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

ABSTRACT
Many universities are offering data science (DS) courses recently to fulfill the growing demands for skilled DS practitioners. Assignments and projects are an essential part of any DS curriculum as they enable students to gain hands-on experience of 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 steps involved in a complete workflow of a DS project; 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 a new kind of synergistic modular assignments where a large number of students work collectively on all DS tasks required to fully 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 introductory information retrieval course. Through the assignment, students collectively developed a fully functional 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
data science education, synergistic modular assignments, experience report

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 [17], [2], [15] emphasize the importance of practical and application-based teaching especially in undergraduate and introductory courses. In most courses today, this is achieved through assignments and projects. However, assignments focus on only a few DS tasks where students only learn limited skills such as Data Analysis and Model Development via Kaggle\textsuperscript{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 a 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. For example, only one student capstone project in a 2 year DS program [15] had a large impact as it won a ACM SIG competition. We believe that if more students work collectively, they can develop large-scale innovative DS products can be developed.

To overcome the limitations of existing assignments and projects in DS courses, we envision a new kind of "synergistic modular assignments" to achieve the following main goals: \textit{Goal1:} Foster large-scale innovation and research as students collectively and iteratively improve upon each component of a real-world DS product every semester. \textit{Goal2:} Each student should comprehensively understand and work on most (if not all) steps involved in the complete workflow of the DS project. This means assignments should be decomposed into sub-tasks which can be performed by a large number of students in parallel. Such assignments would be especially useful for introductory DS courses to build a solid foundation in all the steps of DS and motivate students to pursue further DS research. This means that the assignments should be beginner-friendly requiring minimum prerequisite DS skills and domain expertise. 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 deploying such an assignment in a CS4 course on introductory Text Mining and Information Retrieval. The assignment enabled students to collaboratively build a potentially useful Expert Search Engine by completing four synergistic individual assignments (plus one final project) from which they learn the complete workflow of building a search engine including how to crawl data, how to evaluate algorithms, how to improve a search engine and how to develop a search engine interface. We describe how the assignment was designed and delivered to students and the grading process. We conclude with analyzing student submissions and some lessons learned during the experience.

2 RELATED WORK
We discuss related work in three main categories: 1) traditional assignments in DS; 2) techniques for teaching innovation in DS; 3) platforms supporting DS innovation and research.

\textsuperscript{1}https://www.kaggle.com/
Several works have documented example assignments and suggested best practices for assignment design in DS courses. For example, in [9], [13] authors provide example assignments and projects from their DS courses in Statistics curricula and Data Visualization courses respectively. Love et al. [11] discuss what makes a quality assignment in data mining and share a sample assignment using Open Data. However, current assignments have a limited scope because they focus on only few DS tasks e.g. Model Development. Additionally, assignments 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 subproblems 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.

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.

Many platforms have been developed to support DS research and innovation tasks. Notably, Human Computing and Crowdsourcing [14] 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 a CS4 course on Text Mining and Information Retrieval (IR). The course is targeted towards upper-classmen and graduate students having good programming skills. The core objectives 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. The course typically has 4 programming-based assignments and 1 team project.

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 MeTA toolkit [12] 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 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 to 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 researchers looking for potential collaborators with shared expertise or prospective students looking for advisors. Although there are some publicly available expert finders such as 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 as follows: Web crawling and indexing, developing and evaluating Retrieval Models, annotating Relevance Judgements for evaluation, and developing a web application for search ([18] provides detailed 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-application). 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) to cover one concept each. An additional module (MP2.2) was also released to familiarize students with the MeTA toolkit. Since it was the first offering of the assignment and 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 the 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.

³https://github.com/meta-toolkit/metapy
⁴https://www.mturk.com/
⁵https://kaggle.com/
⁶https://expertisefinder.com/
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 would 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 structure 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, students were required to choose Computer Science or an Engineering department. The chosen webpages were required to be in English language for simplicity.

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 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 scraper code.

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

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

3.2.3 Grading and submission compilation. Student submissions were first downloaded using GitLab API. In addition to checking for completeness, i.e. submission of 2 text files and scraper, we also checked if the 2 files were correctly formatted and if a unique university and department was chosen. To check format correctness, we programatically checked if the files were not empty and if number of URLs was equal to number of bios assuming each new line in the file has one URL per line. To accurately check if any 2 students chose the same university and department, we checked if there was significant overlap between their homepage URLs (>3 matching URLs).

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 \(^9\), 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 retrieval methods on the Cranfield dataset using T-test \(^10\) 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.

3.3.3 Grading. A unit test was provided as a GitLab CI Pipeline \(^11\) 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.

---

\(^{9}\)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

\(^{12}\)https://about.gitlab.com/product/continuous-integration/
would still be binary.

We asked students to log their queries in a shared spreadsheet. A built on the previous iteration of the assignment. It was delivered to learn to create and fine-tune retrieval methods for search.

3.5.1 Objective. The objective of the final module is for students to learn to create and fine-tune retrieval methods for search. It was graded manually based on completion.

3.5.2 Design. This part was designed as a search competition and built on the previous iteration of the assignment. It was delivered using GitLab in our virtual lab. Only the cranfield dataset was fully 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 with index 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. Highest weight was given to the Faculty dataset. A leaderboard (and supporting database) was maintained as shown in Figure 3. Students were given 2 weeks to complete the assignment. Multiple submissions were allowed.

3.5.3 Grading. We implemented a simple Okapi BM25 method 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, to create a useful search engine web-application, we released the task as a course project topic to be chosen by a team. Specifically, we wanted the following features in the application: 1. Search by area of expertise, 2. Filter results by University and Location of 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 search retrieval method from GitLab and use it for ranking the search results. In addition to creating the interface, the project required developing simple methods for extracting name and email from bios and finding locations of 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. It 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 across 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 as it can be used as a “base” to crawl more information like publications, Google Scholar profile, etc. 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 \(\approx 110\) unique queries were annotated, thus giving us \(\approx 2k\) total relevance judgments. Out of these only \(\approx 35\) 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 of 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 12 and Javascript. They used regex to extract emails, and Named Entity tagging 13 to extract faculty name. Further, they used Google Maps API 14 to get the locations of universities. GitLab API was used to upload a given student’s retrieval method identified by a GitLab project ID. The instructors only made minor enhancements to the UI to make it more "pretty".

5 LESSONS LEARNED AND CHALLENGES

In this section, we summarize the lessons learned and some challenges we faced during the experience.

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 also has 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 common infrastructure for delivery and grading of all assignment modules, and hosting all student code repositories and large datasets makes it convenient for both students and instructors to collaboratively work on the large assignment. As mentioned before, we used a virtual lab based on CLaDS for deploying the assignment. Similar cloud-based labs for data science can be used by other instructors too.

2. Build the new assignment upon existing assignments: Developing a large assignment with multiple modules can be time consuming. Extending an existing assignment allows to reuse its parts including grading and delivery and reduces the time required to develop the new assignment. We extended a previous search competition-based assignment. Since many Data Science courses today tend to have similar competition-based assignments, using them as starting points should be feasible.

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 required to understand and work on it.

   In our case, finding expert faculty for choosing advisors and universities was a very relevant problem for students and the CS/engineering domain was especially familiar to them. The problem had a low entry barrier; students knew how and where to find university websites, they were able to judge whether a faculty bio was relevant to the query etc.

4. Design assignment modules carefully: Firstly, the problem chosen should be decomposed into modules that cover all 3 main steps of data science: Data Collection, Data Analysis and Application Development. 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 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 to scraped a unique university listing. In MP2.3, a maximum of 10 students was allowed to judge the same query. This may also allow instructors to provide a complete sample solution with each module that students can use as a reference.

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, 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 aimed 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:

   - 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. Stricter grading is one way for preventing such issues.

   - 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 some 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.

   - Extensive documentation should be maintained to sustain the assignment over time: We realized this need as we started thinking about releasing the assignment in another semester. Students should be able to easily understand previous work and build upon it. Instructors and students could collaboratively maintain the documentation through sites like Wiki. Students should also be asked to follow standard coding styles. This would additionally help prepare students better for jobs in industry.

6 CONCLUSIONS

In this paper, we proposed a new kind of 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 to 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 a highly 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 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, and Application Development, similar modular assignments can be adapted by other data science courses too.

REFERENCES


