Collective Development of Large Scale Data Science Products via Modularized Assignments: An Experience Report

Bhavya
University of Illinois at Urbana-Champaign
bhavya2@illinois.edu

Assma Boughoula
University of Illinois at Urbana-Champaign
boughou1@illinois.edu

Aaron Green
University of Illinois at Urbana-Champaign
aarongg2@illinois.edu

ChengXiang Zhai
University of Illinois at Urbana-Champaign
czhai@illinois.edu

ABSTRACT

Many universities are offering data science (DS) courses to fulfill the growing demands for skilled DS practitioners. Assignments and projects are essential parts of the DS curriculum as they enable students to gain hands-on experience in real-world DS tasks. However, most current assignments and projects are lacking in at least one of two ways: 1) they do not comprehensively teach all the steps involved in the complete workflow of DS projects; 2) students work on separate problems individually or in small teams, limiting the scale and impact of their solutions. To overcome these limitations, we envision novel synergistic modular assignments where a large number of students work collectively on all the tasks required to develop a large-scale DS product. The resulting product can be continuously improved with students' contributions every semester.

We report our experience with developing and deploying such an assignment in an Information Retrieval course. Through the assignment, students collectively developed a search engine for finding expert faculty specializing in a given field. This shows the utility of such assignments both for teaching useful DS skills and driving innovation and research. We share useful lessons for other instructors to adopt similar assignments for their DS courses.

CCS CONCEPTS

• **Social and professional topics → Information systems education**; **Information science education**.

KEYWORDS

practical data science education; synergistic modular assignments; experience report

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCSE '20, March 11–14, 2020, Portland, OR, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366961

ACM Reference Format:
Bhavya, Assma Boughoula, Aaron Green, and ChengXiang Zhai. 2020. Collective Development of Large Scale Data Science Products via Modularized Assignments: An Experience Report. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366961

1 INTRODUCTION

Data Science is an emerging field and the demand for good data scientists in the industry has been growing steadily. According to the McKinsey Big Data Study, 2012 "... (in the next few years) we project a need for 1.5 million additional analysts in the United States who can analyze data effectively...". Many institutions have started offering both undergraduate and graduate level courses in Data Science (DS) to satisfy this growing demand. Universities are also offering both online and on-campus degree programs in Data Science to train a large number of students from all over the world in these areas. The core objective of all these efforts is to teach students how to extract knowledge from real-world or big data.

Previous studies on curriculum design for DS courses [16], [2],[14] emphasize the importance of practical and application-based teaching. In most courses today, this is achieved through assignments and projects. However, assignments today focus on only a few DS tasks such as Data Analysis and Model Development via Kaggle[1]-like competitions. Broadly speaking, Data Collection, Data Analysis, and Data Visualization are the three essential steps of deriving insights from Big Data with Data Science[5]. Ideally assignments should teach the three concepts cohesively so that students can experience the complete workflow of real-world DS applications. On the other hand, projects (e.g. Capstone Projects) are typically required at the end of a course and the goal is to emulate real-word DS projects. So, students usually work on all the 3 steps to complete a project. But since mostly small teams work on one project, the scale and impact of the resulting products is often limited. We believe that if more students work collectively, more large-scale innovative DS products can be developed.

To overcome the limitations of existing DS course assignments and projects, we envision novel "synergistic modularized assignments" to achieve the following main goals: **Goal1**: Foster large-scale innovation and research by collective and iterative development of a real-world DS product. **Goal2**: Each student should comprehensively understand and work on most (if not all) steps

[1] https://www.kaggle.com/

involved in the complete workflow of the product. This means that assignments should be decomposed into sub-tasks that can be performed by a large number of students in parallel. Further, an infrastructure is required to support the assignment where a large number of students can perform computations on real-world datasets.

We report our experience with designing and deploying such an assignment in an upper-level course on introductory Text Mining and Information Retrieval. The assignment enabled students to collaboratively build an Expert Search Engine by completing four synergistic individual assignments (plus one final project). They also learned the complete workflow of building a search engine including how to crawl data, evaluate ranking algorithms, improve search engines, and develop a search engine interface. We conclude with analyzing student submissions and some lessons learned during the experience.

2 RELATED WORK

Traditional assignments in DS courses: Several works have documented example assignments and suggested best practices for assignment design in DS courses. For example, in [9],[12] authors provide example assignments and projects from their DS courses in Statistics and Data Visualization courses respectively. Love et al. [10] discuss what makes a quality assignment in Data Mining and share a sample assignment using Open Data. However, current assignments have a limited scope and are not typically geared towards teaching innovation. Our assignment framework allows students to learn the complete workflow of developing a novel large-scale DS application by decomposing the development into modules. Anderson et al. [2] also decompose large Data Science programming assignments into sub-problems to scaffold students through the learning process. However, the sample assignments reported still have a small scope as each assignment covers the implementation of one DS algorithm, such as Naive Bayes, which is further broken down into sub-parts.

Techniques for teaching innovation in DS courses: Recently, Datathons [3] or hackathons focused on data are used in DS courses to drive innovation. Most hackathons often require collaboration with companies or other organizations and typically have a very short duration. Since such collaborations are not always feasible especially over the long periods of time required to develop every component of a DS product, our goal was to develop self-sustaining assignments. Dinter et al. [6] perform a systematic study of teaching data-driven innovation and emphasize the value of teaching the complete workflow of building innovative products via semester long projects. However, they do not consider the scale of the developed product which often suffers due to small team sizes in projects. Our assignment enables a large number of students to collectively innovate.

Platforms supporting DS research: Many platforms have been developed to support DS research and innovation tasks. Notably, Human Computing and Crowdsourcing [13] based platforms such as Amazon Mechanical Turk [2] enable large-scale data validation and annotation by utilizing the collective skills of workers. However, the goal is not to educate the workers. Platforms like Kaggle [3] that run data science competitions are often used in DS education. But there is no existing platform or framework that allows integration of all the tasks in DS product development in a cohesive manner for education. We use a cloud-based virtual lab based on CLaDS [8] as it allows instructors to create such an integrated framework and deploy it as an assignment.

3 ASSIGNMENT DESCRIPTION

We now describe how we designed, delivered and graded a synergistic modular assignment that enabled students to build an Expert Search Engine in our Text Mining and Information Retrieval course.

3.1 Overview

The assignment was designed for an upper-level course on Text Mining and Information Retrieval. The course is targeted towards upperclassmen and graduate students having good programming skills. The core objective of the course is for students to learn how to develop Search Engines and Intelligent Text Systems. A Cloud-based virtual lab based on CLaDS [8] was previously developed to deliver the course assignments.

During previous iterations of the course, one programming assignment was designed as a competition for students to develop better ranking functions for text retrieval models using the MeTA toolkit [11] with Python as the programming language [4]. A web-based *leaderboard* was maintained to report the performance of student submissions (using evaluation metrics such as accuracy, precision, etc.) on retrieval datasets such as the Cranfield dataset [5]. We extended this assignment into a comprehensive assignment for developing an Expert Search Engine. The assignment was released in Spring 2019 in a class with > 200 students.

As stated in **Goal1**, one goal is to build a large-scale DS product that motivates students to innovate. The general idea is to select any novel application relevant to a course and decompose its development into multiple steps to be delivered as individual assignment modules. We chose the problem of building an Expert Search Engine that would help find people with a given expertise of interest. Such a vertical search engine is required as general purpose search engines like Google do not perform well on this highly specialized task. It would be useful for many use-cases such as prospective students looking for advisors. Although there are some publicly available expert finders such as ExpertiseFinder [6], FacFinder [7], they often have limited scale or are not based on cutting edge retrieval algorithms. Our search engine would utilize the collective efforts of the large number of students in our class both for creating new data sets and improving dedicated retrieval algorithms.

To realize **Goal2**, we first identify instantiations of the 3 steps of DS for developing a Search Engine application. Some of the major tasks required to build a search engine application are web crawling and indexing, developing relevance judgements for evaluation, and developing a web application for search ([17] provides detailed

[2] https://www.mturk.com/

[3] https://www.kaggle.com/

[4] https://github.com/meta-toolkit/metapy

[5] http://ir.dcs.gla.ac.uk/resources/test_collections/cran/

[6] https://expertisefinder.com/

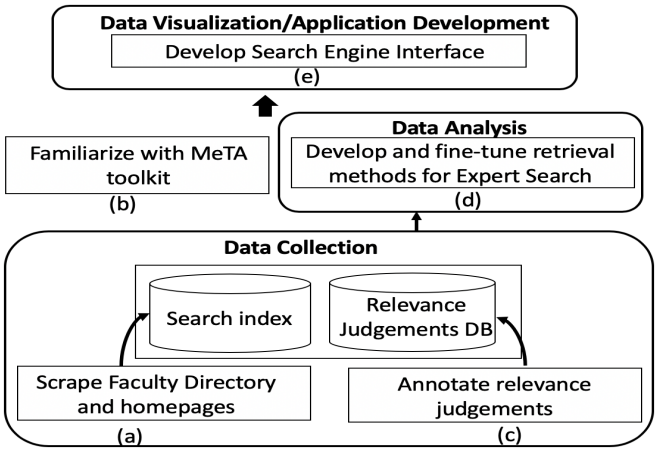


Figure 1: Overall assignment design for developing a novel Expert Search Engine application. The problem is decomposed into modules: (a)MP2.1, (b)MP2.2, (c)MP2.3, (d)MP2.4, (e)Project. These correspond to the general stages of developing DS applications, namely, Data Collection, Analysis and Visualization/Application Development. Arrows mean data or algorithms developed in preceding modules are utilized

discussions on each concept). These correspond to Data Collection (web crawling and indexing, annotating relevance judgements), Data Analysis (developing and evaluating retrieval models), and Application Development (developing search web-applications). We emphasize that even though our assignment is Information Retrieval focused, it can be adapted for other data science sub-domains, e.g Computer Vision, as these similarly instantiate the same high-level stages of DS. The assignment (or Machine Problem; MP for short) was divided into modules (MP2.1, MP2.3, MP2.4)[7] to cover one concept each. An additional module (MP2.2) was also released to familiarize students with the MeTA toolkit. Since developing a basic web application for search does not require a lot of effort, we released this task as a project topic (Project) to be picked by one team. A more collaborative approach involving more students is certainly possible and can be considered in future revisions of the assignment. The overall design is show in figure 1. We now describe each module of the assignment in detail.

3.2 MP2.1

3.2.1 Objective. : The objective of MP2.1 was that each student learns how to collect data through web scraping. Students crawl faculty directory pages (often maintained by most universities) for obtaining all faculty homepage links and scrape faculty homepages to get accurate up-to-date expertise information.

3.2.2 Design. : Each student was tasked with collecting the faculty information of one university using web scraping and parsing. Faculty directories of all departments in a university tend to have similar structures and thus require similar scrapers. To prevent cheating and maximize learning, the first task was that each student choose a unique university and department and document it along with the URL of the corresponding Faculty directory page in a shared spreadsheet. For consistency and simplicity, students were required to choose English webpages of faculty in Computer Science or an Engineering department.

The second task was for students to write Python scrapers with two main functions, one to scrape the directory page and extract faculty homepage URLs, and a second one to scrape all faculty homepage URLs and extract the text from their Biographies(Bios). We chose to use Bios only as a starting point as it is commonly available in the form of a few paragraphs in most homepages. The students were required to submit 2 output text files, one 'bio_urls.txt' containing the list of all homepage URLs with one URL per line, and another file 'bios.txt' containing the corresponding list of Bios (again having one Bio per line). Additionally, they were also required to submit their code.

To further assist students, we provided sample scrapers in a Jupyter[8] notebook and the output files for the CS department faculty of one university. We also provided links to tutorials and libraries for webscraping with Python.

The assignment module was delivered using the GitLab platform within our virtual lab. Students were given a week to complete it.

3.2.3 Grading and submission compilation. : Student submissions were first downloaded using GitLab API[9]. In addition to checking for completeness, i.e. submission of 2 text files and scraper, we checked if the 2 files were correctly formatted. We programatically checked if the files were not empty and if the number of URLs was equal to number of bios assuming each new line in the file has one URL (bio). We also checked if there was a significant overlap (>3 matching URLs) between the homepage URLs submitted by any two students.

Finally, we compiled lists of all unique homepage URLs and their corresponding bios from all student submissions and created a search index using MeTA such that each document contained one faculty bio. Only bios having more than 5 words were indexed.

3.3 MP2.2

3.3.1 Objective. : MP2.2 was aimed to familiarize students with using the MeTA tool for search [10], specifically, building a search index, using, creating and evaluating ranking functions in MeTA.

3.3.2 Design. : Students were asked to create an index for the Cranfield dataset and implement the Inl2 function (an Inverse Document Frequency model with Laplace after-effect and normalization first introduced by [1]). A skeleton code and pointers to MeTA documentation were provided and the task was to fill in the missing details. Additionally, students were asked to compare the Average Precisions of two available MeTA retreival methods on the Cranfield dataset using T-test [11] and submit the p-value in a file. It was delivered using GitLab within our virtual lab. It was released after MP2.1 and students were given one week to complete it.

[7] The numbering scheme starts from 2.1 simply because this was the second assignment in the course

[8] https://jupyter.org/

[9] https://python-gitlab.readthedocs.io

[10] https://github.com/meta-toolkit/metapy/blob/master/tutorials/2-search-and-ir-eval.ipynb

[11] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

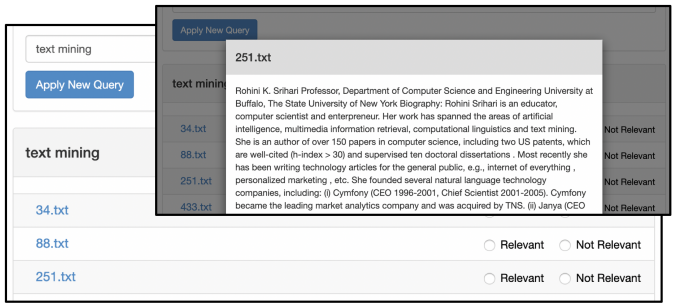


Figure 2: Screenshots of the Annotation tool used in MP2.3

3.3.3 Grading. A unit test was provided as a GitLab CI Pipeline [12] and grading was based on passing the pipeline. It checked if the Mean Average Precision (MAP) of the student's implementation was close to the MAP of the instructor supplied implementation on the Cranfield dataset. In this way, students received immediate feedback on their implementation and could also make multiple submissions. Additionally, we downloaded their submissions and checked for the presence of a file containing a proper p-value (a floating-point value between 0 and 1).

3.4 MP2.3

3.4.1 Objective. MP2.3 was aimed for students to learn how to annotate relevance judgments for evaluating search engines based on the Cranfield Evaluation Methodology[4]. The data set created in this way is a new data set that also enables new research on expert finding. In the future, with student permissions, we can release such data to the research community.

3.4.2 Design. A web based annotation tool was developed within our virtual lab as shown in Figure 2 . Given a query, the tool uses Okapi BM25 [15] in MeTA with default parameters (k1=1.2, b=0.75, k3=500) as the retrieval method and returns a list of top 20 documents (faculties) from index created from MP2.1 submissions as shown in Figure 2. We did not have the faculty names, so we simply used numbers to name the documents. Clicking on a document opens a pop-up showing the associated faculty bio as shown in the top-right corner of Figure 2.

Students were asked to log into the the tool and submit a query to search for expert CS faculty in some area (e.g. "stanford text mining", "computer vision") and evaluate whether the returned faculty bios were relevant or not. They were asked to evaluate the results of at least 1 query. After annotating 20 results for the query, they could submit their judgments. All student queries and judgments were saved in a database. To prevent many students from evaluating the same query, we asked students to log their queries in a shared spreadsheet. A maximum of 10 students were allowed to evaluate the same query.

This module was also released after MP2.1 and students were given 1 week to complete it.

[12] https://about.gitlab.com/product/continuous-integration/

3.4.3 Grading and submission compilation. Grading was based on completion, i.e., evaluation of at least 1 query. This was implemented by setting a flag in the database when a student submitted their judgments for the first query. We also performed string matching to check if the query was already used by >=10 other students and the current student logged the same query in the spreadsheet regardless. If so, the student was not awarded full points.

After releasing this module, we realized that some students might not take the assignment seriously and submit incorrect or unreliable judgements. So, using all submitted judgments might create a noisy data set for evaluating search retrieval methods. To bypass this issue, we only used the judgments for queries evaluated by at least 2 students. We used the rounded average of the all submitted relevance judgements for a query so that the final judgment score would still be binary.

3.5 MP2.4

3.5.1 Objective. The objective of the final module is for students to learn to create and fine-tune retrieval methods for search.

3.5.2 Design. This part was designed as a search competition and extended from a pre-existing assignment. It was delivered using GitLab in our virtual lab. Only the Cranfield dataset was provided to students for testing their retrieval methods locally. Their submissions were evaluated using Normalized Discounted Cumulative Gain at 10 (NDCG@10) scores against relevance judgments on 3 datasets: Faculty dataset compiled from MP2.1 submissions and judgments from MP2.3 submissions, Cranfield, and APNews. The idea was to emulate a real-world scenario where data scientists often have to use different train and test sets and need to be wary of bias-variance tradeoffs A weighted average of the NDCGs on the 3 datasets from the student's latest submission was used as their final score with highest weight given to scores on the Faculty dataset. A leaderboard (Figure 3) and a supporting database were maintained. Students were given 2 weeks to complete the assignment. Multiple submissions were allowed.

3.5.3 Grading. We used Okapi BM25 with default parameters as the baseline solution that each student was required to beat to complete the assignment. A small extra credit was provided to a few top ranked students for incentive. We directly used the database for grading.

3.6 Project

Finally, we released the task of creating a web-application for the search engine as a course project topic to be chosen by one team. Specifically, we wanted the following features in the application: 1. Search by area of expertise 2. Filter results by university and location of the university 3. Display the faculty name, email, link to homepage and context snippets in search results 4. Allow an instructor to upload any student's code submission from GitLab and use it for ranking the search results. In addition to creating the interface, the project required developing simple methods for extracting faculty names and emails from bios and finding university locations from the university names entered in the spreadsheet in MP2.1. They were provided with the top-ranking solution from

MP2.4 to use as the retrieval method for search. The project was graded manually based on completion.

4 STUDENT SUBMISSIONS ANALYSIS

We now report details on students' submissions from the main modules discussing whether students were able to complete the assignment successfully and if/how their submissions cumulatively created the Faculty Expert Search Engine.

MP2.1: A total of ≈ 6500 unique faculty homepage URLs from 139 universities accross 6 countries were collected from MP2.1. This is already much larger than the homepage URLs manually collected in existing related work [7]. Each faculty bio is a few KBs in size resulting in a total collection of size 40 MB. Even though the size of collected data is relatively small as we only asked students to collect the bios, we believe that the 6.5k homepage URLs collected is a very valuable resource it can be used as a base to crawl more information, e.g. publications and Google Scholar profile, in future offerings. Moreover, the submitted scraper scripts can be used to periodically refresh the index to get up-to-date faculty information.

Almost every student completed the assignment successfully, <10% of them lost points due to ill-formatted file submissions.

MP2.2: The goal of MP2.2 was only to familiarize students with the MeTA toolkit and they were able complete this module successfully. We are not providing a more detailed analysis here as it is not integral to the search engine application or synergistic modular assignments in general.

MP2.3: A total of ≈ 110 unique queries were annotated, thus giving us ≈ 2k total relevance judgments. Out of these only ≈ 35 queries were annotated by at least 2 students. This significantly reduced the number of relevance judgments used for evaluating the search engine in MP2.4. But we made this decision to ensure that a reliable dataset was used for evaluating the retrieval methods as mentioned in 3.4.3. Only 1 query, "natural language processing", was evaluated by the maximum allowed number (10) of students.

All queries were short (between 1 and 3 words) and covered broad fields in Computer Science e.g. blockchain, machine learning. This, in general, meant that queries were quite easy and even simple solutions, like our baseline method without any parameter tuning, achieved a very high NDCG@10 score of 0.93 on the relevance judgments from 35 queries.

MP2.4: All but 3 students beat the baseline solution indicating they were able to get a good grasp on building and fine-tuning retrieval methods. Additionally, consistent with [8], we found students were highly engaged with 50% students submitting more than 12 times after assignment completion. After the assignment, many students were interested in knowing the strategies used by top rankers which lead to a very active discussion thread on our forum. Students shared that they experimented with various retrieval methods but found that fine tuning Okapi BM25 worked the best.

It is interesting that the top ranking solution did not have the best NDCG@10 on the Faculty dataset individually as noticed in Figure 3 even though this dataset was given the highest weight while ranking submissions. This indicates that the Faculty dataset might be quite different from Cranfield/APNews datasets. In future, we will consider generating train/test sets from the Faculty dataset alone to facilitate development of a dedicated retrieval method.

Project: A team of 4 students developed a fully functional Faculty Expert Search engine as show in Figure 4 with all desired features listed in Section 3.6. Figure 4 (a) shows features 1-3 and 4 (b) shows feature 4. They developed the website using Python Flask [13] and Javascript. They used regexes to extract emails, and Named Entity tagging [14] to extract faculty names. Further, they used Google Maps API [15] to get the locations of universities. GitLab API was used to upload a given student's code submisson identified by their GitLab project ID.

5 LESSONS LEARNED AND CHALLENGES

As discussed in Section 4, the synergistic modular assignment enabled students to drive innovation as they created a fully functional Expert Search Engine. Moreover, students created two new datasets, one with faculty bios and URLs, and another one with relevance judgments. These new datasets are valuable since they enable new research of expert retrieval algorithms. Also, the high completion rate of all assignment modules and the development of the final application as mentioned in Section 4 both indicate that the students learned all the steps involved in the complete workflow of the DS application. So, we were able to achieve both **Goal1** and **Goal2** as planned, and such a new model of synergistic modular assignments is not only feasible, but has also worked well.

Although we have only experimented with the idea in one course, the methodology we have used can also be adapted to deploy similar assignments in other courses in Data Science. Based on our experience, we can make the following recommendations for any design of such assignments in the future.

1. Use a cloud-based virtual lab: Having a shared infrastructure like CLaDS for delivery and grading of all assignment modules and hosting all the student code repositories and large datasets makes it convenient for both students and instructors to collaboratively work on such large assignments.

2. Build the new assignment upon existing assignments: Developing a large assignment with multiple modules can be time consuming. Extending an existing assignment allowed us to reuse its parts including grading and delivery, and reduce the time required to develop the new assignment.

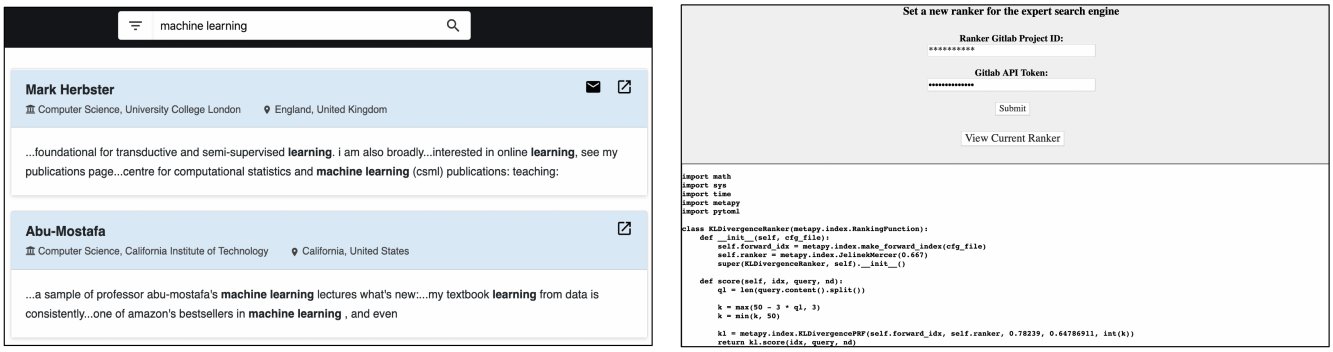
3. Choose novel problems familiar and interesting to students: The data science application chosen for the assignment should be interesting to students so that they are engaged and motivated to develop better solutions. At the same time, the problem domain should be familiar to students so that there is minimal learning curve. In our case, finding expert faculty for choosing advisors was a relevant problem for students and the CS/engineering domain was especially familiar to them. The problem had a low entry barrier; students knew how to find university websites and were able to judge whether a faculty bio was relevant to their query.

4. Design assignment modules carefully: Firstly, the problem chosen should be decomposed into modules that cover the 3 main steps of data science. Further, each module should be carefully designed such that each student works on a task of similar complexity to ensure fairness. Although each student should work

[13] https://flask.palletsprojects.com/en/1.0.x/quickstart/
[14] https://www.nltk.org/_modules/nltk/tag/stanford.html
[15] https://developers.google.com/places/web-service/intro

Rank	Alias	Overall Score		FacultyDataset (0.6)		apnews (0.1)		cranfield (0.3)		Updated	Submissions
		Current	Previous	NDCG@10	Previous	NDCG@10	Previous	NDCG@10	Previous		
1	アルマ獺	0.7197	*0.7197*	0.9332	*0.9332*	0.4372	*0.4372*	0.3867	*0.3867*	2019-03-12 | 23:43:38	25
2	firediamond	0.7184	*0.7184*	0.9333	*0.9333*	0.4355	*0.4355*	0.383	*0.383*	2019-03-12 | 23:45:20	40
3	Ranchoddas Chanchad	0.7161	*0.7153*	0.9414	*0.9399*	0.4196	*0.4196*	0.3645	*0.3645*	2019-03-12 | 23:52:08	17

Figure 3: Screenshot of leaderboard in MP2.4



(a) **(b)**

Figure 4: Screenshots of the Faculty Expert Search Engine

towards the same overall objective of the module, there should be minimum overlap between the actual task performed by each student to minimize cheating, maximize individual learning and maximize the diversity and scale of the resulting work. For example, in MP2.1, each student scraped a *unique* university listing.

5. Assignment modules may be updated every semester to iteratively enhance the product: To allow vertical growth of the product, the same modules may be released every semester. The students would be asked to perform the same steps every semester: collect new data, and refine the data, models and application accordingly. New features and required improvements can also released as additional modules or potential project topics. For example, we plan to release more project topics in upcoming semesters to improve the extraction of faculty names and emails from bios.

We also faced some challenges that may be used as additional considerations while designing such assignments:

1. More stringent grading and requirements might be needed to ensure development of accurate and high-quality products: Since the final DS product is developed entirely from student submissions, it is important that their work is of high-quality and reliable. For example, as mentioned in Section 4, we couldn't use all the relevance judgments collected since they could be unreliable. In future revisions, we could inject a few known non-relevant bios (e.g. faculty bio from a very different department) at random positions in the returned results. Only students that correctly mark the injected bios as non-relevant would get full score ensuring that students carefully go through each result.

2. Extensive documentation should be maintained to sustain the assignment over time: Students should be able to easily understand previous work to build upon it in subsequent course offerings. Instructors and students could collaboratively maintain the documentation through sites like Wiki. Students should also be asked to follow standard coding styles.

6 CONCLUSION

In this paper, we proposed novel synergistic modular assignments for data science courses. The overall idea is to decompose the development of a novel application related to a data science course into assignment modules and have students work collaboratively on each module. Unlike existing DS assignments and projects which have limited scope and impact, such assignments enable students to collaboratively participate in and learn the complete workflow of building a large scale novel DS product.

We shared our experience with deploying such an assignment in a large class of over 200 students. Our experience shows that carefully choosing the data science problem, decomposing it into well-designed modules and supporting the assignment with a suitable cloud-based infrastructure enables students to achieve the desired goals and creates an engaging learning experience. As we have only tested the assignment over one semester, there are still some existing challenges about sustaining and improving the assignment and the developed DS product reliably over time. Finally, even though we share our experience with developing a Search Engine in an Information Retrieval course, since the development of products in other data science sub-domains can be broken into the three main steps, namely Data Collection, Analysis, and Application Development, similar modular assignments can be adapted by other data science courses too.

7 ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1801652

REFERENCES

[1] Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 357–389.

[2] Paul Anderson, James Bowring, Renée McCauley, George Pothering, and Christopher Starr. 2014. An undergraduate degree in data science: curriculum and a decade of implementation experience. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 145–150.

[3] Craig Anslow, John Brosz, Frank Maurer, and Mike Boyes. 2016. Datathons: an experience report of data hackathons for data science education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 615–620.

[4] Cyril W Cleverdon, Jack Mills, and E Michael Keen. 1966. Factors determining the performance of indexing systems,(Volume 1: Design). *Cranfield: College of Aeronautics* (1966), 28.

[5] Ben Daniel. 2015. B ig D ata and analytics in higher education: Opportunities and challenges. *British journal of educational technology* 46, 5 (2015), 904–920.

[6] Barbara Dinter, Christoph Kollwitz, and Albrecht Fritzsche. 2017. Teaching data driven innovation–facing a challenge for higher education. (2017).

[7] Yi Fang, Luo Si, and Aditya Mathur. 2008. Facfinder: Search for expertise in academic institutions. *Department of Computer Science, Purdue University, Tech. Rep. SERC-TR-294* (2008).

[8] Chase Geigle, Ismini Lourentzou, Hari Sundaram, and ChengXiang Zhai. 2018. CLaDS: a cloud-based virtual lab for the delivery of scalable hands-on assignments for practical data science education. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 176–181.

[9] Johanna Hardin, Roger Hoerl, Nicholas J Horton, Deborah Nolan, Ben Baumer, Olaf Hall-Holt, Paul Murrell, Roger Peng, Paul Roback, D Temple Lang, et al. 2015. Data science in statistics curricula: Preparing students to âĂIJthink with dataâĂİ. *The American Statistician* 69, 4 (2015), 343–353.

[10] Matthew Love, Charles Boisvert, Elizabeth Uruchurtu, and Ian Ibbotson. 2016. Nifty with data: can a business intelligence analysis sourced from open data form a nifty assignment?. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 344–349.

[11] Sean Massung, Chase Geigle, and ChengXiang Zhai. 2016. Meta: A unified toolkit for text retrieval and analysis. In *Proceedings of ACL-2016 System Demonstrations*. 91–96.

[12] Deborah Nolan and Jamis Perrett. 2016. Teaching and Learning Data Visualization: Ideas and Assignments. *The American Statistician* 70, 3 (2016), 260–269. https://doi.org/10.1080/00031305.2015.1123651 arXiv:https://doi.org/10.1080/00031305.2015.1123651

[13] Alexander J Quinn and Benjamin B Bederson. 2011. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1403–1412.

[14] Bina Ramamurthy. 2016. A practical and sustainable model for learning and teaching data science. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 169–174.

[15] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[16] Il-Yeol Song and Yongjun Zhu. 2016. Big data and data science: what should we teach? *Expert Systems* 33, 4 (2016), 364–373.

[17] ChengXiang Zhai and Sean Massung. 2016. *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool.