natural Language Representation.

The recent development of big data and deep learning techniques has greatly advanced both task-oriented and non-oriented dialogue systems, which has encouraged a huge amount of deep learning based researches in dialogue systems. In this article, we aim to (1) give an overview about dialogue systems especially recent advances from deep learning; and (2) discuss possible research directions. The remaining of the article is organized as follows. We review task-oriented dialogue systems including pipeline and end-to-end methods in Section 2. In Section 3, we first introduce neural generative methods including popular models and hot research topics; and then detail the retrieval-based methods. In Section 4, we conclude the work with discussions on some research directions.

2. TASK-ORIENTED DIALOGUE SYSTEMS

Task-oriented dialogue systems have been an important branch of spoken dialogue systems. In this section, we will review pipeline and end-to-end methods for task-oriented dialogue systems.

2.1 Pipeline Methods

The typical structure of a pipeline based task-oriented dialogue system is demonstrated in Figure 1. It consists of four key components:

- Language understanding. It is known as natural language understanding (NLU), which parses the user utterance into predefined semantic slots.
- Dialogue state tracker. It manages the input of each turn along with the dialogue history and outputs the current dialogue state.
- Dialogue policy learning. It learns the next action based on current dialogue state.
- Natural language generation (NLG). It maps the selected action to its surface and generates the response.

In the following subsections, we will give more details about each component with the state-of-the art algorithms.

2.1.1 Language Understanding

Given an utterance, natural language understanding maps it into semantic slots. The slots are pre-defined according to different scenarios. Table 1 illustrates an example of natural language representation, where "New York" is the location specified as slot values, and the domain and intent are also specified, respectively. Typically, there are two types of representations. One is the utterance level category, such as the user's intent and the utterance category. The other is the word-level information extraction such as named entity recognition and slot filling.

An intent detection is performed to detect the intent of a user. It classifies the utterance into one of the pre-defined intents. Deep learning techniques have been successively applied in intent detection [10; 73; 99]. In particular, [19] used convolutional neural networks (CNN) to extract query vector representations as features for query classification. The CNN-based classification framework also resembled [22] and [64]. Similar approaches are also utilized in category or domain classification.

Slot filling is another challenging problem for spoken language understanding. Unlike intent detection, slot filling is usually defined as a sequence labeling problem, where words in the sentence are assigned with semantic labels. The input is the sentence consisting of a sequence of words, and the output is a sequence of slot/concept IDs, one for each word. [11] and [10] used deep belief networks (DBNs), and achieved superior results compared to CRF baselines. [41; 102; 56; 100] applied RNN for slot filling. The semantic representation generated by NLU is further processed by the dialogue management component. A typical dialogue management component includes two stages – dialogue state tracking and policy learning.

2.1.2 Dialogue State Tracking

Tracking dialogue states is the core component to ensure a robust manner in dialog systems. It estimates the users goal at every turn of the dialogue. A dialogue state H_t denotes the representation of the dialogue session till time t. This classic state structure is commonly called slot filling or semantic frame. The traditional methods, which have been widely used in most commercial implementations, often adopt hand-crafted rules to select the most likely result [17]. However, these rule-based systems are prone to frequent errors as the most likely result is not always the desired one [88].

A statistical dialog system maintains a distribution over multiple hypotheses of the true dialog state, facing with noisy conditions and ambiguity [104]. In Dialog State Tracking Challenge (DSTC) [87; 86], the results are in the form of a probability distribution over each slot for each turn. A variety of statistical approaches, including robust sets of hand-crafted rules [80], conditional random fields [28; 27; 53], maximum entropy models [85] and web-style ranking [88] have emerged in Dialog State Tracking Challenge (DSTC) shared tasks.

Recently, [20] introduced deep learning in belief tracking. It used a sliding window to output a sequence of probability distributions over an arbitrary number of possible values. Though it was trained in one domain, it can be easily transferred to new domains. [48] developed multi-domain RNN dialog state tracking models. It first used all the data available to train a very general belief tracking model, and then specialized the general model for each domain to learn the domain-specific behavior. [49] proposed a neural belief tracker (NBT) to detect the slot-value pairs. It took the system dialogue acts preceding the user input, the user utterance itself, and a single candidate slot-value pair which it needs to make a decision about, as input, and then iterated over all candidate slot-value pairs to determine which ones have just been expressed by the user.

2.1.3 Policy learning

Conditioned on the state representation from the state tracker, the policy learning is to generate the next available system action. Either supervised learning or reinforcement learn-

ing can be used to optimize policy learning [9]. Typically, a rule-based agent is employed to warm-start the system [98]. Then, supervised learning is conducted on the actions generated by the rules. In online shopping scenario, if the dialogue state is "Recommendation", then the "Recommendation" action is triggered, and the system will retrieve products from the product database. If the state is "Comparison", then the system will compare target products/brands[98]. The dialogue policy can be further trained end-to-end with reinforcement learning to lead the system making policies toward the final performance. [9] applied deep reinforcement learning on strategic conversation that simultaneously learned the feature representation and dialogue policy, the system outperformed several baselines including random, rule-based, and supervised-based methods.

2.1.4 Natural Language Generation

The natural language generation component converts an abstract dialogue action into natural language surface utterances. As noticed in [68], a good generator usually relies on several factors: adequacy, fluency, readability, and variation. Conventional approaches to NLG typically perform sentence planning. It maps input semantic symbols into the intermediary form representing the utterance such as tree-like or template structures, and then converts the intermediate structure into the final response through the surface realization [77; 69].

[81] and [82] introduced neural network-based (NN) approaches to NLG with a LSTM-based structure similar with RNNLM [42]. The dialogue act type and its slot-value pairs are transformed into a 1-hot control vector and is given as the additional input, which ensures that the generated utterance represents the intended meaning. [81] used a forward RNN generator together with a CNN reranker, and backward RNN reranker. All the sub-modules are jointly optimized to generate utterances conditioned by the required dialogue act. To address the slot information omitting and duplicating problems in surface realization, [82] used an additional control cell to gate the dialogue act. [72] extended this approach by gating the input token vector of LSTM with the dialogue act. It was then extended to the multi-domain setting by multiple adaptation steps [83]. [110] adopted an encoder-decoder LSTM-based structure to incorporate the question information, semantic slot values, and dialogue act type to generate correct answers. It used the attention mechanism to attend to the key information conditioned on the current decoding state of the decoder. Encoding the dialogue act type embedding, the neural network-based model is able to generate variant answers in response to different act types. [14] also presented a natural language generator based on the sequence-to-sequence approach that can be trained to produce natural language strings as well as deep syntax dependency trees from input dialogue acts. It was then extended with the preceding user utterance and responses [13]. It enabled the model entraining (adapting) to users ways of speaking, which provides contextually appropriate responses.

2.2 End-to-End Methods

Despite a lot of domain-specific handcrafting in traditional task oriented dialogue systems, which are diffcult to adapt to new domains [4], [107] further noted that, the conventional pipeline of task-oriented dialogue systems has two main limitations. One is the credit assignment problem, where the end user's feedback is hard to be propagated to each upstream module. The second issue is process interdependence. The input of a component is dependent on the output of another component. When adapting one component to new environment or retrained with new data, all the other components need to be adapted accordingly to ensure a global optimization. Slots and features might change accordingly. This process requires significant human efforts.

With the advance of end-to-end neural generative models in recent years, many attempts have been made to construct an end-to-end trainable framework for task-oriented dialogue systems. Note that more details about neural generative models will be discussed when we introduce the non-task-oriented systems. Instead of the traditional pipeline, the end-to-end model uses a single module and interacts with structured external databases. [84] and [4] introduced a network-based end-to-end trainable task-oriented dialogue system, which treated dialogue system learning as the problem of learning a mapping from dialogue histories to system responses, and applied an encoder-decoder model to train the whole system. However, the system is trained in a supervised fashion – not only does it require a lot of training data, but it may also fail to find a good policy robustly due to the lack of exploration of dialogue control in the training data. [107] first presented an end-to-end reinforcement learning approach to jointly train dialogue state tracking and policy learning in the dialogue management in order to optimize the system actions more robustly. In the conversation, the agent asks the user a series of Yes/No questions to find the correct answer. This approach was shown to be promising when applied to the task-oriented dialogue problem of guessing the famous people users think of. [36] trained the end-to-end system as a task completion neural dialogue system, where its final goal is to complete a task, such as movie-ticket booking.

Task-oriented systems usually need to query outside knowledge base. Previous systems achieved this by issuing a symbolic query to the knowledge base to retrieve entries based on their attributes, where semantic parsing on the input is performed to construct a symbolic query representing the beliefs of the agent about the user goal[84; 90; 36]. This approach has two drawbacks: (1) the retrieved results do not carry any information about uncertainty in semantic parsing, and (2) the retrieval operation is non differentiable, and hence the parser and dialog policy are trained separately. This makes online end-to-end learning from user feedback difficult once the system is deployed. [15] augmented existing recurrent network architectures with a differentiable attention-based key-value retrieval mechanism over the entries of a knowledge base, which is inspired by key-value memory networks[44]. [12] replaced symbolic queries with an induced "soft" posterior distribution over the knowledge base that indicates which entities the user is interested in. Integrating the soft retrieval process with a reinforcement learner. [89] combined an RNN with domain-specific knowledge encoded as software and system action templates.

3. NON-TASK-ORIENTED DIALOGUE SYSTEMS

Unlike task-oriented dialogue systems, which aim to complete specific tasks for user, non-task-oriented dialogue sys-

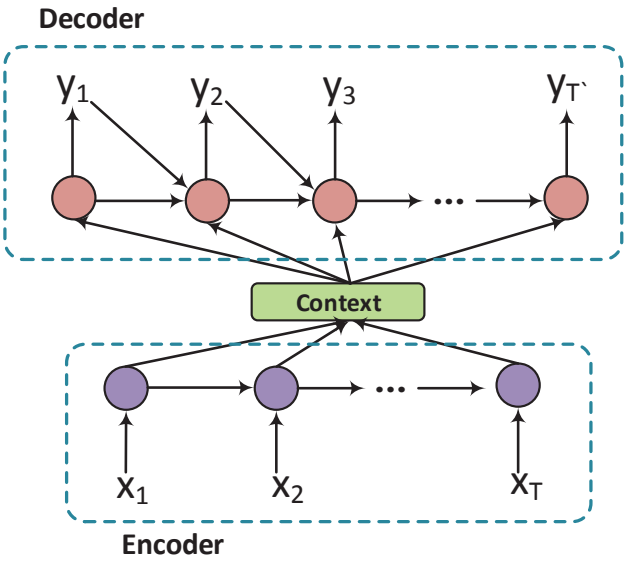


Figure 2: An Illustration of the Encoder-Decoder Model.

tems (also known as chatbots) focus on conversing with human on open domains [54]. In general, chat bots are implemented either by generative methods or retrieval-based methods. Generative models are able to generate more proper responses that could have never appeared in the corpus, while retrieval-based models enjoy the advantage of informative and fluent responses[23], because they select a proper response for the current conversation from a repository with response selection algorithms. In the following sections, we will first dive into the neural generative models, one of the most popular research topics in recent years, and discuss their drawbacks and possible improvements. Then, we introduce recent advances of deep learning in retrieval based models.

3.1 Neural Generative Models

Nowadays, a large amount of conversational exchanges is available in social media websites such as Twitter and Reddit, which raise the prospect of building data-driven models. [54] proposed a generative probabilistic model, which is based on phrase-based Statistical Machine Translation [105], to model conversations on micro-blogging. It viewed the response generation problem as a translation problem, where a post needs to be translated into a response. However, generating responses was found to be considerably more difficult than translating between languages. It is likely due to the wide range of plausible responses and the lack of phrase alignment between the post and the response. The success of applying deep learning in machine translation, namely Neural Machine Translation, spurs the enthusiasm of researches in neural generative dialogue systems.

In the following subsections, we first introduce the sequence-to-sequence models, the foundation of neural generative models. Then, we discuss hot research topics in the direction including incorporating dialogue context, improving the response diversity, modeling topics and personalities, leveraging outside knowledge base, the interactive learning and evaluation.

3.1.1 Sequence-to-Sequence Models

Given a source sequence (*message*) $X = (x_1, x_2, ..., x_T)$ consisting of T words and a target sequence (*response*) $Y = (y_1, y_2, ..., y_{T'})$ of length $T^{'}$,the model maximizes the generation probability of Y conditioned on X: $p(y_1, ..., y_{T'}|x_1, ..., x_T)$. Specifically, a sequence-to-sequence model (or Seq2Seq) is in an encoder-decoder structure. Figure 2 is a general illustration of such structure. The encoder reads X word by word and represents it as a context vector c through a recurrent neural network (RNN),and then the decoder estimates the generation probability of Y with c as the input. The encoder RNN calculates the context vector $\mathbf{c}$ by

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}),$$

where $\mathbf{h}_t$ is the hidden state at time step t, f is a non-linear function such as long-short term memory unit (LSTM) [18] and gated recurrent unit (GRU) [7], and c is the hidden state corresponding to the last word $\mathbf{h}_T$. The decoder is a standard RNN language model with an additional conditional context vector $\mathbf{c}$. The probability distribution $\mathbf{p}_t$ of candidate words at every time t is calculated as

$$\mathbf{s}_t = f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c}),$$
$$\mathbf{p}_t = softmax(\mathbf{s}_t, y_{t-1}),$$

where $\mathbf{s}_t$ is the hidden state of the decoder RNN at time t and y_{t1} is the word at time $t1$ in the response sequence. The objective function of Seq2Seq is defined as:

$$p((y_1, ..., y_{T'}|x_1, ..., x_T) = p(y_1|\mathbf{c}) \prod_{t=2}^{T'} p(y_t|\mathbf{c}, y_1, ..., y_{t-1}).$$

[2] then improved the performance by the attention mechanism, where each word in Y is conditioned on different context vector $\mathbf{c}$, with the observation that each word in Y may relate to different parts in x. In particular, y_i corresponds to a context vector $\mathbf{c}_i$, and $\mathbf{c}_i$ is a weighted average of the encoder hidden states $\mathbf{h}_1, ..., \mathbf{h}_T$:

$$\mathbf{c}_i = \Sigma_{j=1}^{T} \alpha_{ij} \mathbf{h}_j,$$

where $\alpha_{i,j}$ is computed by:

$$\alpha = \frac{exp(e_{ij})}{\Sigma_{k=1}^{T} exp(e_{ik})},$$
$$e_{ij} = g(\mathbf{s}_{t-1}, \mathbf{h}_j),$$

where g is a multilayer perceptron.

[61] applied the recurrent neural network encoder-decoder framework [7] to generate responses on Twitter-style micro-blogging websites, while [74] utilized a similar model described in [40]. In general, these models utilize neural networks to represent dialogue histories and to generate appropriate responses. Such models are able to leverage a large amount of data in order to learn meaningful natural language representations and generation strategies, while requiring a minimum amount of domain knowledge and handcrafting.

3.1.2 Dialogue Context

The ability to take into account previous utterances is key to building dialog systems that can keep conversations active and engaging. [67] addressed the challenge of the context

sensitive response generation by representing the whole dialogue history (including the current message) with continuous representations or embeddings of words and phrases. The response is generated as RNN language model [42], the same as the decoder in [7]. [58] used hierarchical models, first capturing the meaning of individual utterances and then integrating them as discourses. [96] extended the hierarchical structure with the attention mechanism [2] to attend to important parts within and among utterances with word level attention and utterance level attention, respectively. [71] conducted a systematic comparison among existing methods (including non-hierarchical and hierarchical models) and proposed a variant that weights the context with respect to context-query relevance. It found that (1) hierarchical RNNs generally outperform non-hierarchical ones, and (2) with context information, neural networks tend to generate longer, more meaningful and diverse replies.

3.1.3 Response Diversity

A challenging problem in current sequence-to-sequence dialogue systems is that they tend to generate trivial or non-committal, universally relevant responses with little meaning, which are often involving high frequency phrases along the lines of *I dont know* or *Im OK* [67; 74; 58].

This behavior can be ascribed to the relative high frequency of generic responses like *I dont know* in conversational datasets, in contrast with the relative sparsity of more informative alternative responses. One promising approach to alleviate such challenge is to find a better objective function. [30] pointed out that neural models assign high probability to "safe responses when optimizing the likelihood of outputs given inputs. They used a Maximum Mutual Information (MMI), which was first introduced in speech recognition [3; 5], as an optimization objective. It measured the mutual dependence between inputs and outputs, where it took into consideration the inverse dependency of responses on messages. [101] incorporated inverse document frequency (IDF) [55] into the training process to measure the response diversity.

Some researches realized that the decoding process is another source of redundant candidate responses. [62] and [34] recognized that the beam search lacked diversity when generating candidates in the beam. [62] introduced a stochastic beam-search procedure, while [34] added an additional term for beam search scoring to penalize the siblings–expansions of the same parent node in the search. [30; 67; 62] further performed a re-ranking step with global features to avoid generating dull or generic responses. [47] conjectured that not only the problem lies in the objective of decoding and response frequency, but also the message itself may lack sufficient information for the replay. It proposed to use pointwise mutual information (PMI) to predict a noun as a keyword, reflecting the main gist of the reply, and then generates a reply containing the given keyword.

Another series of works have focused on generating more diverse outputs by introducing a stochastic latent variable. They demonstrated that natural dialogue is not deterministic – replies for a same message may vary from person to person. However, current response is sampled from a deterministic encoder-decoder model. By incorporating a latent variable, these models have the advantage that, at the generation time, they can sample a response from the distribution by first sampling an assignment of the latent variables, and then decoding deterministically. [6] presented a latent variable model for one-shot dialogue response. The model contained a stochastic component z in the decoder $P(Y|z, X)$, where z is computed following the variational auto-encoder framework[26; 25; 65]. [59] introduced latent variables to the hierarchical dialogue modeling framework [58]. The latent variable is designed to make high-level decisions like topic or sentiment. [57] extended this method by learning a sequence of the latent variables, where the latent variable of next utterance is only conditioned on the previous latent variable sequence. [63] conditioned the latent variable on explicit attributes to make the latent variable more interpretable. These attributes can be either manually assigned or automatically detected such topics, and personality.

3.1.4 Topic and Personality

Learning the inherent attributes of dialogues explicitly is another way to improve the diversity of dialogues and ensures the consistency. Among different attributes, topic and personality are widely explored.

[95] noticed that people often associate their dialogues with topically related concepts and create their responses according to these concepts. They used Twitter LDA model to get the topic of the input, fed topic information and input representation into a joint attention module and generated a topic-related response. A small improvement in decoder had achieved a better result in [94]. [8] made a more thorough generalization of the problem. They classified each utterance in the dialogue into one domain, and generated the domain and content of next utterance accordingly.

[109] added emotion embedding into a generative model and achieved good performance in perplexity. [51] gave the system an identity with profile so that the system can answer personalized question consistently. [31] further took the information of addressee into consideration to create a more realistic chatbot.

Since the training data comes from different speakers with inconsistency, [106] proposed a two-phase training approach which initialized the model using large scale data and then fine-tuned the model to generate personalized response. [45] used transfer reinforcement learning to eliminate inconsistencies.

3.1.5 Outside Knowledge Base

An important distinction between human conversation and dialogue system is whether it is combined with reality. Incorporating an outside Knowledge Base (KB) is a promising approach to bridge the gap of background knowledge between a dialogue system and human.

Memory network is a classic method dealing with question answering tasks with knowledge base. Thus, it is quite straightforward to apply it in dialogue generation. [16] made attempts on top of this and has achieved good performance in open-domain conversations. [75] also worked on open-domain conversations with background knowledge by coupling CNN embedding and RNN embedding into multimodal space and made progress in perplexity. A similar task is to generate an answer for a question according to outside knowledge. Unlike the general method of tuple retrieval in knowledge base, [103] used words from knowledge base together with common words in generation process. Empirical studies demonstrated that the proposed model was capable of generating natural and right answers to the questions by

referring to the facts in the knowledge base.

3.1.6 Interactive Dialogue learning

Learning through interaction is one of the ultimate goals of dialogue systems. [35] simulated dialogues between two virtual agents. They defined simple heuristic approximations to rewards that characterize good conversations: good conversations are forward-looking [1] or interactive (a turn suggests a following turn), informative, and coherent. The parameters of an encoder-decoder RNN defined a policy over an infinite action space consisting of all possible utterances. The agent learned a policy by optimizing the long-term developer-defined reward from ongoing dialogue simulations using policy gradient methods [91], rather than the MLE objective defined in the standard SEQ2SEQ models. [33] further attempted to improve the bot's ability to learn from interaction. By using policy learning and forward prediction on both textual and numerical feedback, the model can improve itself by interacting with human in a (semi-)online way.

As most human respondents may ask for clarification or hints when not confident about the answer, it is natural to make the bot owning such a capability. [32] defined three situations where the bot has problems in answering a question. Compared the experimental results of not using a asking-question way, this method made great improvement in some scenarios. [29] explored the task on negotiation dialogues. As conventional sequence-to-sequence models simulate human dialogues but fail to optimize a specific goal, this work took a goal-oriented training and decoding approach and demonstrated a worthwhile perspective.

3.1.7 Evaluation

Evaluating the quality of the generated response is an important aspect of dialogue response generation systems[37]. Task-oriented dialogue system can be evaluated based on human-generated supervised signals, such as a task completion test or a user satisfaction score[76; 46; 24], however, automatically evaluating the quality of generated responses for non-task-oriented dialogue systems remains an open question. Despite the fact that word overlap metrics such as BLEU, METEOR, and ROUGE have been widely used to evaluate the generated responses, [37] found that those metrics, as well as word embedding metrics derived from word embedding models such as Word2Vec[43] have either weak or no correlation with human judgements, although word embedding metrics are able to significantly distinguish between baselines and state-of-the-art models across multiple datasets. [70] proposed to use two neural network models to evaluate a sequence of turn-level features to rate the success of a dialogue.

3.2 Retrieval-based Methods

Retrieval-based methods choose a response from candidate responses. The key to retrieval-based methods is message-response matching. Matching algorithms have to overcome semantic gaps between messages and responses [21].

3.2.1 Single-turn Response Matching

Early studies of retrieval-based chatbots mainly focus on response selection for single-turn conversation[78], where only the message is used to select a proper response. Typically, the context and the candidate response are encoded as a vector respectively, and then the matching score is computed based on those two vectors. Suppose $\mathbf{x}$ is the vector representation of a message and $\mathbf{y}$ corresponds to the response vector representation, the matching function between $\mathbf{x}$ and $\mathbf{y}$ can be as simply as bilinear matching:

$$match(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{y},$$

where $\mathbf{A}$ is a pre-determined matrix, or more complicated ones. [39] proposed a DNN-based matching model for short texts response selection and combined the localness and hierarchy intrinsic in the structure. [21] improved the model by utilizing a deep convolutional neural network architecture to learn the representation of message and response, or directly learn the interacted representation of two sentences, followed by a multi-layer perceptron to compute the matching score. [79] extracted dependency tree matching patterns and used them as sparse one-hot inputs of a deep feed-forward neural network for context-response matching. [92] incorporated the topic vector generated by Twitter LDA model [108] into the CNN based structure to boost responses with rich content.

3.2.2 Multi-turn Response Matching

In recent years, multi-turn retrieval-based conversation draws more and more attention. In multi-turn response selection, current message and previous utterances are taken as input. The model selects a response that is natural and relevant to the whole context. It is important to identify important information in previous utterances and properly model the utterances relationships to ensure conversation consistency. [38] encoded the context (a concatenation of all previous utterances and current message) and candidate response into a context vector and a response vector through a RNN/LSTM based structure, respectively, and then computed the matching degree score based on those two vectors. [97] selected the previous utterances in different strategies and combined them with current messages to form a reformulated context. [111] performed context-response matching on not only the general word level context vector but also the utterance level context vector. [93] further improved the leveraging of utterances relationship and contextual information by matching a response with each utterance in the context on multiple levels of granularity with a convolutional neural network, and then accumulated the vectors in a chronological order through a recurrent neural network to model relationships among utterances.

3.3 Hybrid Methods

Combing neural generative and retrieval based models can have significant effects on performance.[66] and [52] attempted to combine both methods. Retrieval-based systems often give precise but blunt answers, while generation-based systems tend to give fluent but meaningless responses. In an ensemble model, the retrieved candidate, along with the original message, are fed to an RNN-based response generator. The final response is given by a post-reranker. This approach combined the advantages of retrieval and generation based models, which was appealing in performance. [60] integrated natural language generation and retrieval models, including template-based models, bag-of-words models, sequence-to-sequence neural network and latent variable neural network models and applied reinforcement learning to crowdsourced data and real-world user interactions to select

an appropriate response from the models in its ensemble.

4. DISCUSSION AND CONCLUSION

Deep learning has become a basic technique in dialogue systems. Researchers investigated on applying neural networks to the different components of a traditional task-oriented dialogue system, including natural language understanding, natural language generation, dialogue state tracking. Recent years, end-to-end frameworks become popular in not only the non-task-oriented chit-chat dialogue systems, but also the task-oriented ones. Deep learning is capable of leveraging large amount of data and is promising to build up a unified intelligent dialogue system. It is blurring the boundaries between the task-oriented dialogue systems and non-task-oriented systems. In particular, the chit-chat dialogues are modeled by the sequence-to-sequence model directly. The task completion models are also moving towards an end-to-end trainable style with reinforcement learning representing the state-action space and combing the whole pipelines.

It is worth noting that current end-to-end models are still far from perfect. Despite the aforementioned achievements, the problems remain challenging. Next, we discuss some possible research directions:

- Swift Warm-Up. Although end-to-end models have drawn most of the recent research attention, we still need to rely on traditional pipelines in practical dialogue engineering, especially in a new domain warm-up stage. The daily conversation data is quite "big", however, the dialogue data for a specific domain is quite limited. In particular, domain specific dialogue data collection and dialogue system construction are laborsome. Neural network based models are better at leveraging large amount of data. We need new way to bridge over the warm-up stage. It is promising that the dialogue agent has the ability to learn by itself from the interactions with human.

- Deep Understanding. Current neural network based dialogue systems heavily rely on the huge amount of different types of annotated data, and structured knowledge base and conversation data. They learn to speak by imitating a response again and again, just like an infant, and the responses are still lack of diversity and sometimes are not meaningful. Hence, the dialogue agent should be able to learn more effectively with a deep understanding of the language and the real world. Specifically, it remains much potential if a dialogue agent can learn from human instruction to get rid of repeatedly training. Since a great quantity of knowledge is available on the Internet, a dialogue agent can be smarter if it is capable of utilizing such unstructured knowledge resource to make comprehension. Last but not least, a dialogue agent should be able to make reasonable inference, find something new, share its knowledge across domains, instead of repeating the words like a parrot.

- Privacy Protection. Widely applied dialogue system serves a large number of people. It is quite necessary to notice the fact that we are using the same dialogue assistant. With the ability of learning through interactions, comprehension and inference, a dialogue assistant can inadvertently and implicitly store some of sensitive information [50]. Hence, it is important to protect users' privacy while building better dialogue systems.

Acknowledgements

Xiaorui Liu and Jiliang Tang are supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940.

5. REFERENCES

[1] J. Allwood, J. Nivre, and E. Ahlsén. On the semantics and pragmatics of linguistic feedback. *Journal of semantics*, 9(1):1–26, 1992.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] L. Bahl, P. Brown, P. De Souza, and R. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 49–52. IEEE, 1986.

[4] A. Bordes, Y. L. Boureau, and J. Weston. Learning end-to-end goal-oriented dialog. 2017.

[5] P. F. Brown. The acoustic-modeling problem in automatic speech recognition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1987.

[6] K. Cao and S. Clark. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 182–187, Valencia, Spain, April 2017. Association for Computational Linguistics.

[7] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] S. Choudhary, P. Srivastava, L. Ungar, and J. Sedoc. Domain aware neural dialog system. 2017.

[9] H. Cuayhuitl, S. Keizer, and O. Lemon. Strategic dialogue management via deep reinforcement learning. *arxiv.org*, 2015.

[10] L. Deng, G. Tur, X. He, and D. Hakkani-Tur. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 210–215. IEEE, 2012.

[11] A. Deoras and R. Sarikaya. Deep belief network based semantic taggers for spoken language understanding. 2013.

[12] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng. Towards end-to-end reinforce-

ment learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[13] O. Dušek and F. Jurcicek. A context-aware natural language generator for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 185–190, Los Angeles, September 2016. Association for Computational Linguistics.

[14] O. Dušek and F. Jurcicek. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany, August 2016. Association for Computational Linguistics.

[15] M. Eric and C. D. Manning. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*, 2017.

[16] M. Ghazvininejad, C. Brockett, M. W. Chang, B. Dolan, J. Gao, W. Yih, and M. Galley. A knowledge-grounded neural conversation model. 2017.

[17] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. A form-based dialogue manager for spoken language applications. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 701–704. IEEE, 1996.

[18] A. Graves. Long short-term memory. *Neural Computation*, 9(8):1735, 1997.

[19] H. B. Hashemi, A. Asiaee, and R. Kraft. Query intent detection using convolutional neural networks.

[20] M. Henderson, B. Thomson, and S. Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, 2013.

[21] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[22] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[23] Z. Ji, Z. Lu, and H. Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*, 2014.

[24] C. Kamm. User interfaces for voice applications. *Proceedings of the National Academy of Sciences*, 92(22):10031–10037, 1995.

[25] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, 4:3581–3589, 2014.

[26] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[27] S. Lee. Structured discriminative model for dialog state tracking. In *SIGDIAL Conference*, pages 442–451, 2013.

[28] S. Lee and M. Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *SIGDIAL Conference*, pages 414–422, 2013.

[29] M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2433–2443, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[30] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, June 2016. Association for Computational Linguistics.

[31] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany, August 2016. Association for Computational Linguistics.

[32] J. Li, A. H. Miller, S. Chopra, M. Ranzato, and J. Weston. Learning through dialogue interactions by asking questions. *arXiv preprint*.

[33] J. Li, A. H. Miller, S. Chopra, M. Ranzato, and J. Weston. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*, 2016.

[34] J. Li, W. Monroe, and J. Dan. A simple, fast diverse decoding algorithm for neural generation. 2016.

[35] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas, November 2016. Association for Computational Linguistics.

[36] X. Li, Y.-N. Chen, L. Li, and J. Gao. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

[37] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas, November 2016. Association for Computational Linguistics.

[38] R. Lowe, N. Pow, I. Serban, and J. Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–

294, Prague, Czech Republic, September 2015. Association for Computational Linguistics.

[39] Z. Lu and H. Li. A deep architecture for matching short texts. In *International Conference on Neural Information Processing Systems*, pages 1367–1375, 2013.

[40] T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics.

[41] G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. *Interspeech*, 2013.

[42] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[43] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[44] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas, November 2016. Association for Computational Linguistics.

[45] K. Mo, S. Li, Y. Zhang, J. Li, and Q. Yang. Personalizing a dialogue system with transfer reinforcement learning. 2016.

[46] S. Möller, R. Englert, K. Engelbrecht, V. Hafner, A. Jameson, A. Oulasvirta, A. Raake, and N. Reithinger. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *Ninth International Conference on Spoken Language Processing*, 2006.

[47] L. Mou, Y. Song, R. Yan, G. Li, L. Zhang, and Z. Jin. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[48] N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gasic, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 794–799, Beijing, China, July 2015. Association for Computational Linguistics.

[49] N. Mrkšić, D. Ó Séaghdha, T.-H. Wen, B. Thomson, and S. Young. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[50] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *ICLR*, 2017.

[51] Q. Qian, M. Huang, H. Zhao, J. Xu, and X. Zhu. Assigning personality/identity to a chatting machine for coherent conversation generation. 2017.

[52] M. Qiu, F.-L. Li, S. Wang, X. Gao, Y. Chen, W. Zhao, H. Chen, J. Huang, and W. Chu. Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 498–503, 2017.

[53] H. Ren, W. Xu, Y. Zhang, and Y. Yan. Dialog state tracking using conditional random fields. In *SIGDIAL Conference*, pages 457–461, 2013.

[54] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *Conference on Empirical Methods in Natural Language Processing*, pages 583–593, 2011.

[55] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[56] R. Sarikaya, G. E. Hinton, and B. Ramabhadran. Deep belief nets for natural language call-routing. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5680–5683, 2011.

[57] I. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. Courville. Multiresolution recurrent neural networks: An application to dialogue response generation. 2017.

[58] I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models, 2016.

[59] I. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. 2017.

[60] I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

[61] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China, July 2015. Association for Computational Linguistics.

[62] L. Shao, S. Gouws, D. Britz, A. Goldie, and B. Strope. Generating long and diverse responses with neural conversation models. 2017.

[63] X. Shen, H. Su, Y. Li, W. Li, S. Niu, Y. Zhao, A. Aizawa, and G. Long. A conditional variational

framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 504–509, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[64] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM, 2014.

[65] K. Sohn, X. Yan, and H. Lee. Learning structured output representation using deep conditional generative models. In *International Conference on Neural Information Processing Systems*, pages 3483–3491, 2015.

[66] Y. Song, R. Yan, X. Li, D. Zhao, and M. Zhang. Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*, 2016.

[67] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[68] A. Stent, M. Marge, and M. Singhai. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 341–351, 2005.

[69] A. Stent, R. Prasad, and M. Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics, 2004.

[70] P.-H. Su, D. Vandyke, M. Gasic, D. Kim, N. Mrksic, T.-H. Wen, and S. Young. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. *arXiv preprint arXiv:1508.03386*, 2015.

[71] Z. Tian, R. Yan, L. Mou, Y. Song, Y. Feng, and D. Zhao. How to make context more useful? an empirical study on context-aware neural conversational models. In *Meeting of the Association for Computational Linguistics*, pages 231–236, 2017.

[72] V. K. Tran and L. M. Nguyen. Semantic refinement gru-based neural language generation for spoken dialogue systems. In *PACLING*, 2017.

[73] G. Tur, L. Deng, D. Hakkani-Tür, and X. He. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5045–5048. IEEE, 2012.

[74] O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[75] P. Vougiouklis, J. Hare, and E. Simperl. A neural network approach for knowledge-driven response generation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3370–3380, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[76] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. Association for Computational Linguistics, 1997.

[77] M. A. Walker, O. C. Rambow, and M. Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409–433, 2002.

[78] H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[79] M. Wang, Z. Lu, H. Li, and Q. Liu. Syntax-based deep matching of short texts. 03 2015.

[80] Z. Wang and O. Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL Conference*, pages 423–432, 2013.

[81] T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284, Prague, Czech Republic, September 2015. Association for Computational Linguistics.

[82] T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[83] T.-H. Wen, M. Gašić, N. Mrkšić, L. M. Rojas-Barahona, P.-H. Su, D. Vandyke, and S. Young. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129, San Diego, California, June 2016. Association for Computational Linguistics.

[84] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gasic, L. M. Rojas Barahona, P.-H. Su, S. Ultes, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain, April 2017. Association for Computational Linguistics.

[85] J. Williams. Multi-domain learning and generalization in dialog state tracking. In *SIGDIAL Conference*, pages 433–441, 2013.

[86] J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.

[87] J. D. Williams. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 23–24, 2012.

[88] J. D. Williams. Web-style ranking and slu combination for dialog state tracking. In *SIGDIAL Conference*, pages 282–291, 2014.

[89] J. D. Williams, K. Asadi, and G. Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[90] J. D. Williams and G. Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*, 2016.

[91] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[92] Y. Wu, W. Wu, Z. Li, and M. Zhou. Topic augmented neural network for short text conversation. 2016.

[93] Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[94] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W. Y. Ma. Topic augmented neural response generation with a joint attention mechanism. 2016.

[95] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W.-Y. Ma. Topic aware neural response generation. 2017.

[96] C. Xing, W. Wu, Y. Wu, M. Zhou, Y. Huang, and W. Y. Ma. Hierarchical recurrent attention network for response generation. 2017.

[97] R. Yan, Y. Song, and H. Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 55–64, New York, NY, USA, 2016. ACM.

[98] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li. Building task-oriented dialogue systems for online shopping, 2017.

[99] D. Yann, G. Tur, D. Hakkani-Tur, and L. Heck. Zero-shot learning and clustering for semantic utterance classification using deep learning, 2014.

[100] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi. Spoken language understanding using long short-term memory neural networks. pages 189 – 194, 2014.

[101] K. Yao, B. Peng, G. Zweig, and K. F. Wong. An attentional neural conversation model with improved specificity. 2016.

[102] K. Yao, G. Zweig, M. Y. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *Interspeech*, 2013.

[103] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2972–2978. AAAI Press, 2016.

[104] S. Young, M. Gai, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. 24(2):150–174, 2010.

[105] R. Zens, F. J. Och, and H. Ney. Phrase-based statistical machine translation. In *German Conference on Ai: Advances in Artificial Intelligence*, pages 18–32, 2002.

[106] W. Zhang, T. Liu, Y. Wang, and Q. Zhu. Neural personalized response generation as domain adaptation. 2017.

[107] T. Zhao and M. Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, Los Angeles, September 2016. Association for Computational Linguistics.

[108] W. X. Zhao, J. Jiang, J. Weng, J. He, E. P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *European Conference on Advances in Information Retrieval*, pages 338–349, 2011.

[109] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. 2017.

[110] H. Zhou, M. Huang, and X. Zhu. Context-aware natural language generation for spoken dialogue systems. In *COLING*, pages 2032–2041, 2016.

[111] X. Zhou, D. Dong, H. Wu, S. Zhao, D. Yu, H. Tian, X. Liu, and R. Yan. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381, 2016.

Delve: A Dataset-Driven Scholarly Search and Analysis System

Uchenna Akujuobi, Xiangliang Zhang
Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology (KAUST)
Saudi Arabia
{uchenna.akujuobi,xiangliang.zhang}@kaust.edu.sa

ABSTRACT

Research and experimentation in various scientific fields are based on the observation, analysis and benchmarking on datasets. The advancement of research and development has thus, strengthened the importance of dataset access. However, without enough knowledge of relevant datasets, researchers usually have to go through a process which we term "manual dataset retrieval". With the accelerated rate of scholarly publications, manually finding the relevant dataset for a given research area based on its usage or popularity is increasingly becoming more and more difficult and tedious. In this paper, we present Delve, a web-based dataset retrieval and document analysis system. Unlike traditional academic search engines and dataset repositories, Delve is dataset driven and provides a medium for dataset retrieval based on the suitability or usage in a given field. It also visualizes dataset and document citation relationship, and enables users to analyze a scientific document by uploading its full PDF. In this paper, we first discuss the reasons why the scientific community needs a system like Delve. We then proceed to introduce its internal design and explain how Delve works and how it is beneficial to researchers of all levels.

1. INTRODUCTION

The word "Data" according to the Webster's English dictionary [35], is defined as "a collection of facts, observations, or other information related to a particular question or problem". Based on the above definition, data (physical or digital) can be attributed to being a "cornerstone" of various scientific researches which have led to the advancement of science and technology. In various scientific fields, the research process involves exploration, analysis and evaluation based on a set of **data**. For instance, various computer science fields (machine learning, data mining, database, computer vision, pattern recognition, etc.) usually evaluate the effectiveness of a proposed approach based on experiments conducted on a set of benchmark datasets. In several other scientific fields like the environmental and biological sciences, the credibility of proposed models designed from data and available knowledge in consort with end-users and simulations is usually critically analyzed and reviewed based on the model's performance on a particular range of data spectrum [22; 16; 5; 2].

The amelioration of science and technology has made it possible not just to approach problems that could have never been solvable in the past but to also improve upon the performance of previous methods. Data is essential to both cases. However, scholarly search based on dataset usage even when familiar with the research field might require a significant amount of time and effort due to the unprecedented rate of scholarly publications [23]. For example, consider the query: "Find all papers using the MOA datasets and working on relational learning". Typically, a two-step manual method for answering this query is to 1) query the academic search engines for papers on relational learning, and 2) spend a lot of time reading through the searched papers to filter out the papers using the MOA datasets. This process can be quite tedious and becomes more complex as a paper "A" might specify it used the same dataset as in paper "B", in which case a researcher also needs to search and go through paper "B" to determine what dataset was used. Another relevant query can be: "Find popular datasets used in relational learning". It will also require a vast amount of time to be dedicated to reading many articles.

It is vital to have a data-driven search engine to exploit the rich semantics of dataset information available in academic documents, which current scholarly search engines fail to provide. With the availability of different academic web search engines and databases (e.g., Google scholar[1], Microsoft Academic[2], Semantic scholar[3]), information on papers or authors in different fields, topics, can be easily accessed. Also, dataset portals and repositories like Opendata[4] and UCI repository [21] provide a medium where users can search for datasets. However, having these two systems independently, even though each individually performs its respective functions, offers only a meager benefaction in finding relevant papers working on a given dataset or finding relevant datasets for a given problem. To the best of our knowledge, only one academic search engine[3] currently integrates the use of dataset in academic document search. It uses dataset as a filter medium to their search results, rather than allowing datasets as a search query, i.e., not answering a simple query like "Find all the papers using MOA dataset".

In this paper, we present Delve, an online dataset-driven system that provides a medium for dataset or document search, visual analysis of the citation relations among documents and datasets, and online document analysis. More

[1]https://scholar.google.com
[2]https://academic.microsoft.com
[3]https://www.semanticscholar.org
[4]https://data.opendatasoft.com

specifically, Delve offers users a simple and easy-to-use interface for

- Finding a set of benchmark datasets for a research topic/field interest;
- Finding a set of research papers that used the same datasets;
- Visually analyzing the citation relations of academic documents and datasets;
- Instantly online analyzing an academic document and showing its citation relations w.r.t. other documents and datasets, when a PDF of the document is provided.

With these above-mentioned features, Delve is useful for different purposes, e.g., finding relevant papers for literature review, finding appropriate datasets for a specific research interest, understanding document and dataset citation relationships.

The rest of the paper is organized as follows. In section 2, we discuss the importance of data availability and access for scientific research. Section 3 presents a brief overview of the previous works done in the area of document and dataset search. We explicitly introduce the Delve system in section 4 and show how the Delve system works in section 5 by presenting some use case scenarios. We then provide some possible extension of the system in section 6 and conclude in section 7.

2. DATA ACCESS IN SCIENCE

Ancient civilizations like the Egyptians, Babylonians, Indians, and Chinese; practiced what many could refer to today as applied science and mathematics [15]. Using the knowledge obtained from recording and studying the stars and heavenly bodies, they were able to predict seasons and develop principles of direction that they then applied to agriculture and navigation. The availability of the recorded data about the stars and heavenly bodies played a significant role in the study of seasons and navigation. Data have always been a driving force in the evolution of science and technology. With the current progression of science, the crucial need for data becomes more and more pronounced. In this section, we will discuss the importance of dataset access and analysis in scientific research.

2.1 Empirical Evaluations

Empirical evaluations are one of the fundamental procedures in scientific research, for validating and analyzing the performance of different methods. Usually, the evaluations provide a comparison showing which method is superior in a given problem setting [12]. As commonly noted by several authors [12; 8; 28; 7], evaluations can sometimes be seriously misleading, and need to be made in a fair and objective way. Also, as noted by Keogh [19], most empirical evaluations are data biased because the choice of dataset has a substantial effect on the results reported in many scholarly papers. For instance, let us consider a researcher with a prior research interest in Natural Language Processing (NLP) who might be interested in developing a new method or extending a previous method in a new research area of interest (e.g., *image annotation*). To show the performance of her method, the researcher would need to compare the performance of her method with that of some prior methods on the different datasets used by the prior works. Without the availability and prior knowledge of such datasets, a fair and objective comparison cannot be made. Easy access to information about datasets and how they have been used will reduce the data biases in empirical evaluations and curtail the dilemma of choosing the wrong datasets for this step of scientific research. This information would improve the quality and validity of empirical evaluations and increase the efficiency of researchers.

2.2 Reproducibility and Data Analysis

A scientific work needs to be repeatable given the same procedure, parameters, and data. Reproducibility makes a research work easier to understand by other researchers both to verify the reported results or to extend the work. However, the lack of reproducibility has continued to be a significant problem in science, and several authors [14; 19; 27] have warned against it. The availability of dataset used in a scientific work is quite crucial for the reproducibility of the work, as methods perform differently with different datasets [12]. Providing information and easy access to datasets used in various scientific research works would enhance the reproducibility of these works, and thus enable an objective analysis and validation of research works, for promoting the quality of scientific research.

Another advantage of providing access to dataset information is that the value of data can be better explored by more researchers. This data exploration could lead to further insights and observations, bringing about more knowledge discoveries from the dataset by using it for different purposes, which might not be the initial intention of gathering the dataset. A valid example of this, as mentioned by Vanschoren et al. [33], is the Sloan Digital Sky Survey (SDSS) data. The SDSS project commissioned to take spectra and images of about 35% of the night sky has so far created the most comprehensive astrophysical catalog in the world [36]. This collected data, which was initially confined only to the members of the project, was opened up to the public [30] and has since been used in different research studies. Due to the availability of the data, scientists were able to ask different questions from the dataset [29; 24; 25; 13; 4], leading to a vast number of discoveries. An example of a significant discovery from the SDSS data is the discovery of the emission light galaxy known as "Green peas" via the Galaxy Zoo project. The Galaxy Zoo project employs the help of astronomy enthusiasts to classify millions of galaxies in data obtained from different sources including the SDSS. Volunteers studying the SDSS data provided by the Galaxy Zoo project discovered the emission light galaxy by noting their peculiarity which was then unresolved in Sloan Digital Sky Survey imaging [4].

3. PREVIOUS WORK

Over the years, the importance of access to scholarly documents and datasets has been increasingly recognized by researchers. There have been works directed towards making information to scholarly documents and datasets easily accessible. However, these efforts have mostly been disjoint.

3.1 Scholarly Search Engines

Citeseer[5], an open-source scholarly search engine, was introduced in 1997 by Giles et al. [10] as an automatic citation indexing system. Citeseer later became CiteseerX in 2007, which is a scholarly search engine, digital library, and repository for scientific and academic papers with a focus on scholarly papers in computer science [1]. Citeseer is considered to be the first academic search engine and only indexes publicly available documents. In 2000, Scirus[6] was launched as a joint work between FAST, a Norwegian search engine company, and the Elsevier Science publishing group to address the problem of access to scholarly documents from both authoritative sources like publishers and non-authoritative sources like university websites. Scirus has since been retired in 2014 and replaced with Scopus[7].

Two of the more recent scholarly search engines are Google Scholar[1] and Microsoft Academic[2]. Google scholar was initially launched in 2004 as a way to improve the efficiency of researchers by providing access to scholarly literature information and knowledge [20]. Over the years more features have been added to the search engine including saving search results and organizing author citations. Microsoft Academic was initially introduced as Windows Live Academic Search in 2006 to compete with Google's scholarly search engine, and then was retired after two years. In 2016, Microsoft Academic, a relaunch of Microsoft Academic Search, was introduced as a free public scholarly search engine, which essentially has the same function as other scholarly search engines.

Some systems extend the idea of academic search engines by applying machine learning techniques on the academic documents to retrieve other information from the scholarly materials. AMiner[8] (former Arnetminer) was created in 2006 to search and analyze researcher networks [31]. In 2015, Semantic Scholar[3] was created to provide a smart search service for journals by applying some machine learning techniques and a layer of semantic analysis. Semantic Scholar incorporates the use of datasets as a filter parameter in generating their results. Currently, there is a considerable amount of scholarly search engines available on the web, each with its features. However, none of the currently available academic search engines is dataset driven.

3.2 Dataset Repositories and Portals

The creation of standard collections of datasets has made the reproduction and empirical evaluations of scientific work easier and fair [12]. There are a lot of dataset repositories and data portals currently available. Some of these like the UCI repository [21], KDD achive[9], Mldata[10], OpenDataSoft[4], Data.Gov[11], SDSS[12] and LDC[13] are openly accessible to the public. There are also the commercial dataset collections including Datamarket[14], Xignite[15], and IEEEDataPort[16]. With the increasing advocacy of open data, more and more closed datasets are being made public. Open data has been shown to benefit both the academic community and the data owner [33; 14; 12; 17].

Vast number of dataset repositories and data portals mean more available datasets to use, but also mean more difficulties for researchers to find appropriate datasets and relevant references. It is often that researchers end up using datasets they have heard or read about, even though there might be better datasets available and more suitable for their research problem. Having a platform possessing information on datasets from different dataset repository and data portals, ranking them by relevancy to a search keyword or phrase, and providing the relevant datasets to researchers will not only provide researchers with better dataset choices but also provide exposure to various good dataset repositories and data portals.

4. DELVE

Delve[17] is a dataset-driven system with a focus on dataset retrieval and document analysis [3]. The advanced features Delve has over other scholarly search engines are

- Delve can be used to **retrieve dataset-driven results**. For example, in Delve homepage (shown in Figure 8a), a user can give a query word, which can be a dataset name or a research topic, e.g., *image/video annotaiton*. Matched datasets will be returned and ranked by their relevance (shown in the middle part of Figure 6), as well as relevant research documents (given in the right part of Figure 6). This feature is very useful for finding relevant datasets and surveying relevant research documents.
- Delve can be used to **understand the relationship** between papers and dataset. This relationship can be paper-to-paper, paper-to-dataset, or dataset-to-dataset. For example, the graph in Figure 7b and Figure 8b show the citation relationship among papers and datasets. The visualization of the relationship among papers and datasets can help on easily getting more insights about a paper or dataset.
- Delve can **analyze PDFs** of academic documents. This feature can be used in analyzing a document even before submitting it to a journal or conference to evaluate its relationship to other published papers. With this, a researcher will be able to detect a paper that might be of advisable to cite or read through. Figure 8 demonstrates one example of this feature.

Detailed works behind each feature will be discussed in next subsections. At the time of writing this paper, we have to admit some disadvantages of Delve:

- The size of the Delve database (currently including 2 million scholarly documents) is limited when compared with the database of other popularly used academic search engines. This limitation would be less

[5]http://citeseerx.ist.psu.edu
[6]http://www.scirus.com
[7]https://www.scopus.com
[8]https://aminer.org
[9]https://kdd.ics.uci.edu
[10]http://mldata.org
[11]https://www.data.gov
[12]http://www.sdss.org
[13]http://www.ldc.upenn.edu
[14]http://www.qlik.com/us/products/qlik-data-market
[15]http://www.xignite.com/
[16]https://ieee-dataport.org/
[17]https://delve.kaust.edu.sa

of an issue as the Delve database will be continuously expanded with more collections.

- There are false citation relationships displayed on Delve. This issue is due to the limited document collection size and the simple algorithm Delve currently uses for citation relationship prediction. We expect this to be curtailed with improvements to the applied algorithms and the addition of more document collections.

4.1 The Delve Database

The Delve database was initialized by collecting papers published in 17 different conferences and journals between 2001 to 2016, including AAAI, IJCAL, TKDD, NIPS, CIKM, VLDB, ICML, ICDM, PKDD, WSDM, SDM, ICDE, KDD, DMKD, KAIS, WWW, and TKDE. Using the Microsoft graph dataset[18], the Delve database was extended to include references and the references of their references (up to 2 hops away) of the papers in the initially selected conferences and journals. This extension thus enlarged our database. At the time of writing this paper, the Delve database includes more than 2 million scholarly documents from more than 1000 different sources including conferences, journals, and books.

Documents and datasets are treated as nodes in Delve. A large citation graph is then built by linking papers and papers, papers and datasets, datasets and datasets if there exist citation/usage relations among them. For supporting dataset-driven search, Delve explores not only the **node content** (text of scholarly documents or datasets), but also the **edge labels** (positive labels indicating the dataset relevance, e.g., paper A used dataset D, or paper A citing paper B because of the common datasets they used). The initial labeling work was conducted by crowd-sourcing on papers and datasets cited by these papers published in ICDE, KDD, ICDM, SDM, and TKDE from 2001 to 2014. These labels (accounting for 5% of the whole graph edges) have been manually verified to be correct by three qualified participants. Due to the high cost of crowd-sourcing, it is infeasible to label the remaining 95% of edges manually. Therefore, one of the principal challenges that arise in Delve is to develop an efficient and effective method to assign labels to a large number of unlabeled edges. To tackle this issue, we developed a semi-supervised learning method using ideas adopted from the label propagation algorithm [9] for edge label inference, which will be discussed later in section 4.3.

4.2 Document Parsing

For preparing node content, we acquired publicly available PDF documents of papers in the Delve dataset when accessible. For nodes when PDFs are not accessible, we acquired their other content information such as title, authors, abstract, publication venue, publication year, URLs, etc. The documents were collected through web crawling from different sources. The web crawler was designed to go through a list of scholarly document URLs to locate and download PDF files that are openly available. This URL list of papers was obtained from the Microsoft graph dataset and by crawling the web. We were able to collect about 680,000 PDFs, according for 32% of the nodes in the whole graph.

[18]https://academicgraph.blob.core.windows.net/graph-2015-11-06/index.htm

The downloaded PDFs are converted into text using the Linux pdf2text tool. Then using methods proposed in [6; 32], we sectionize the text. We extract the following sections from each document:

Header: This is composed of the title of the paper, the author(s) information, and the keywords when available;

Abstract: We extract the paper abstract when available;

Paper body text: This is composed of the document text excluding the header information, abstract and references;

Citation: We extract the document references made in the paper, which is then parsed further to separate the different parts of the citation: author, title, year of publication, and page numbers;

Citation context: These are the sentences encompassing a citation reference in the body of the document;

Cited links: These are the URLs cited in the paper. These URLs could be links relevant to the research work, link to datasets used, or link to the implementation codes.

Due to the variety of the documents we currently have in our database, we still experience some of the parsing issues due to variations in formatting as noted in [10]. We expect in time for this problem to be reduced with the improvement to the parsing algorithm.

The citation information of the papers is extracted from the paper text, Microsoft dataset, and the web. We proceed to identify and merge the different citations to the same article, and then build the citation network. The apparent difficulty in dealing with citations made in different conferences and journals is the variations in the formatting of documents and their citation methods, such as the MLA, APA, Chicago, Harvard, and other formats. There exist also papers that do not follow any particular guideline citation format, even include typos in citation.

In order to split each cited paper in *References* into sections such as *authors*, *title*, *publication year* and so on, we developed a rule-based method and combined it with the method proposed in [6]. The heuristics in the constructed rules have considered the variation of reference styles in different documents. For instance, the author section normally appear first, and often separated by comma from each other. The publication year, a double quote or a full-stop usually separates the authors and the title sections. However, these observations do not present a generalization over all the citation syntaxes which we incorporated in our splitting method. After the splitting, a reference paper appearing in different styles are merged as a single one. Then, we proceed to create the citation network of the system by building the links between papers and datasets based on their citation relationship. Next, we discuss the citation network building and labeling.

4.3 Delve System Design

The Delve system is made up of two main modes of operation. A high-level view of the system architecture is shown in Figure 1. The offline processing module includes the remote structure, framework, and design of the system to ensure

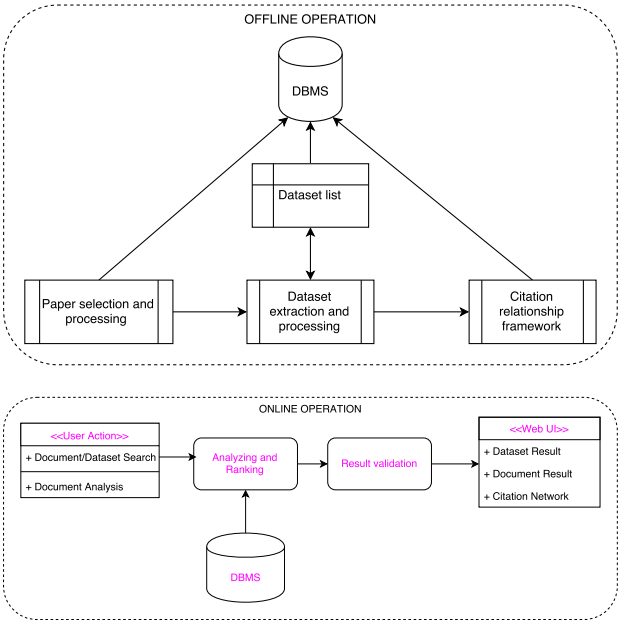


Figure 1: Delve Offline and Online Process

its functionality. New improvements and updates are regularly made to the system. In online processing, Delve web interface accepts user query (search phrase or file upload). Delve analyzes and executes the query, launching different processes that perform the execution, ranking, and result analysis.

We express a paper or dataset source in our system database to an *entity*. Each *entity* is made up of a set of attributes (title, authors, papers cited, papers citing it, its citation information, etc). From this information, a citation network $G = \{V, E\}$ is built through linking two entities if one cites the other. An edge between v_i and v_j are labeled **positive** (dataset related) if v_i cites v_j because v_i uses the dataset in v_j, or **negative** (not dataset related) otherwise. As discussed previously, the data-driven search will explore nodes that are linked by positive edges. A significant challenge in Delve is the edge label assignment in a large citation network with only 5% known labels, which are crowd-sourced.

4.3.1 Edge Label Assignment

Based on the logical assumption that a highly cited dataset entity will most likely gain more positive citations than negative citations in the future, we see that for our problem, entity citations labels are interrelated. We adopted two methods (label propagation [37] and PageRank [26]) and modified them to infer labels for the edges with unknown labels. They are selected due to their advantages in the amount of priori needed, better run time, and fitness our problem, considering that we are working with a large network with missing information.

Before using the inferred edge labels in query answering, we conducted extensive experiments to evaluate the developed methods on a total of 101,503 labeled edges. The results are reported in Table 1. The studied network is overwhelmingly unbalanced with most of the nodes having very low in-degree. Therefore, to ease the performance assessment,

Table 1: Performance results of edge label inference using modified PageRank and label propagation

	Pagerank	Label Propagation
AUC	0.8231	0.8979
Precision	0.9933	1.0
Recall	0.6391	0.6260

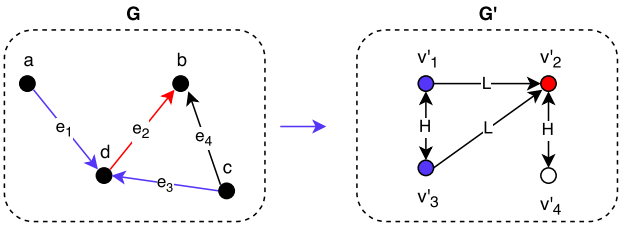


Figure 2: Graph G is reconstructed into G', where $e_i = v'_i$, $L = 0.5 + Sim_{ij}$, $H = 1 + Sim_{ij}$. In G, different types of link relationship are illustrated by colors, blue links are positive, red links are negative, while a black edge (e_4) shows a link with a missing label. In G', these edges are represented by nodes in corresponding colors, while the white node v'_4 signifies the black edges e_4 with a missing label.

we randomly sample 10% of incoming edges to nodes with high in-degree as test datasets. The results are obtained from running the evaluation five times with random samples and then taking the average. We measure the performance of these methods based on the precision, recall, and AUC value. The performance of both methods are comparable. However, label propagation is a little better and more stable than RageRank w.r.t. parameter settings. Thus, the adoption of the label propagation method as the main labeling algorithm for the Delve system. The details of these two approaches are given next.

4.3.1.1 Label Propagation.

Label propagation is a popular graph-based semi-supervised learning framework which is efficient in large graphs. The classic problem setup is defined as follows: given a graph $G = \{V, E\}$, a set of nodes V_l have known labels, while the remaining nodes V_{ul} have unknown labels. Label propagation infers the labels of V_{ul} by progagating the known labels from V_l to V_{ul}.

We aim to label the edges, and thus restructure our graph to $G' = \{V', E', W'\}$, where the set of nodes V' is the set of edges E in graph G and E' is the set of generated edges whose weight W' show the calculated similarities between two edges corresponding to nodes $(v'_i, v'_j), \forall v'_i, v'_j \in V'$. The edges E' are generated by linking each edge e_i in the original G (v'_i in G') to the top 20 similar edges e_j (v'_j in the original G') that have the same target node as e_i or where the target node of e_i is the source node of e_j.

A simple example of this reconstruction is shown in Figure 2. Since this is not an undirected graph, using the reconstructed graph G' as described above is not enough to ensure convergence. A general way to ensure convergence is to disregard the directions or make the graph matrix stochastic and irreducible [26]. The problem with these solutions w.r.t. our work is that they implicitly make a vague assumption that all papers and dataset are somewhat related.

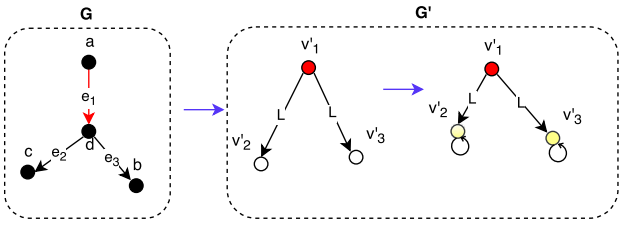


Figure 3: A simple case of a dead-end. A preprocessing step is applied on the reconstructed graph - assigning an "unknown" label to each node with a dead-end (in this case nodes v'_2 and v'_3; shown in yellow).

We, therefore, opt for a different solution and introduce a new label "unknown". Before running the propagation algorithm, we scan through the reconstructed graph, searching and assigning the label "unknown" and append a self-loop to the nodes $v'_d \in V'$ with a dead-end. Figure 3 presents a simple example of a dead-end preprocessing.

To define the similarity between v'_i and v'_j (two edges e_i and e_j in the original graph G), we extract the number of dataset keywords[19] from each citation context (i.e., the sentences which encompass the citations). We then defined a Gaussian similarity score between pairs of edges (e_i and e_j in G)

$$Sim_{ij} = \exp(-\frac{||d_i - d_j||^2}{2\sigma^2}) \quad (1)$$

where $d_i = \frac{n_d}{n_c}$, n_d is the number of dataset related words in the sentences which encompass the citation depicted as d_i, and n_c is the number of such sentences in the source papers.We then assign a weight:

$$W'_{ij} = \begin{cases} 1 + Sim_{ij} & \text{if } v'_i \text{ and } v'_j \text{ have the same target } v_t \in G, \\ 0.5 + Sim_{ij} & \text{otherwise.} \end{cases} \quad (2)$$

With the constructed graph $G' = \{V', E', W'\}$ where a small portion of V' have verified labels, label propagation algorithm is run to propagate the given labels to unlabeled V'. After the labeling step, the graph G' is reconstructed back to the original graph G.

4.3.1.2 PageRank.

PageRank algorithm [26] determines the importance of a web page based on the importance of other web pages with which it has in-links. A web page that has more in-links will have higher importance (measured as PageRank score). In-links can be considered as weighted votings, where the weight of a link depends on the importance of the source page and also the number of out-links the source page has. In our built network, there is a general observation: if a paper node or dataset node is highly cited with positive edges, the likelihood of a new unknown citation to this node to be also positive is high. This observation conforms to the mechanism of PageRank: web page with many in-links are good, and in-links from a "good" web page are better than in-links from a "bad" web page. We correspond dataset citation links to "good" links and others to "bad" links. We

[19]We manually compiled a list of dataset related words and phrases, such as: "used dataset from", "gene banks","copora", etc.

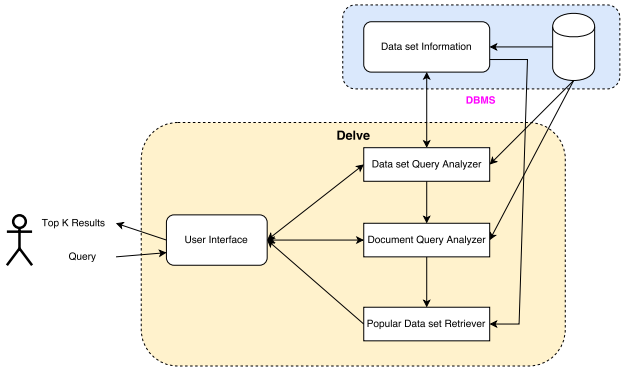


Figure 4: Delve search schema

then, apply PageRank to rank the nodes, with an expectation that highly cited nodes relevant to dataset are ranked higher than others.

To do this, we construct a Markov model M that represents the graph G as a sparse matrix whose element $M_{u,v}$ is the probability of moving from node u to node v in one-time step. We then compute the adjustments to make our matrix stochastic and irreducible (see [26]). The PageRank scores are then calculated using a modified version of the quadratic extrapolation algorithm, which accelerates the convergence of the power method [18]. In the original PageRank algorithm, the PageRank score r_u for node $u \in V$ can be recursively defined as:

$$r_u = \Sigma_{v \in L_u} \frac{r_v}{N_v}, \quad \forall u \in V. \quad (3)$$

where N_v is the number of out-links from node v, and L_u is the set of nodes that are connected to u.

To include the known positive labels of edges, we modified the algorithm such that the PageRank score is recursively defined as:

$$r_u = \Sigma_{v \in L_u} \begin{cases} \frac{r_u}{N_v} & \text{if edge } (v,u) \text{ is 1,} \\ -\frac{r_u}{N_v} & \text{otherwise,} \end{cases} \quad (4)$$

Equation (4) is set such that incoming negative links will decrease the rank score of a node, while incoming positive links will increase the rank score of a node. An initial rank score of $\frac{1}{N}$ is assigned to each node (N is the total number of nodes). After converged ranking scores are obtained for each node, a threshold is applied to nodes, whose incoming citations are labeled as positive if its score is above the threshold, and negative otherwise. It is worth noting that the threshold has a high impact on the inference accuracy. We set it be the 85th percentile after cross-validation.

4.3.2 Delve Search

Figure 4 presents a scenario where a user is searching for a dataset to use as a benchmark dataset for her research project. She enters her research topic of interest as a *query* (input). Delve analyzes this query and presents the user with results (outputs) ranging from matched datasets to papers that used these datasets, which all ranked by relevance. An example of the search result page is shown in Figure 6. To save users' time on filtering out unusable data sets, we separate the invalid datasets (datasets that are no longer available) from the valid ones. Although unavailable (see

the third tab in the middle of Figure 6), the information of these datasets is still presented for the literature review and survey purpose. It is worth mentioning that a dataset node can actually be a paper if it contains direction or descriptions of the dataset used in other papers.

Delve is also capable of handling queries based on snippets of the dataset name. For example, the user might have a dataset in mind, e.g., the PTB Diagnostic ECG Database "http://physionet.org/physiobank/database/ptbdb/". However, the user only knows that the dataset is from *physionet* and is called *ptb*. Entering "*physionet ptb*" will return the correct result.

When a user inputs a query using the user interface, the search phrase is parsed and sent to the dataset query analyzer and the document query analyzer for processing and analysis the search result results. The search schema is made up of three main layers namely: Dataset Query Analyzer, Document Query Analyzer, and Popular Dataset Retriever.

Dataset Query Analyzer (DaQA): Given an input search phrase, DaQA converts the search phrase into a dataset query to search the Delve database for dataset items that match the user search phrase. These dataset items are the nodes associated with positive incoming edges. They can be dataset entities, or paper entities containing dataset information. The matched entities are validated and ranked according to their relevance scores. The result is sent both as output to the user and as input to the Document Query Analyzer to retrieve documents that use the returned dataset items.

Document Query Analyzer (DoQA): The document query analyzer receives as input the user search phrase and the dataset items matching the search phrase. The DoQA converts the search phrase into a document search query, queries the database for documents matching the query, and returns a ranked result. Papers citing the matched datasets items are assigned a boosted weight in the relevance ranking algorithm. The returned results are in turn sent as output to the user and its indexes sent as input to the *Popular dataset retriever* to get the prevalent datasets used by papers matching the search query.

Popular Dataset Retriever (PoDR): The work of the popular dataset retriever is to query the database for the popular dataset items cited by the papers returned by the DoQA. More specifically, it retrieves dataset nodes that have incoming positive links from nodes presenting papers returned by the document query analyzer. These datasets are then ranked according to their citation count.

After the query processing and analysis, Delve returns to the user the result from the different stages of analysis. Figure 6 presents a sample of the result returned by the different stages. The dataset list displayed in the middle panel of Figure 6 shows the results from the DaQA (matched datasets). The result from the first (DaQA) and third stage (PoDR) can be seen under the "matched" and "popular" tabs respectively. Results from the second stage (DoQA) are displayed on the rightmost panel having a list of documents with their metadata including the document title, authors, venue, and abstracts.

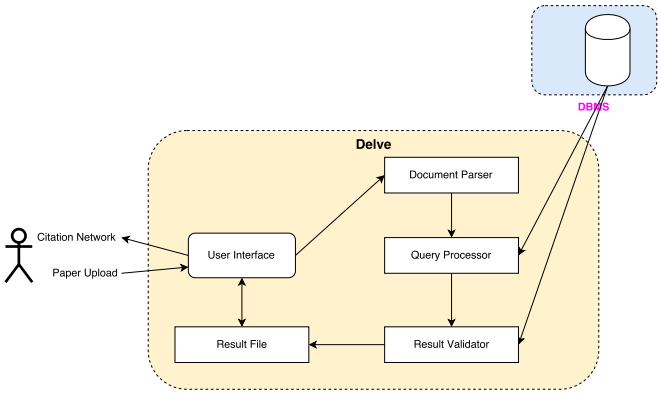


Figure 5: Delve document analysis schema

4.3.3 *Document Analysis*

Delve queries can range from just a keyword search, dataset search, or by uploading a file for on-line analysis. The document analysis provides a medium where researchers can quickly analyze a scholarly document regarding how it is relevant to other documents, without checking the references and searching and reading each of them. This analysis could be for different reasons, like to understand the relationship between a given paper (published or unpublished) with other scholarly papers, to check for other similar papers for further research, or to see the common datasets used in the citation network community of a given paper. Delve allows users to upload the PDF file of the paper for analysis. Delve analyzes the PDF, translates the results into a query, processes the query, and displays the result as a visual citation graph (see Figure 5). The user can then use the Delve citation graph GUI to analyze the paper further (see Figure 8). We plan to provide more information from the document analysis. Further additions will be made to the system later.

The document analysis is organized into three sub-processes namely: document parsing, query processing, and result validation. These sub-processes are explained below:

Document Analyzer: When a document PDF is uploaded, Delve converts the PDF to text using the Linux pdf2text tool. Then using our citation parsing algorithm, we parse and extract the reference list from the academic document. The result is a list of tuples containing the references made in the paper. Each reference tuple is composed of three items - authors, title, and other information.

Query Processor: On receiving the reference list, the query processor sends a query to the DBMS for retrieving the relationship between the document and the items in the Delve database, as well as for the relationship between the references of this paper and items in Delve database. This process is done by converting this paper and each of its references to a database query.

Result Processor: The result processor validates the query result and sets the format to the citation network syntax. The result is written to a temporary file. The file name is returned to the user interface which then reads the file and displays the citation network.

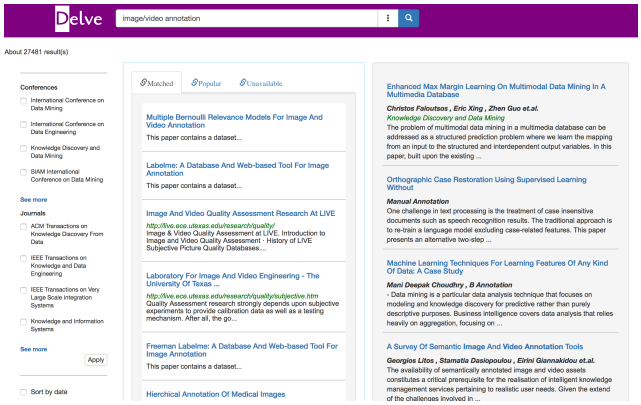


Figure 6: Delve search result page, showing the matched datasets (middle) and documents (right). On the left, there are different filter selections for making advanced search and modifying the returned results.

5. HOW IT WORKS

As discussed before, Delve offers data-driven search and document analysis. In this section, we demonstrate how Delve works in these two modes. For better understanding Delve, please try it at https://delve.kaust.edu.sa to further investigate the interesting results and features provided by Delve.

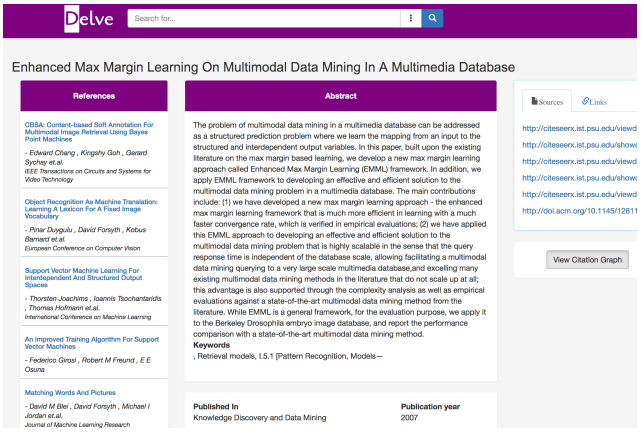
5.1 Delve Search System

Delve supports two search modes: a normal search and an advanced search. The normal search is structured to be intuitive and simple. The default output includes all the papers in satisfying the query, ranked by relevance as explained in section *4.3.2* above. The advanced search provides an extended medium for querying the system. A user can make use of a combination of different filter selections provided by the Delve web interface to modify the returned results. These filters include searching specific conferences, journals, authors, etc, as shown in the left part of Figure 6.

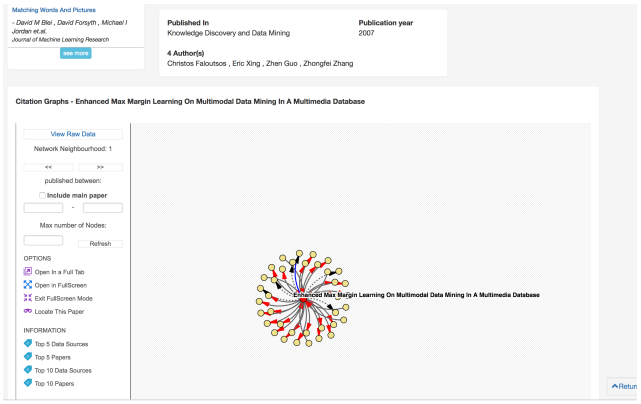
The search result page area is split into two: the dataset result (displayed in the middle of Figure 6) and document search results (displayed on the right in Figure 6). The dataset results panel is composed of 3 tabs: *Matched*, *Popular* and *Unavailable*. The *Matched* tab contains the datasets matching the search query, the *Popular* tab contains the list of popular dataset used by the documents returned by the document search query, and the *Unavailable* tab contains the temporary or permanently unavailable dataset (whose web links are no longer accessible).

Figure 6 shows the search result of the phrase "image/video annotation". The dataset search result is either a URL (pointing to the web page of the dataset), or a paper (where the dataset is introduced or used). A click on the dataset result, if a link, will take the user to the web page of the dataset. If it is a paper, a click on it will open the information page of the paper. A click on a paper search result item will also open the information page of the selected document, as shown in Figure 7. In this case, it is a paper entitled "Enhanced Max Margin Learning On Multimodal Data Mining In A Multimedia Database" [11]. The information page is composed of several sections :

Document Metadata: including the document abstract, authors, publication year and venue;



(a) Information page of the selected item showing the item metadata



(b) Information page of the selected item showing the item citation network

Figure 7: More details about a selected item are provided on the item's information page

References: this shows the list of references in the paper. Selecting "see more" will open a full list of the document references;

URL section: the section is made up of two subsections - Sources (links to document file) and Links (web links referenced in the document);

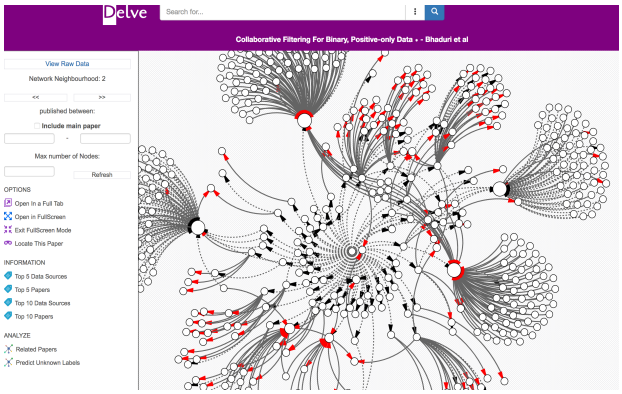
Citation Graph: showing the citation network of the paper. More details about this section are presented in Section 5.3.

5.2 On-line Document Analysis

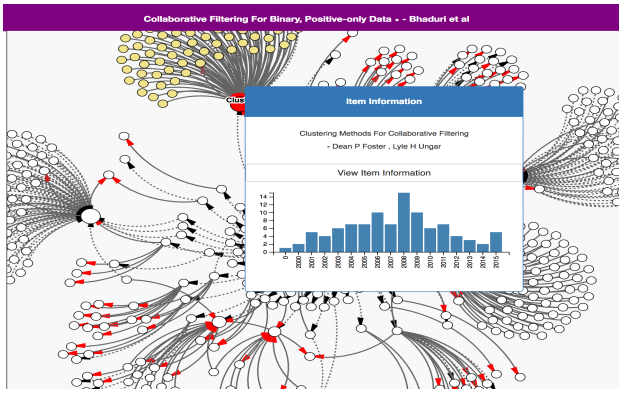
As discussed previously, Delve offers users a document analysis system, which is an important feature of Delve. On the Delve homepage (see Figure 8a), a user can click on "Analyze file", and then select a PDF document to be analyzed, and upload it. The PDF document is then analyzed by the system, and the result is displayed on a new page in the form of a citation graph.

Figure 8b shows the result of analyzing a paper entitled "Collaborative Filtering for Binary, Positively Data" [34], which is recently published at *ACM SIGKDD Explorations*

(a) Delve homepage



(b) Results shown after analyzing a PDF file



(c) Bar chart showing the number of citations per year in the displayed citation network

Figure 8: The citation network produced by the document analysis system after analyzing a given document

Newsletter volume 19 in 2017. It is worth noting that this paper is not included in our system database at the moment of analysis. However, some of its reference papers are included in Delve database. Therefore, we can analyze how this paper and its references are relevant to other papers. The citation network in Figure 8b is centered at this analyzed paper and shows its 2-hop neighbors (by setting *Network Neighborhood: 2* at the top-left corner). Actually, by changing the setting of *Network Neighborhood: k* Delve can display k-hop citation network centered at the analyzed paper. More discussion about the citation network analysis is given next.

5.3 Citation Network Analysis

One of the aims of our system is to provide researchers with a medium to visualize and analyze paper and dataset relationships. Delve provides a simple GUI interface of the citation network of the scholarly documents in its database. Each node represents a paper or a dataset. The color of a node signifies how it is cited. A darker color shows that a node is cited mainly based on dataset. The citation relationship between nodes is shown by a directed link. A blue edge shows a dataset based relationship (with a positive label); a red edge shows a non-dataset based relationship (with a negative label), or a broken edge (with an unknown label). Mouse hovering over a node displays the node title.

The nodes can be interacted with in 3 different way: 1) a left click on a node displays more information about the item - including a bar chart showing the number of citations per year to the selected node based on the displayed citation network (see Figure 8c). Clicking on a bar in the displayed bar chart shows a list of documents citing the selected node document published in the year shown by the bar corresponding to the selected paper. 2) A right-click on a node pops out a list of documents citing or being cited by the node document. And 3) a double-click on a node opens up the information page of the document corresponding to the node.

The tool panel located at the left of the citation network provides some additional tool for analysis. The user can increase the size of the network neighborhood, filter selected papers by year of publication, select the most related papers based on the citation relationship, etc. The raw data used in constructing the citation network can be retrieved by clicking on the "View Data" button.

Another feature of Delve is the online edge inference feature. Currently, Delve uses a modified version of label propagation (see section *4.3.1.1*) to predict the unknown labels in the citation network. When a user clicks on "Infer unknown", Delve reads the citation file, applies the inference algorithm, updates the file with the result and signals the web interface of completion. The web interface reads the updated file and displays the new result.

6. FUTURE WORK

Delve is already launched in public for noncommercial free use. However, it is still young. There are several directions to promote the system. In this section, we present and discuss some future plans for the Delve system.

6.1 Algorithmic Improvement and Database Extension

There are different areas of algorithmic improvement. We plan to improve the document parsing algorithm to improve the Delve database and also the performance of the Delve document analysis system. Another area of improvement is in the citation relationship inference. We plan to apply a more sophisticated inference method. We also need to make all these algorithms efficient for big data.

We plan to extend the Delve database by including papers in conferences and journals out of data mining and machine learning fields, like Bioinformatics, Geology, Biology, Computer vision, etc. With the database extension, we plan to extract more datasets, thus, enriching the Delve dataset database.

6.2 Document Analysis and Citation Network

Currently, Delve shows a binary citation relationship (dataset related and non-dataset related). We plan to extend this to include different types of relationships. For instance, a non-dataset based citation from one paper to another exists probably because of the similar method they used, or because one is the prior work of another, or just because they are from the same authors though having irrelevant content. This feature will improve the document analysis experience as it will provide users with more information about the document.

Another exciting direction is to not only show the citation relationship but also show how citation changes over the years. Knowing how citation changes over time would provide a better understanding of the papers - the significance and impact of the papers to their respective research field and science in general. We also plan to provide more network and document analysis tools. Some features might include the following:

- Recommend uses documents to read, datasets to use, and authors to follow, given the query history;
- Show the top K popular datasets in different research areas;
- Identify influential papers based on the citation network analysis by understanding the roles they played when being cited.

6.3 Structured Document Information

Another plan of our future research direction is to generate structured abstracts from documents texts. A structured abstract is an abstract structured in sections (e.g., objectives, method, results) providing a general summary of the whole document. With this feature, Delve document analysis will provide users with rich and concise information about a given paper, and saving users' time on reading.

7. CONCLUSIONS

The availability and access to dataset have been shown to be a driving factor in several scientific research fields and the advancement of science in general. This paper presents Delve, a system for academic search with a focus on dataset retrieval and document analysis. The Delve search system provides researchers with a medium for data-driven searches. The search result includes datasets and documents ranked by relevance. Delve also presents more information on the documents, the citation network, and useful analysis tools. The Delve document analysis feature allows users to upload the PDF of a scholarly paper and then returns to users a citation graph showing how the given document relates to other documents. Users can further take the citation analysis tools to analyze the results. With additions to the system, we plan to retrieve and show more information from the analyzed paper.

In contrast to prior systems, Delve provides researchers with 1) an easy-to-use medium to locate and retrieve information on relevant documents and dataset, 2) a medium to analyze and visualize the relationship between documents and datasets. This system can answer questions that no scholarly search engine has been able to answer so far. We showed how Delve is beneficial to researchers and for scientific research in general. We believe the Delve system will not just reduce the time required to analyze a paper or find a relevant dataset, benchmark or scholarly document, but will improve the quality of research by providing the user with a platform to understand these entities better and how they interrelate with each other.

8. REFERENCES

[1] About citeseerx. http://citeseerx.ist.psu.edu/about/site.

[2] M. P. Adams, C. J. Collier, S. Uthicke, Y. X. Ow, L. Langlois, and K. R. OBrien. Model fit versus biological relevance: Evaluating photosynthesis-temperature models for three tropical seagrass species. *Scientific reports*, 7, 2017.

[3] U. Akujuobi and X. Zhang. Delve: A data set retrieval and document analysis system. In *ECML-PKDD Demo*, 2017.

[4] C. Cardamone, K. Schawinski, M. Sarzi, S. P. Bamford, N. Bennert, C. Urry, C. Lintott, W. C. Keel, J. Parejko, R. C. Nichol, et al. Galaxy zoo green peas: discovery of a class of compact extremely star-forming galaxies. *Monthly Notices of the Royal Astronomical Society*, 399(3):1191–1205, 2009.

[5] G. Cedersund and J. Roll. Systems biology: model based evaluation and comparison of potential explanations for given biological data. *The FEBS journal*, 276(4):903–922, 2009.

[6] I. G. Councill, C. L. Giles, and M.-Y. Kan. ParsCit: an open-source CRF reference string parsing package. In *LREC*, volume 2008, 2008.

[7] R. P. Duin. A note on comparing classifiers. *Pattern Recognition Letters*, 17(5):529–536, 1996.

[8] B. Efron. [statistical modeling: The two cultures]: Comment. *Statistical Science*, 16(3):218–219, 2001.

[9] Y. Fujiwara and G. Irie. Efficient label propagation. In *Proceedings of the 31st international conference on machine learning (ICML)*, pages 784–792, 2014.

[10] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98. ACM, 1998.

[11] Z. Guo, Z. Zhang, E. Xing, and C. Faloutsos. Enhanced max margin learning on multimodal data mining in a multimedia database. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 340–349. ACM, 2007.

[12] D. J. Hand et al. Classifier technology and the illusion of progress. *Statistical science*, 21(1):1–14, 2006.

[13] H. C. Harris, J. A. Munn, M. Kilic, J. Liebert, K. A. Williams, T. von Hippel, S. E. Levine, D. G. Monet, D. J. Eisenstein, S. Kleinman, et al. The white dwarf luminosity function from sloan digital sky survey imaging data. *The Astronomical Journal*, 131(1):571, 2006.

[14] H. Hirsh. Data mining research: Current status and future opportunities. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 1(2):104–107, 2008.

[15] T. L. Isenhour. *The Evolution of Modern Science*. Bookboon, 2015.

[16] A. J. Jakeman, R. A. Letcher, and J. P. Norton. Ten iterative steps in development and evaluation of environmental models. *Environmental Modelling & Software*, 21(5):602–614, 2006.

[17] M. Janssen, Y. Charalabidis, and A. Zuiderwijk. Benefits, adoption barriers and myths of open data and open government. *Information systems management*, 29(4):258–268, 2012.

[18] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating pagerank computations. In *Proceedings of the 12th international conference on World Wide Web*, pages 261–270. ACM, 2003.

[19] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

[20] S. Levy. The gentleman who made scholar, 2015. https://medium.com/backchannel/the-gentleman-who-made-scholar-d71289d9a82d.

[21] M. Lichman. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.

[22] National Research Council and others. *Models in environmental regulatory decision making*. National Academies Press, 2007.

[23] National Science Board (US). *Science & engineering indicators*, volume 1. National Science Board, 2012.

[24] N. Padmanabhan, D. J. Schlegel, D. P. Finkbeiner, J. Barentine, M. R. Blanton, H. J. Brewington, J. E. Gunn, M. Harvanek, D. W. Hogg, Ž. Ivezić, et al. An improved photometric calibration of the sloan digital sky survey imaging data. *The Astrophysical Journal*, 674(2):1217, 2008.

[25] N. Padmanabhan, D. J. Schlegel, U. Seljak, A. Makarov, N. A. Bahcall, M. R. Blanton, J. Brinkmann, D. J. Eisenstein, D. P. Finkbeiner, J. E. Gunn, et al. The clustering of luminous red galaxies in the sloan digital sky survey imaging data. *Monthly Notices of the Royal Astronomical Society*, 378(3):852–872, 2007.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[27] T. Pedersen. Empiricism is not a matter of faith. *Computational Linguistics*, 34(3):465–470, 2008.

[28] S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[29] I. Strateva, Ž. Ivezić, G. R. Knapp, V. K. Narayanan, M. A. Strauss, J. E. Gunn, R. H. Lupton, D. Schlegel, N. A. Bahcall, J. Brinkmann, et al. Color separation of galaxy types in the sloan digital sky survey imaging data. *The Astronomical Journal*, 122(4):1861, 2001.

[30] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The sdss skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 570–581. ACM, 2002.

[31] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[32] D. Tkaczyk, P. Szostek, P. J. Dendek, M. Fedoryszak, and L. Bolikowski. Cermine–automatic extraction of metadata and references from scientific literature. In *Document Analysis Systems (DAS), 11th IAPR International Workshop on*, pages 217–221. IEEE, 2014.

[33] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[34] K. Verstrepen, K. Bhaduriy, B. Cule, and B. Goethals. Collaborative filtering for binary, positiveonly data. *ACM SIGKDD Explorations Newsletter*, 19(1):1–21, 2017.

[35] N. Webster. *Webster's Revised Unabridged Dictionary of the English Language*. G. & C. Merriam Company, 1913.

[36] D. G. York, J. Adelman, J. E. Anderson Jr, S. F. Anderson, J. Annis, N. A. Bahcall, J. Bakken, R. Barkhouser, S. Bastian, E. Berman, et al. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120(3):1579, 2000.

[37] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.