Instroogle

Danny Swisher
Tim Crossley
Patrick Healy
Matt York

December 17, 2009

Instroogle

Aggregate Instructor Information

Information about the University of Washington's colleges, departments, course offerings, and instructors is spread across many different internet services. Students who try to learn about their instructors from many different angles, such as reviews and teaching performance, struggle by searching across multiple websites and search engines. This information is critical in deciding which courses to take and which instructor to take them from. Even if a student spent the time finding these resources, there is no easy way to analyze all the data and compare between different instructors, departments, and courses.

In Autumn 2009 we developed Instroogle.com, a site dedicated to aggregating, organizing, and presenting the valuable information about UW instructors in one place. Our service is unique and allows students to learn more about their university and make well-informed decisions when planning their class schedules.

We learned a lot from building Instroogle, such as how to handle data collection processes that take multiple hours and the importance of proper testing. This was a very positive experience, and the amount of effort and amount of learning that went into this project is immediately apparent went viewing the ridiculously awesome application at Instroogle.com.

System Overview

Entities

Instroogle lets users compare three types to each other: instructors, departments, and courses. We compile metric averages such as salaries and course evaluations for all the instructors that teach a given course, all the instructors in a department, and for each individual instructor. This abstraction allows a user to compare Stuart Reges to the Dance department, compare all the teachers of Operating Systems to the Business School, or compare the instructor averages of CSE 142 against Hank Levy.

Primary Functions

Instroogle lets users find public instructor information and compare instructors to each other, departments, courses, or the University of Washington as a whole.

Find

Instroogle is a search engine at heart. Our main goal of the project was to allow students to learn more about the professors teaching the courses at the University of Washington. Our Find technology allows users to search for data-rich profiles on instructors, departments, or courses.

Compare

Instroogle allows users to compare instructors, courses, and departments side by side. This technology may only be simulated elsewhere by opening many different pages of UW course evaluations or RateMyProfessors reviews. Instroogle's comparison engine offers a service found nowhere else.

Leaderboards

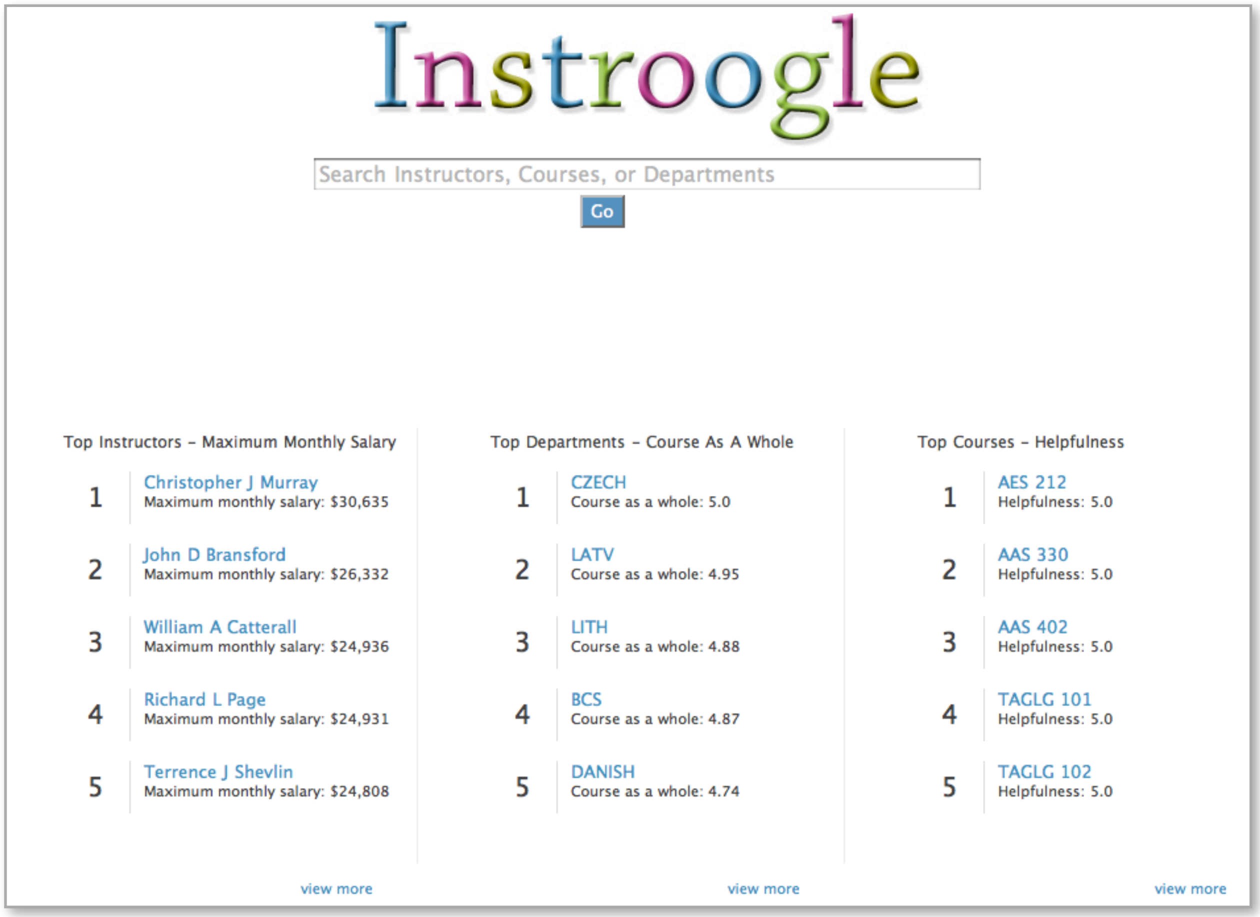
Instroogle's leaderboards page allows users to view rankings of courses, departments, and instructors based on a wide variety of metrics. From maximum monthly salary to highest teaching effectiveness, users can stack up instructors, courses, and departments against each other to learn more about their university.

Development Environment

We built our entire system in Ruby on Rails, including many crawlers and number crunchers. Though we developed on local machines (mostly Macs with one bad apple running Windows), Instroogle.com is running on a Dreamhost virtual host through Phusion Passenger, an Apache module to run Rails apps. We are currently running on a SQLite3 database, though because SQLite3 is a very slow and lightweight database management system, we will transfer to MySQL very soon. We used the subversion version control system.

User Interface

Homepage



Top Instructors – Maximum Monthly Salary

1	Christopher J Murray Maximum monthly salary: $30,635
2	John D Bransford Maximum monthly salary: $26,332
3	William A Catterall Maximum monthly salary: $24,936
4	Richard L Page Maximum monthly salary: $24,931
5	Terrence J Shevlin Maximum monthly salary: $24,808

view more

Top Departments – Course As A Whole

1	CZECH Course as a whole: 5.0
2	LATV Course as a whole: 4.95
3	LITH Course as a whole: 4.88
4	BCS Course as a whole: 4.87
5	DANISH Course as a whole: 4.74

view more

Top Courses – Helpfulness

1	AES 212 Helpfulness: 5.0
2	AAS 330 Helpfulness: 5.0
3	AAS 402 Helpfulness: 5.0
4	TAGLG 101 Helpfulness: 5.0
5	TAGLG 102 Helpfulness: 5.0

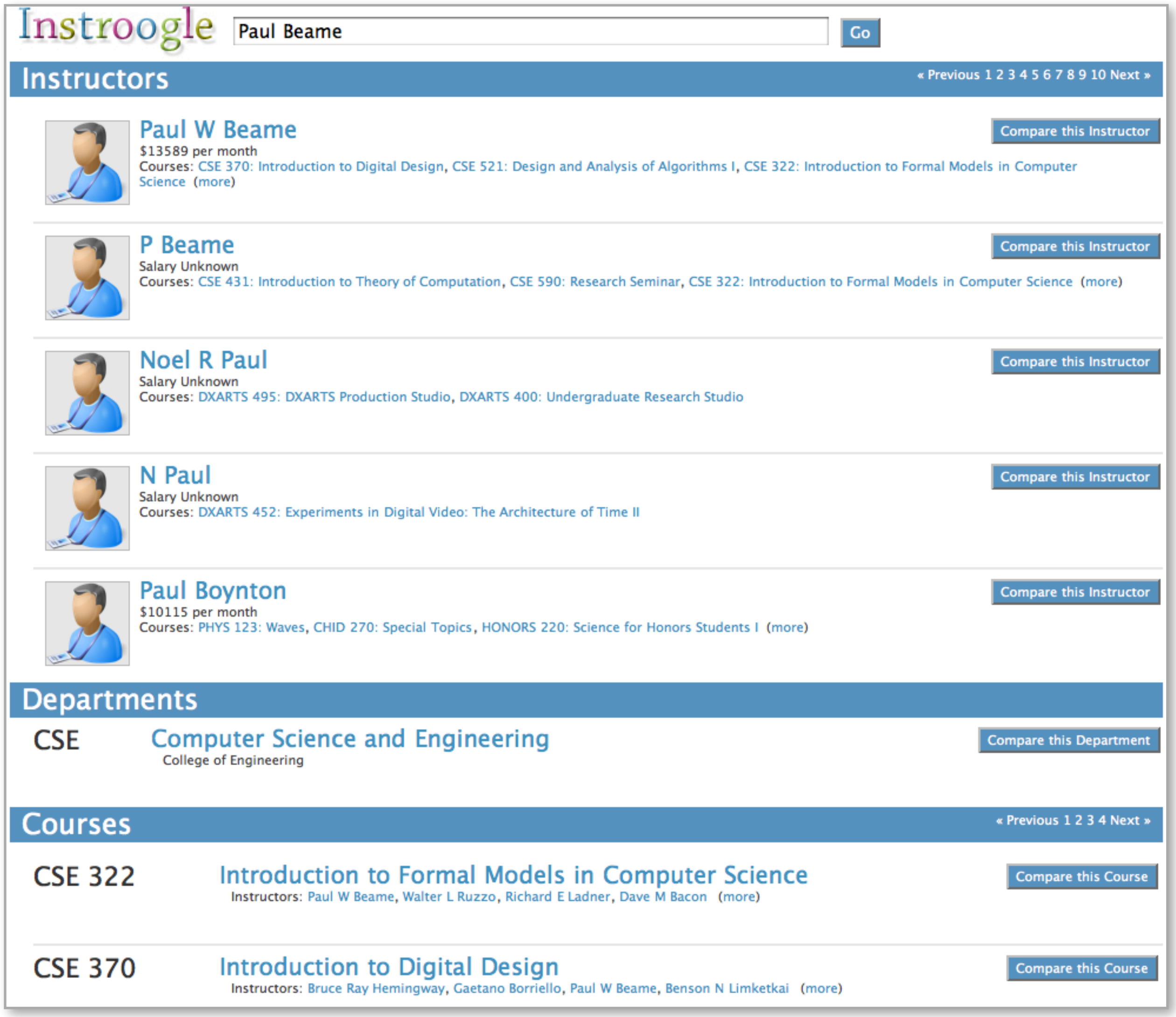
view more

Figure 1: Instroogle Homepage

Our homepage is designed to be as simple and minimalistic as possible. It features our main site navigation at the top (“Find”, “Compare”, “Leaderboards”), a search bar which features autocomplete results, and three leaderboard samples – one for instructors, courses, and departments – sorted by randomly-selected metrics.

Search Results

Because Instroogle supports three main types of entities, we can structure our search results uniquely around those types.



Instroogle | Paul Beame | Go

Instructors « Previous 1 2 3 4 5 6 7 8 9 10 Next »

Paul W Beame — Compare this Instructor
$13589 per month
Courses: CSE 370: Introduction to Digital Design, CSE 521: Design and Analysis of Algorithms I, CSE 322: Introduction to Formal Models in Computer Science (more)

P Beame — Compare this Instructor
Salary Unknown
Courses: CSE 431: Introduction to Theory of Computation, CSE 590: Research Seminar, CSE 322: Introduction to Formal Models in Computer Science (more)

Noel R Paul — Compare this Instructor
Salary Unknown
Courses: DXARTS 495: DXARTS Production Studio, DXARTS 400: Undergraduate Research Studio

N Paul — Compare this Instructor
Salary Unknown
Courses: DXARTS 452: Experiments in Digital Video: The Architecture of Time II

Paul Boynton — Compare this Instructor
$10115 per month
Courses: PHYS 123: Waves, CHID 270: Special Topics, HONORS 220: Science for Honors Students I (more)

Departments

CSE Computer Science and Engineering — Compare this Department
College of Engineering

Courses « Previous 1 2 3 4 Next »

CSE 322 Introduction to Formal Models in Computer Science — Compare this Course
Instructors: Paul W Beame, Walter L Ruzzo, Richard E Ladner, Dave M Bacon (more)

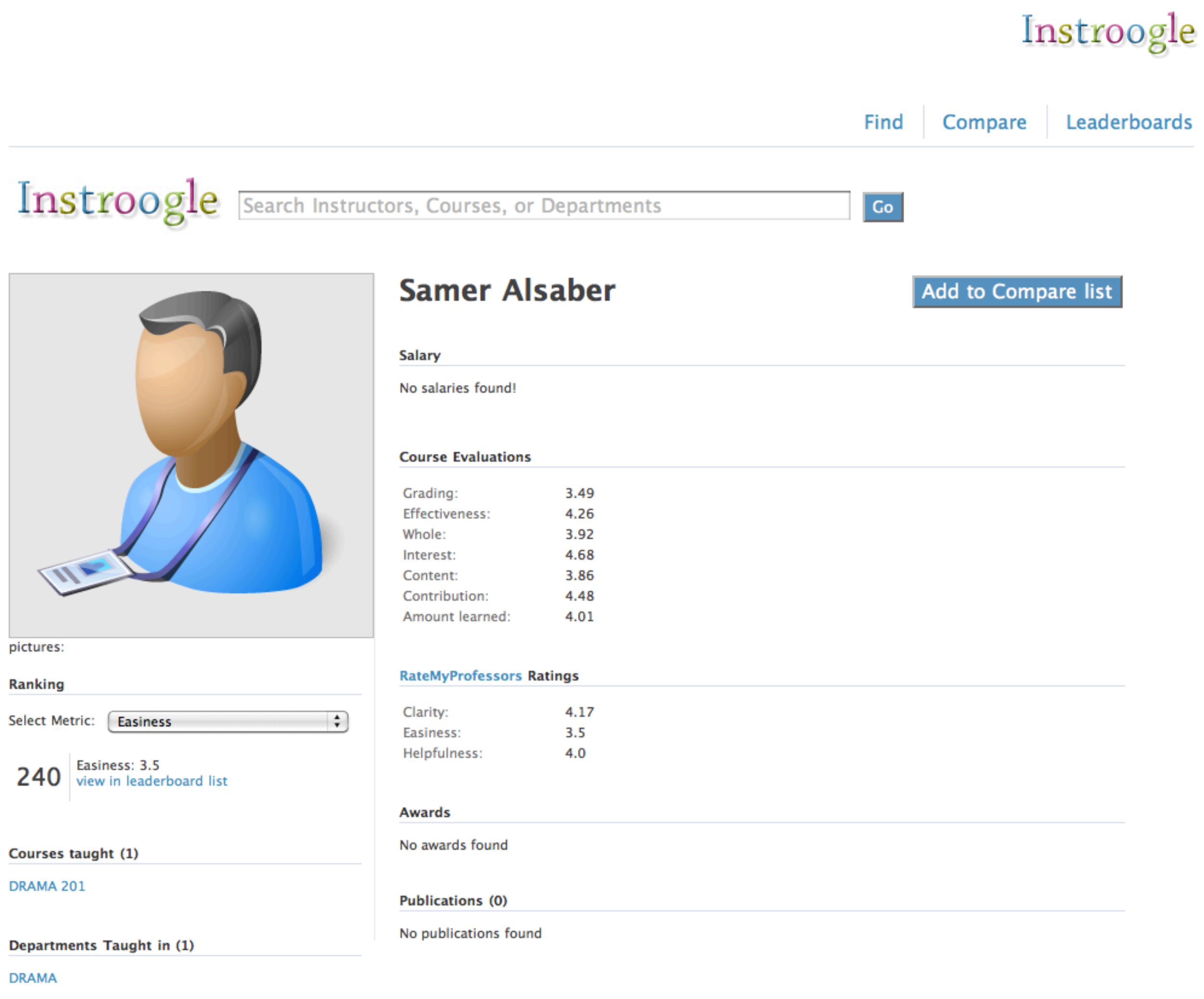
CSE 370 Introduction to Digital Design — Compare this Course
Instructors: Bruce Ray Hemingway, Gaetano Borriello, Paul W Beame, Benson N Limketkai (more)

Figure 2: Search page

In the above search-results page, users can see all things related to “Paul Beame” including his homepage, the department he teaches in, and the courses he has taught. Each result links to the appropriate profile pages.

Profile Pages

Instroogle has three main classes: Instructors, Courses, and Departments. Each one of these types has a profile page with dynamically-generated content based on the respective object one is viewing. Here is an example of an instructor home page.



Samer Alsaber

pictures:

Ranking

Select Metric: Easiness

240 Easiness: 3.5
view in leaderboard list

Courses taught (1)

DRAMA 201

Departments Taught in (1)

DRAMA

Salary

No salaries found!

Course Evaluations

Grading:	3.49
Effectiveness:	4.26
Whole:	3.92
Interest:	4.68
Content:	3.86
Contribution:	4.48
Amount learned:	4.01

RateMyProfessors Ratings

Clarity:	4.17
Easiness:	3.5
Helpfulness:	4.0

Awards

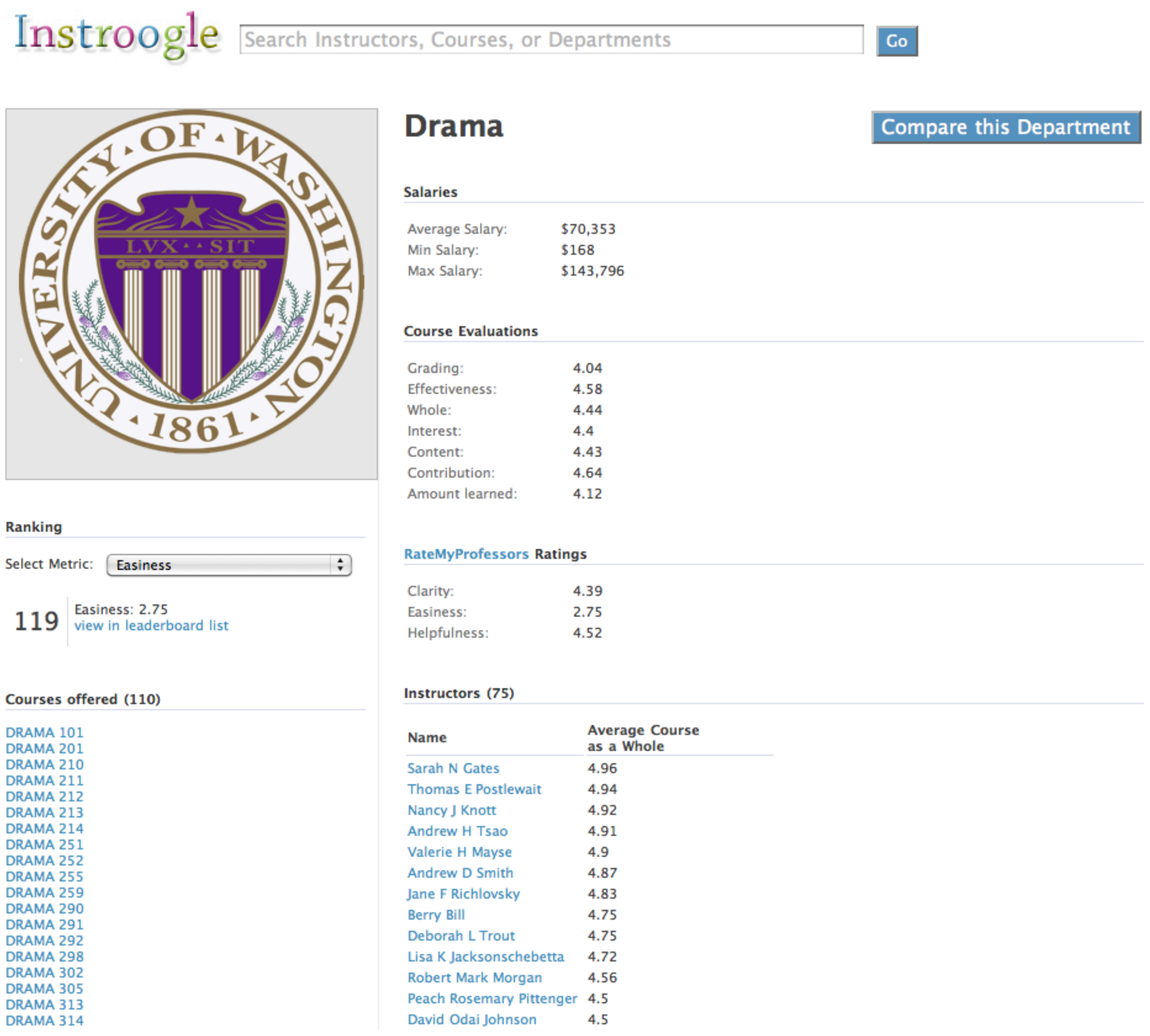
No awards found

Publications (0)

No publications found

Figure 3: Instructor Profile Page

This page displays information for the specified instructor, such as name, courses this instructor has taught, and aggregated data from RMP and course evaluations. If this instructor had won any awards, they would be listed under the "Awards" header. And similarly, if this instructor had any publications that our publications crawler had found, pointers to said publications would be given under the "Publications" header. Under the "Courses taught" section and the "Departments taught in" section, users have pointers to other data in Instroogle. If we proceed to the "DRAMA" department from the above page, we are presented with an example of a Department profile page.



Instroogle Search Instructors, Courses, or Departments Go

Drama

Compare this Department

Salaries

Average Salary:	$70,353
Min Salary:	$168
Max Salary:	$143,796

Course Evaluations

Grading:	4.04
Effectiveness:	4.58
Whole:	4.44
Interest:	4.4
Content:	4.43
Contribution:	4.64
Amount learned:	4.12

Ranking

Select Metric: Easiness

119 Easiness: 2.75
view in leaderboard list

RateMyProfessors Ratings

Clarity:	4.39
Easiness:	2.75
Helpfulness:	4.52

Courses offered (110)

DRAMA 101
DRAMA 201
DRAMA 210
DRAMA 211
DRAMA 212
DRAMA 213
DRAMA 214
DRAMA 251
DRAMA 252
DRAMA 255
DRAMA 259
DRAMA 290
DRAMA 291
DRAMA 292
DRAMA 298
DRAMA 302
DRAMA 305
DRAMA 313
DRAMA 314

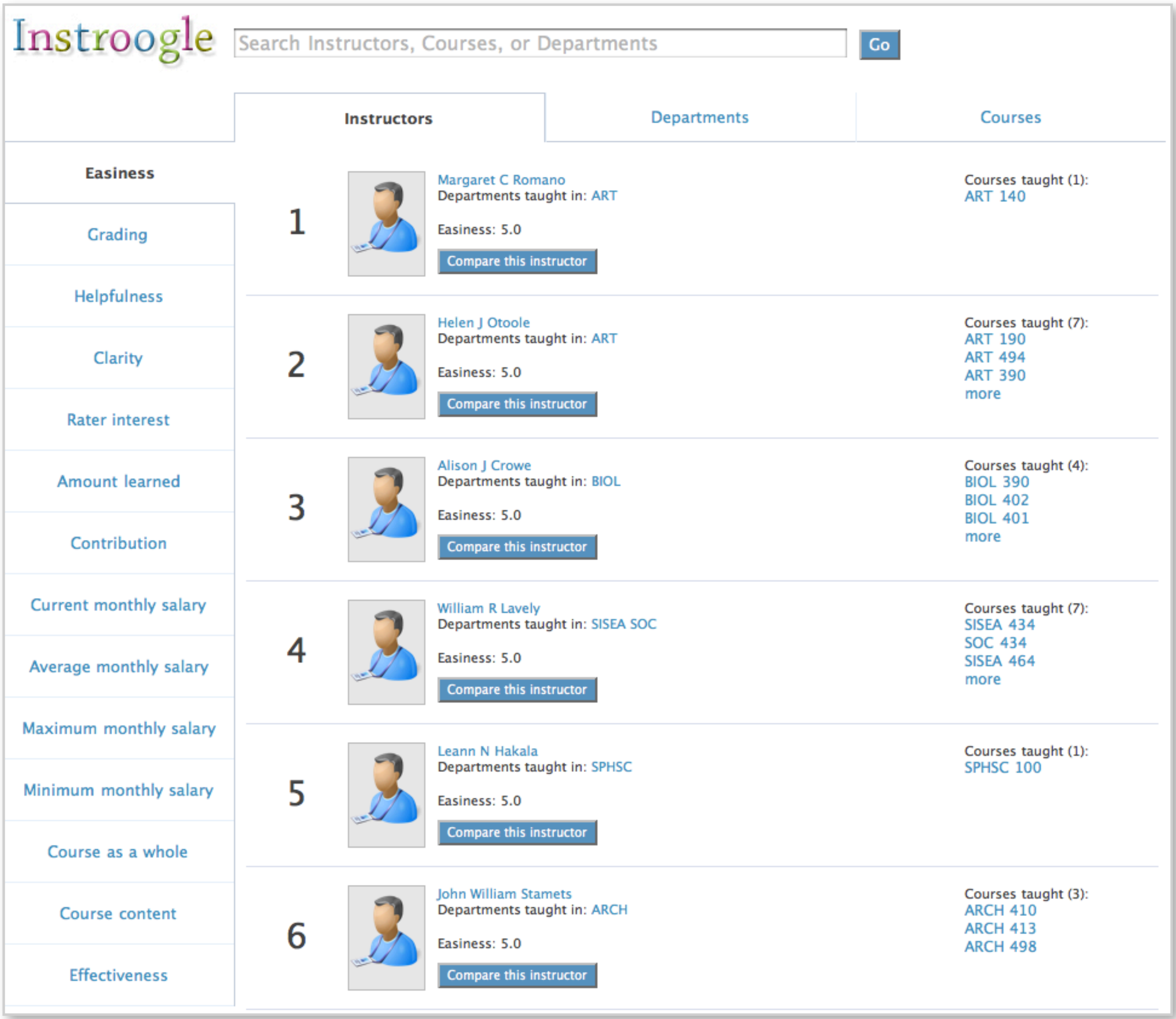
Instructors (75)

Name	Average Course as a Whole
Sarah N Gates	4.96
Thomas E Postlewait	4.94
Nancy J Knott	4.92
Andrew H Tsao	4.91
Valerie H Mayse	4.9
Andrew D Smith	4.87
Jane F Richlovsky	4.83
Berry Bill	4.75
Deborah L Trout	4.75
Lisa K Jacksonschebetta	4.72
Robert Mark Morgan	4.56
Peach Rosemary Pittenger	4.5
David Odai Johnson	4.5

Figure 4: Department Profile Page

Department profile pages display the courses that are offered in the course as well as all the instructors that have taught in that department, sorted by their average course-as-a-whole course evaluation scores. The course evaluation and RateMyProfessors review data visible on this page and course profile pages is simply an aggregation of every instructor who has taught in the respective department or course. Since we only have data about individual instructors, this is the best way to compare departments and courses. The course profile page is very similar to the department and instructor home page except that the profile image is coded to display the image of the instructor who has the highest rating for that course.

Leaderboards



Instroogle

Search Instructors, Courses, or Departments

Go

Instructors | Departments | Courses

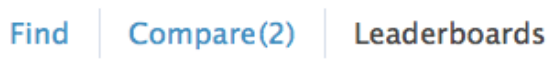
Easiness
Grading
Helpfulness
Clarity
Rater interest
Amount learned
Contribution
Current monthly salary
Average monthly salary
Maximum monthly salary
Minimum monthly salary
Course as a whole
Course content
Effectiveness

1	Margaret C Romano Departments taught in: ART Easiness: 5.0 Compare this instructor	Courses taught (1): ART 140
2	Helen J Otoole Departments taught in: ART Easiness: 5.0 Compare this instructor	Courses taught (7): ART 190 ART 494 ART 390 more
3	Alison J Crowe Departments taught in: BIOL Easiness: 5.0 Compare this instructor	Courses taught (4): BIOL 390 BIOL 402 BIOL 401 more
4	William R Lavely Departments taught in: SISEA SOC Easiness: 5.0 Compare this instructor	Courses taught (7): SISEA 434 SOC 434 SISEA 464 more
5	Leann N Hakala Departments taught in: SPHSC Easiness: 5.0 Compare this instructor	Courses taught (1): SPHSC 100
6	John William Stamets Departments taught in: ARCH Easiness: 5.0 Compare this instructor	Courses taught (3): ARCH 410 ARCH 413 ARCH 498

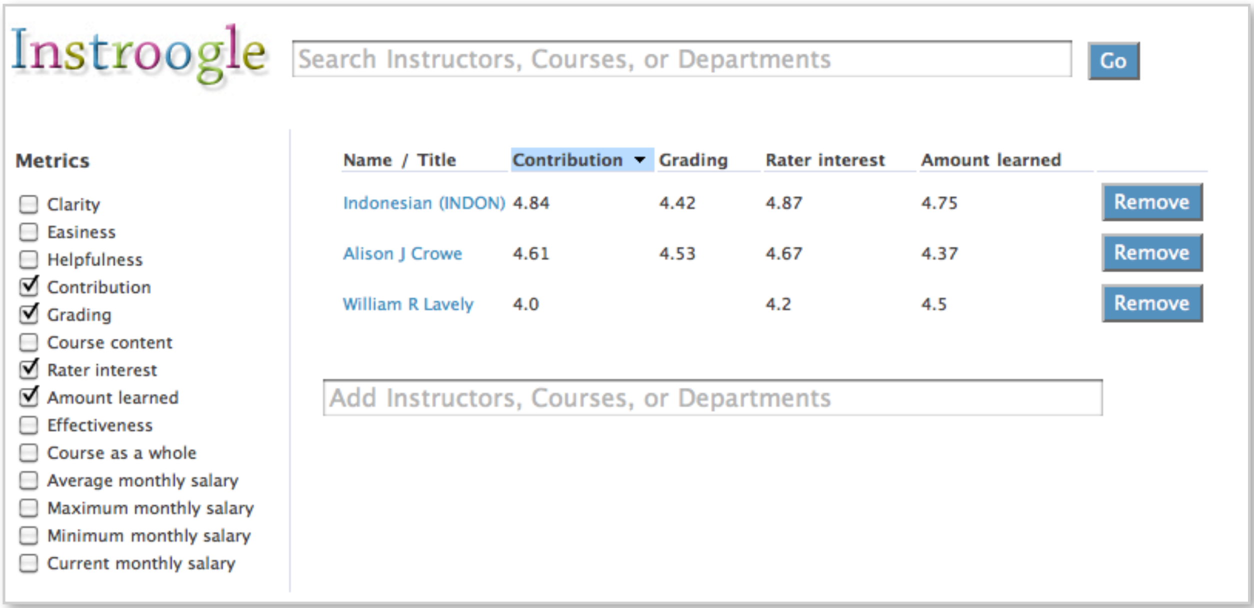
Figure 5: Leaderboard Page

In the above example, every instructor in Instroogle's system is ranked according to the metric "Easiness," a metric which we pull from RateMyProfessors.com. The results are paginated with 10 results per page. Users can change the metric of the ranking by simply clicking the links on the left side, or change class being ranked to department or course by clicking the links at the top of the page.

You may have noticed the "Compare" buttons from the profile pages and leaderboards page. Instroogle also features the ability to compare individual instances of courses, departments, and instructors directly. Clicking "Compare this instructor" or "Compare" adds the specified object to a "comparison cart" for the user's current session. The number of items in the comparison cart is displayed at the top of the page in the central-navigation area.



Clicking on "Compare (2)" brings the user to the comparison page which features list of objects sortable by all the different metrics Instroogle calculates.



- [ ] Clarity
- [ ] Easiness
- [ ] Helpfulness
- [x] Contribution
- [x] Grading
- [ ] Course content
- [x] Rater interest
- [x] Amount learned
- [ ] Effectiveness
- [ ] Course as a whole
- [ ] Average monthly salary
- [ ] Maximum monthly salary
- [ ] Minimum monthly salary
- [ ] Current monthly salary

Name / Title	Contribution ▾	Grading	Rater interest	Amount learned	
Indonesian (INDON)	4.84	4.42	4.87	4.75	Remove
Alison J Crowe	4.61	4.53	4.67	4.37	Remove
William R Lavely	4.0		4.2	4.5	Remove

Figure 6: Compare Page

On this page, users can add new instructors, departments, or courses directly and sort the list based on the metrics they've chosen on the left.

Sample User Scenario

Scenario 1: Find the Right Teacher for a Course

Annie is planning on taking Dance 101 next quarter. She's heard good things about the course from friends of hers who took the course. However, her friends have opposing opinions about their professors who taught the course. Lisa, who took the course from Tonya Lockyer, like the way her instructor taught it, but Becky, who took Dance 101 from Jamie Hall, didn't. Annie goes to the Instroogle homepage and searches for "Dance 101".

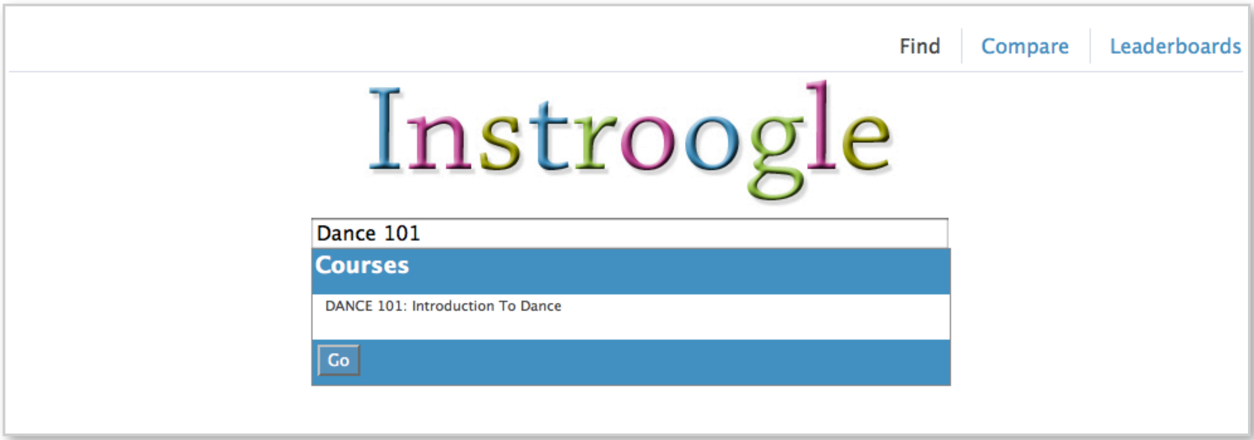
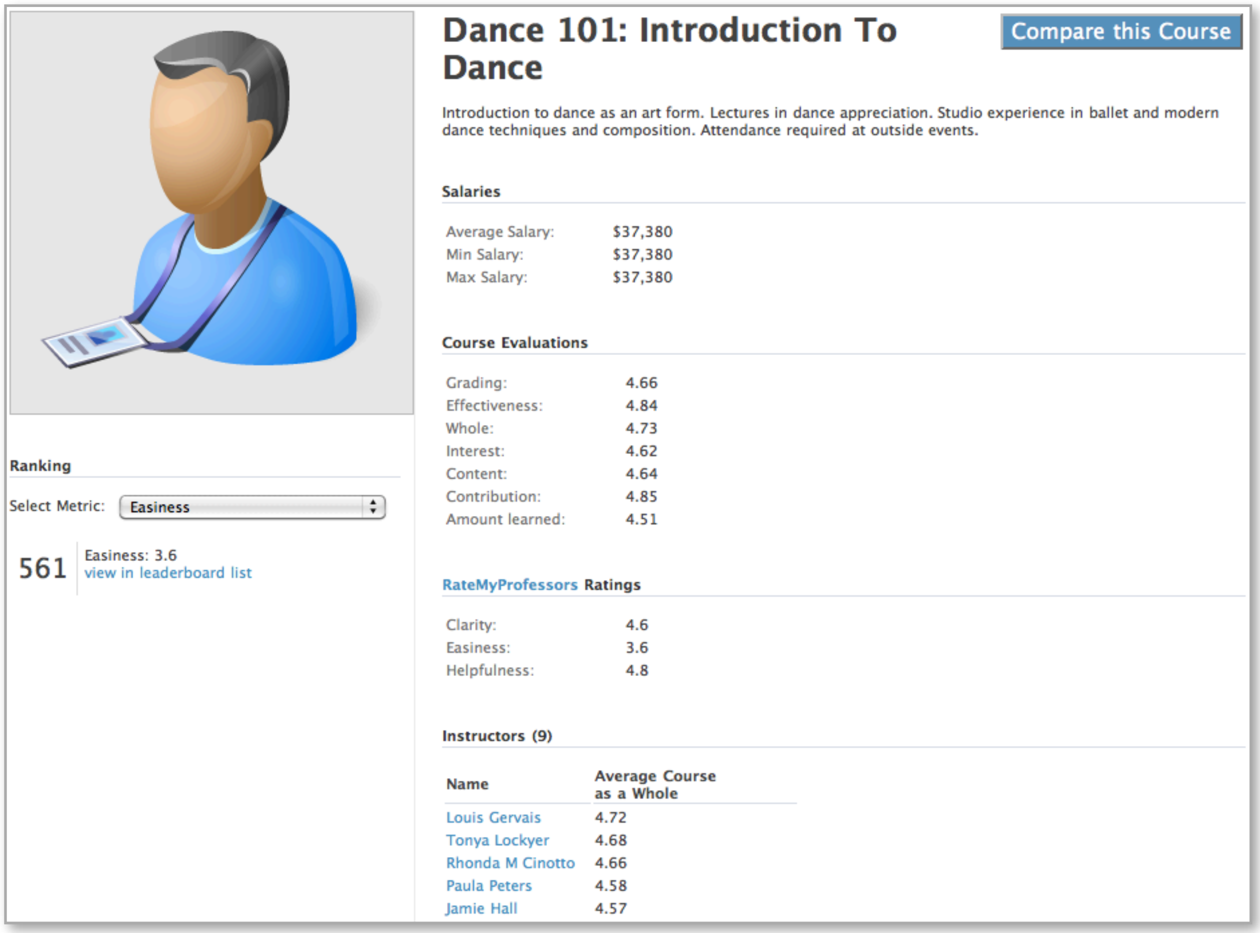


Figure 7: Instroogle homepage, query "Dance 101"

Clicking "Dance 101" in the autocomplete results brings her to Dance 101's homepage.



Dance 101: Introduction To Dance

Introduction to dance as an art form. Lectures in dance appreciation. Studio experience in ballet and modern dance techniques and composition. Attendance required at outside events.

Salaries

Average Salary:	$37,380
Min Salary:	$37,380
Max Salary:	$37,380

Course Evaluations

Grading:	4.66
Effectiveness:	4.84
Whole:	4.73
Interest:	4.62
Content:	4.64
Contribution:	4.85
Amount learned:	4.51

RateMyProfessors Ratings

Clarity:	4.6
Easiness:	3.6
Helpfulness:	4.8

Instructors (9)

Name	Average Course as a Whole
Louis Gervais	4.72
Tonya Lockyer	4.68
Rhonda M Cinotto	4.66
Paula Peters	4.58
Jamie Hall	4.57

Ranking

Select Metric: Easiness

561 Easiness: 3.6
view in leaderboard list

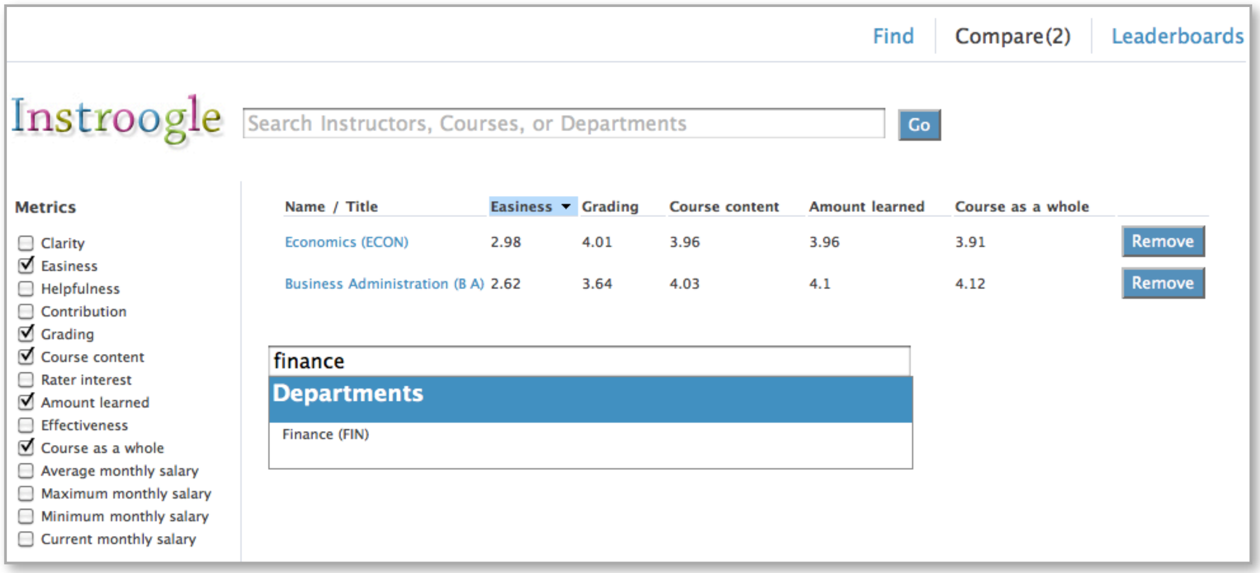
Figure 8: Dance 101 profile page

From this screen, Annie can see the ranking of instructors who have taught Dance 101 before. Annie decides to enroll for Tonya Lockyer's section because she has a higher rating than Jamie Hall.

Without Instroogle, Annie would have had to consult RateMyProfessors.com and average all the reviews for each potential instructor of Dance 101 to make this decision.

Scenario 2: Comparisons

Jake is finishing up his freshman year at the University of Washington and trying to decide what to major in. He knows that he is a wiz with money and loves learning about business, and has narrowed his choices down two three departments: Economics, Business Administration, and Finance. Jake uses Instroogle's comparison engine to compare the three departments. From the Comparison page, he adds the three departments.



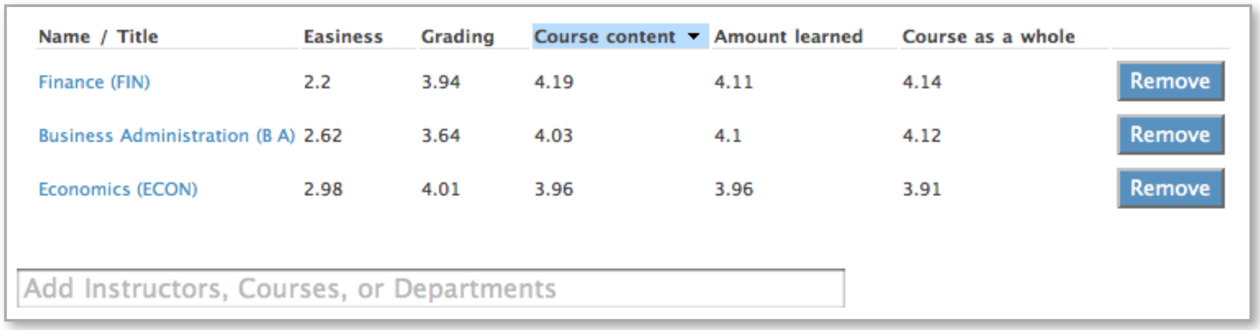
Metrics

- [ ] Clarity
- [✓] Easiness
- [ ] Helpfulness
- [ ] Contribution
- [✓] Grading
- [✓] Course content
- [ ] Rater interest
- [✓] Amount learned
- [ ] Effectiveness
- [✓] Course as a whole
- [ ] Average monthly salary
- [ ] Maximum monthly salary
- [ ] Minimum monthly salary
- [ ] Current monthly salary

Name / Title	Easiness ▾	Grading	Course content	Amount learned	Course as a whole	
Economics (ECON)	2.98	4.01	3.96	3.96	3.91	Remove
Business Administration (B A)	2.62	3.64	4.03	4.1	4.12	Remove

Figure 9: Compare Page

After adding the 3 prospective departments, Jake adds the metrics he wants to use for comparison. He can tell by sorting on “Easiness” that Economics is the easiest major to study in. But when Jake sorts on “Course content” to find the department with the courses that have the highest-rated course content on average, he learns that Finance is the best.



Name / Title	Easiness	Grading	Course content ▾	Amount learned	Course as a whole	
Finance (FIN)	2.2	3.94	4.19	4.11	4.14	Remove
Business Administration (B A)	2.62	3.64	4.03	4.1	4.12	Remove
Economics (ECON)	2.98	4.01	3.96	3.96	3.91	Remove

Figure 10: Sort on “Course Content”

With all this data at Jake’s fingertips, he decides to major in Economics because he feels that majoring in ECON is the best opportunity for him to have an enjoyable time in college and still learn a lot about money.

Without Instroogle, there is no way Jake could have compared departments – the technology does not exist elsewhere.

Data Flow

We get specific data from many different data sources on the web. We divided the data crawlers (a very loose definition of the word “crawl” since we mostly just scrape) into different modules based on the pages they scrape. Once we have acquired all the data, we then run the module MetricMaker that computes and compiles the data into a Metrics table. A data flow with the dependencies can be seen in the diagram below. Most of the crawlers depend on having a list of instructors, hence the reason Names crawler, Time Schedule Crawler, and Evaluation Crawler go first.

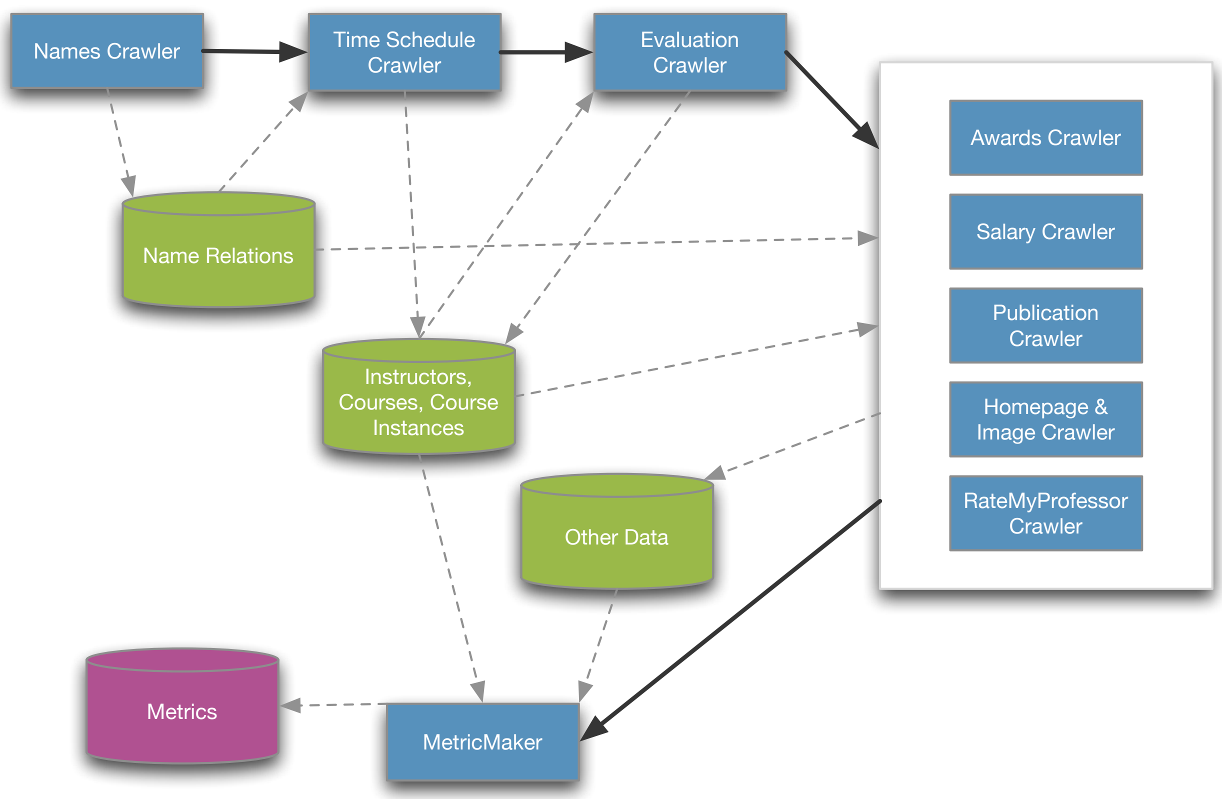
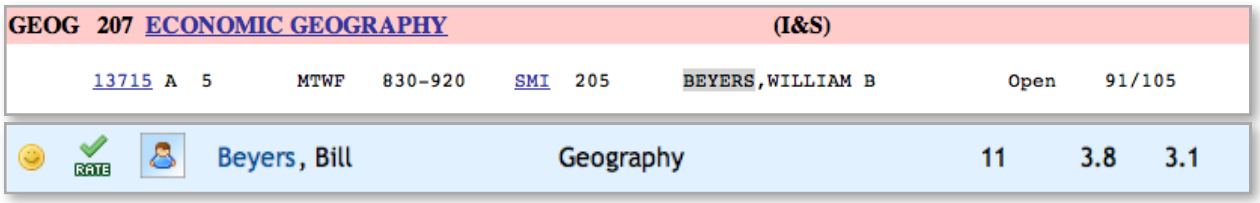


Figure 11: Data Pipeline

Name Normalization

Inconsistent naming was the biggest problem with our crawlers.



GEOG 207 ECONOMIC GEOGRAPHY						(I&S)			
13715	A	5	MTWF	830-920	SMI 205	BEYERS,WILLIAM B		Open	91/105
			Beyers, Bill		Geography		11	3.8	3.1

Figure 12: Inconsistent Naming Between RateMyProfessors and Time Schedule

An instructor might be listed as “William B Beyers” in the Time Schedule, but listed as “Bill Beyers” on RateMyProfessors. To solve this, we built a name normalization algorithm that utilizes nicknames and common spellings of the same name.

```
find_by_name name
     find all instructors with given first name and last name
     return entry if only one entry
     If there are no entries, try all related names using BehindTheName
          returning if one found
     If still no entries, try remove punctuation from the name
          return if one found
     If more than one result, try middle initial
          return if one found
     If still more than one result, write to common names log and return nil
```

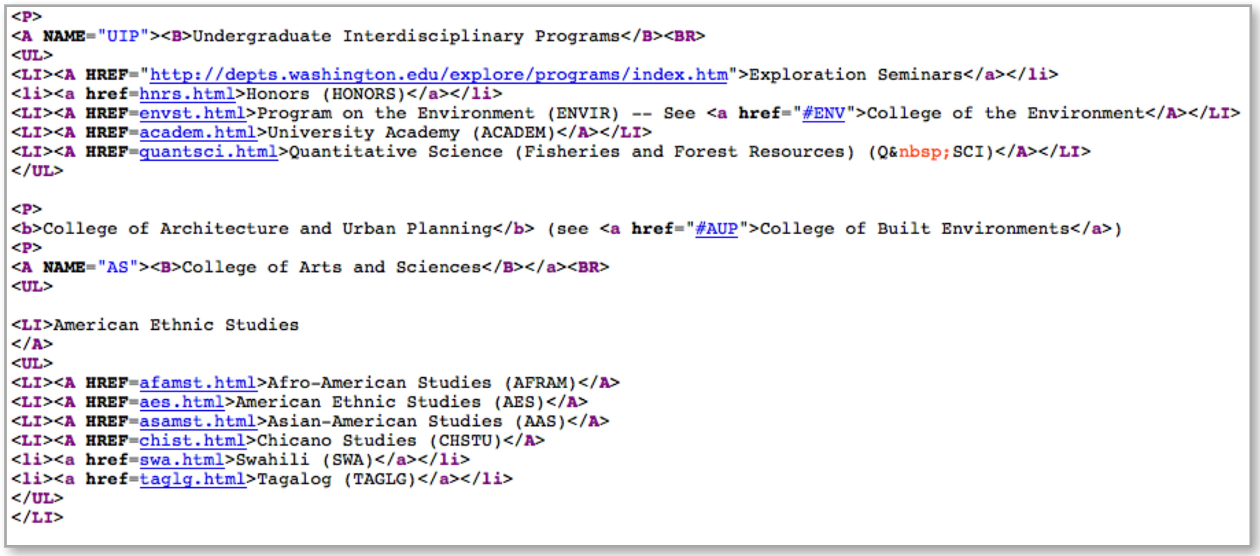
This algorithm is able to resolve names such as “Marty Stepp” to “Martin Stepp”, “Stu Reges” to “Stuart Reges”, and “Yong-Pin Zhou” to “Yongpin Zhou”.

The Names Crawler crawls http://www.behindthename.com/ for maps of names to related names. Related names are names that are used interchangeably or as common nicknames. For example people named Richard often prefer to be called Dick, this example clearly shows how these related names are much more complicated than simply shortened or simplified versions of names like Dan and Daniel.

Time Schedule Crawler

The first and most important web crawler we have is our crawler of UW’s Time Schedule, which contains course data for every quarter between Autumn 2006 and Winter 2010. Our crawler screen scrapes data from every available quarter for information such as courses, departments, instances of courses, and of course instructors for those courses. We also grab every course’s description and load all of this data into our relational database. With this data we are able to tell which courses any given department offers (e.g. CSE offers 142, 143, …), and additionally the instructors who taught those courses for every quarter. The aggregation of this data is already unique and very useful; discovering this data by hand right now would be tedious.

UW’s time schedule HTML contains many flaws which really impeded the progress of the crawler for many weeks. For example, the HTML structure is invalid in some ways, such as the following example.

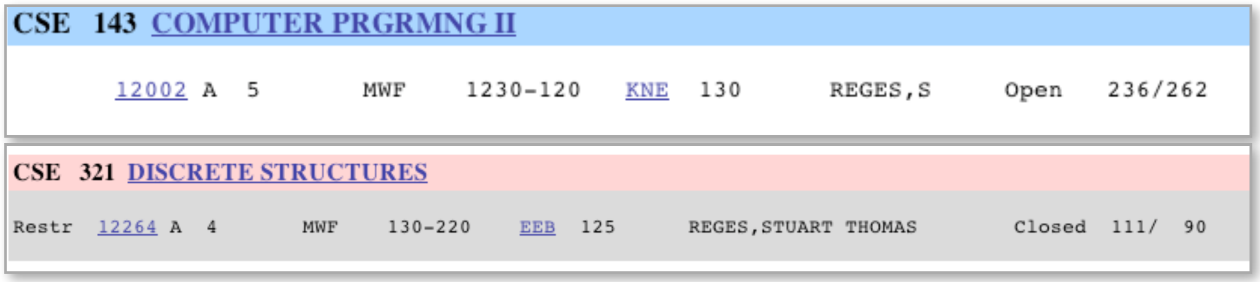


```
<P>
<A NAME="UIP"><B>Undergraduate Interdisciplinary Programs</B><BR>
<UL>
<LI><A HREF="http://depts.washington.edu/explore/programs/index.htm">Exploration Seminars</a></li>
<li><a href=hnrs.html>Honors (HONORS)</a></li>
<LI><A HREF=envst.html>Program on the Environment (ENVIR) -- See <a href="#ENV">College of the Environment</A></LI>
<LI><A HREF=academ.html>University Academy (ACADEM)</A></LI>
<LI><A HREF=quantsci.html>Quantitative Science (Fisheries and Forest Resources) (Q&nbsp;SCI)</A></LI>
</UL>

<P>
<b>College of Architecture and Urban Planning</b> (see <a href="#AUP">College of Built Environments</a>)
<P>
<A NAME="AS"><B>College of Arts and Sciences</B></a><BR>
<UL>

<LI>American Ethnic Studies
</A>
<UL>
<LI><A HREF=afamst.html>Afro-American Studies (AFRAM)</A>
<LI><A HREF=aes.html>American Ethnic Studies (AES)</A>
<LI><A HREF=asamst.html>Asian-American Studies (AAS)</A>
<LI><A HREF=chist.html>Chicano Studies (CHSTU)</A>
<li><a href=swa.html>Swahili (SWA)</a></li>
<li><a href=taglg.html>Tagalog (TAGLG)</a></li>
</UL>
</LI>
```

Figure 13: Bad HTML Formatting

The HTML is structured in such a way that the beginning of a <P> tag signifies the beginning of a new college (e.g. College of Arts and Sciences). However, the page does not close open <P> tags before starting new ones, which made XPath queries especially difficult. Luckily, the version of Hpricot had a bug that turned out to be a feature: it recognized the beginning of a new <P> tag as also the end of a previously-opened <P> tag.

In addition to battling with poorly structured HTML, we also came across naming conflicts between different quarters.



```
CSE   143  COMPUTER PRGRMNG II
       12002 A  5       MWF    1230-120    KNE   130      REGES,S      Open    236/262

CSE  321  DISCRETE STRUCTURES
Restr   12264 A  4     MWF    130-220    EEB   125     REGES,STUART THOMAS      Closed  111/  90
```

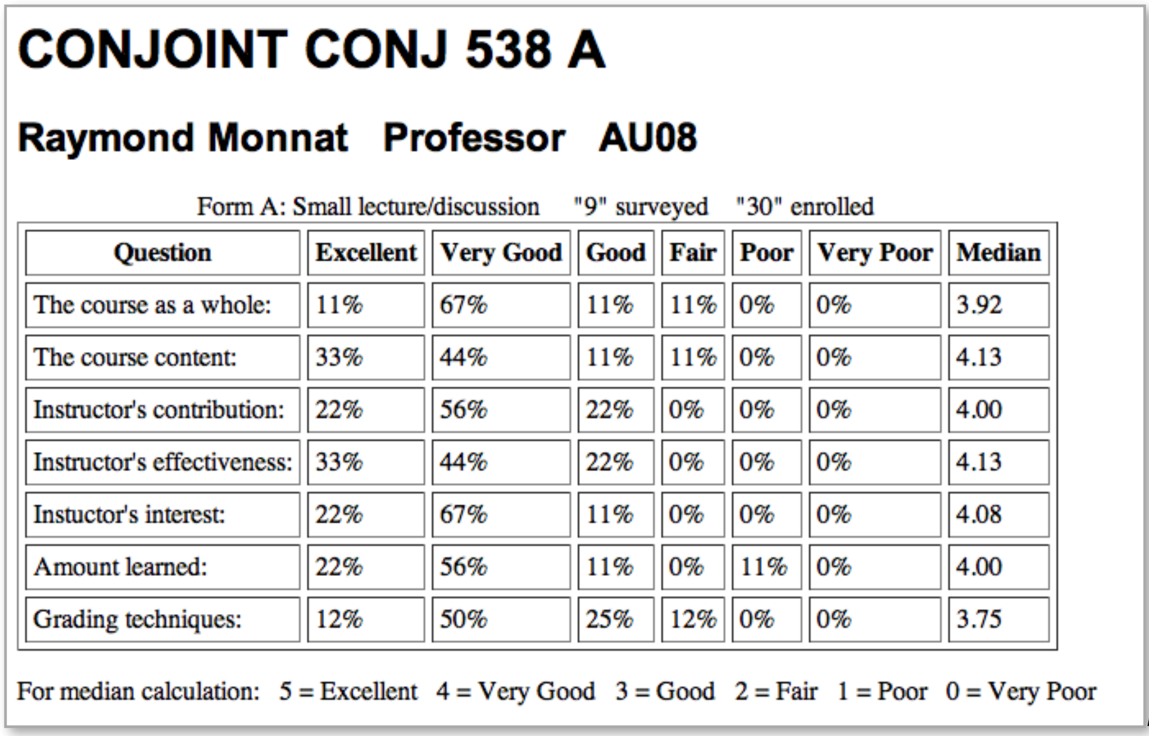
Figure 14: Naming Conflicts

Clearly Stuart Reges taught both of the courses listed above and yet the time schedule, which one would expect to have one central database for producing the output above, prints different parts of an instructor's name for no apparent reason. These inconsistencies were very troublesome. In fact, our time schedule crawler performs check against about 8 different complex regular expressions to attempt to parse an instructor's name out of the HTML code. We eventually consulted our BehindTheName data, a "nickname database", which helped resolve some of the inconsistencies. See Name Normalization for more detail.

After our time schedule crawler loaded our database with instructors, we were able to consult a vast array of other sources for information about UW instructors.

Course Evaluation Crawler

The course evaluation crawler grabs all the compiled student course evaluation information from the Course Evaluation Catalog (CEC). To get access to the CEC you need to authenticate using the UW's UWNetID PubCookie authentication system. After spending multiple days trying to right a script that simulated a user login and running into all sorts of "your browser isn't running javascript" errors and being led in circles, we logged in manually and used a Firefox extension called DownThemAll to download the CEC in bulk and crawled it locally.



CONJOINT CONJ 538 A

Raymond Monnat   Professor   AU08

Form A: Small lecture/discussion   "9" surveyed   "30" enrolled

Question	Excellent	Very Good	Good	Fair	Poor	Very Poor	Median
The course as a whole:	11%	67%	11%	11%	0%	0%	3.92
The course content:	33%	44%	11%	11%	0%	0%	4.13
Instructor's contribution:	22%	56%	22%	0%	0%	0%	4.00
Instructor's effectiveness:	33%	44%	22%	0%	0%	0%	4.13
Instuctor's interest:	22%	67%	11%	0%	0%	0%	4.08
Amount learned:	22%	56%	11%	0%	11%	0%	4.00
Grading techniques:	12%	50%	25%	12%	0%	0%	3.75

For median calculation: 5 = Excellent 4 = Very Good 3 = Good 2 = Fair 1 = Poor 0 = Very Poor

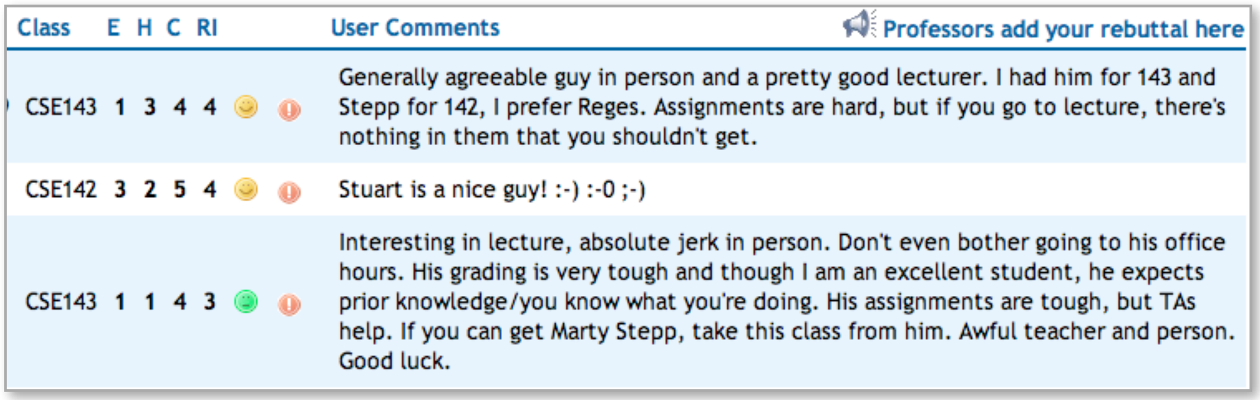
Figure 15: Course Evaluation Sample

The CEC is very well structured and could be easily scraped using an html parser and a few regular expressions. Each course evaluation is linked with a course instance found by the Time Schedule Crawler.

There are instances where the course evaluations report a different instructor teaching the course than the time schedule does. In these cases (522 total), we overwrote the time schedule crawler data because it is older data, and because a course evaluation is more likely to contain the name of the person who actually ended up teaching the course.

RateMyProfessors.com Crawler

Our RateMyProfessors.com (RMP) screen scraper crawls RMP's University of Washington homepage for instructor reviews. RMP's HTML formatting is precisely structured in such a way that we were about to execute XPath queries easily on the raw HTML to gather data.



Class	E	H	C	RI			User Comments
CSE143	1	3	4	4	●	●	Generally agreeable guy in person and a pretty good lecturer. I had him for 143 and Stepp for 142, I prefer Reges. Assignments are hard, but if you go to lecture, there's nothing in them that you shouldn't get.
CSE142	3	2	5	4	●	●	Stuart is a nice guy! :-) :-0 ;-)
CSE143	1	1	4	3	●	●	Interesting in lecture, absolute jerk in person. Don't even bother going to his office hours. His grading is very tough and though I am an excellent student, he expects prior knowledge/you know what you're doing. His assignments are tough, but TAs help. If you can get Marty Stepp, take this class from him. Awful teacher and person. Good luck.

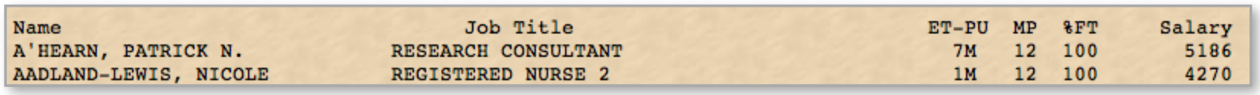
Professors add your rebuttal here

Figure 16: RMP sample instructor review page

RMP allows users to write reviews of instructors they took a course from. Users specify a rating between 0 and 5 for the instructor's easiness, helpfulness, clarity, and rater's interest. We screen-scraped RMP for every review of every instructor we had in our database.

Salaries Crawler

Because UW instructors are employees of the state, their salaries are public data. lbloom.net gives salary information for all state employees, and there is a neat compiled list of UW salary data for odd years. Only years 2009 and 2007 give good enough data to allow us to accurately compare salaries between instructors.



```
Name                            Job Title                       ET-PU  MP  %FT    Salary
A'HEARN, PATRICK N.             RESEARCH CONSULTANT                7M  12  100      5186
AADLAND-LEWIS, NICOLE           REGISTERED NURSE 2                 1M  12  100      4270
```

Figure 17: lbloom.net salary sample

This information includes employee type (ET, 6 for faculty), pay unit (PU, M for monthly), and full time percent. The crawler matches the name given with an instructor in the database, storing a new salary if an instructor is found.

Publications Crawler

The publications crawler relies on CiteSeer, a service hosted by Penn State University, to find publications matching a given instructor name. It has a built in search capability which the publications crawler makes use of in order to target specific names, instead of crawling through the entire database. The general algorithm used is as follows:

```
For each instructor:
    Search for instructor.name
    For each result on first page:
        Follow link, extract title and author list from page
        For each author in list:
            If author name matches an instructor in our database:
                Add author name, publication title to database
```

For efficiency, the crawler remembers which pages it has viewed, and will not revisit a page. It also remembers which instructors it has searched for, in order to better restart in case of errors or other shutdowns while crawling. Because of the large amount of pages crawled, the entire crawl takes a very long time to complete. It is also vital to throttle the network access, as unrestricted crawling would put a very heavy toll on the remote server.

Homepage & Images Crawler

The homepage and image crawler uses the public Google search API to scan potential instructor home pages in the washington.edu domain. Searches are conducted by instructor name, and all results are examined and tested via the homepage classifier. This classifier is the only component of our application to do any form of classification or similar AI. It is trained on a set of 64 examples, 41 negative and 23 positive. These examples were the first results from a Google search on an instructor's name, and were labeled by hand. The current features detected by the classifier are just words; HTML formatting is stripped, and the URL is not taken into account in the base classifier. However, there is a prior step, in which certain URL patterns are rejected outright. An example would be the pattern '/crscat', which contains course catalog information.

Once a page is determined to be a homepage, we scan the page for images, attempting to find one that is likely to be an image of the instructor in question. A potential image is one having a width to height ratio within a certain range (0.5 - 2) and a certain minimum size (100 x 100px). We store all images that fit these criteria. In the future, we would like to include some functionality allowing users to validate images, as the image processing needed to automatically discover images of humans is currently beyond the scope of this website.

Awards Crawler

The awards crawler is probably the simplest crawler used to gather data. It scrapes a single page:
http://www.washington.edu/uaa/teachingacademy/awards-past.html
and reads the instructors listed under each award category. Because the list of awards goes back much further than our other sources of data, many awards are not matched to current instructors. These awards are discarded, under the assumption that the professor is no longer actively teaching classes at UW.

Compiling Data: MetricMaker

Once we've crawled all the data, we pre-compute many metrics on the data for rapid lookup during runtime of our program. For example, we pre-compute the averages of all of the RMP review entities for every course, department, and instructor. With these averages and the averages of every other metric we have, we are able to run our comparison & leaderboards engine very quickly. If these values were not pre-computed, the page-load time would surely take a long time.

Database

The heart of our system is our relational database which stores instructors, courses, departments, reviews, course evaluations, salaries, and more.

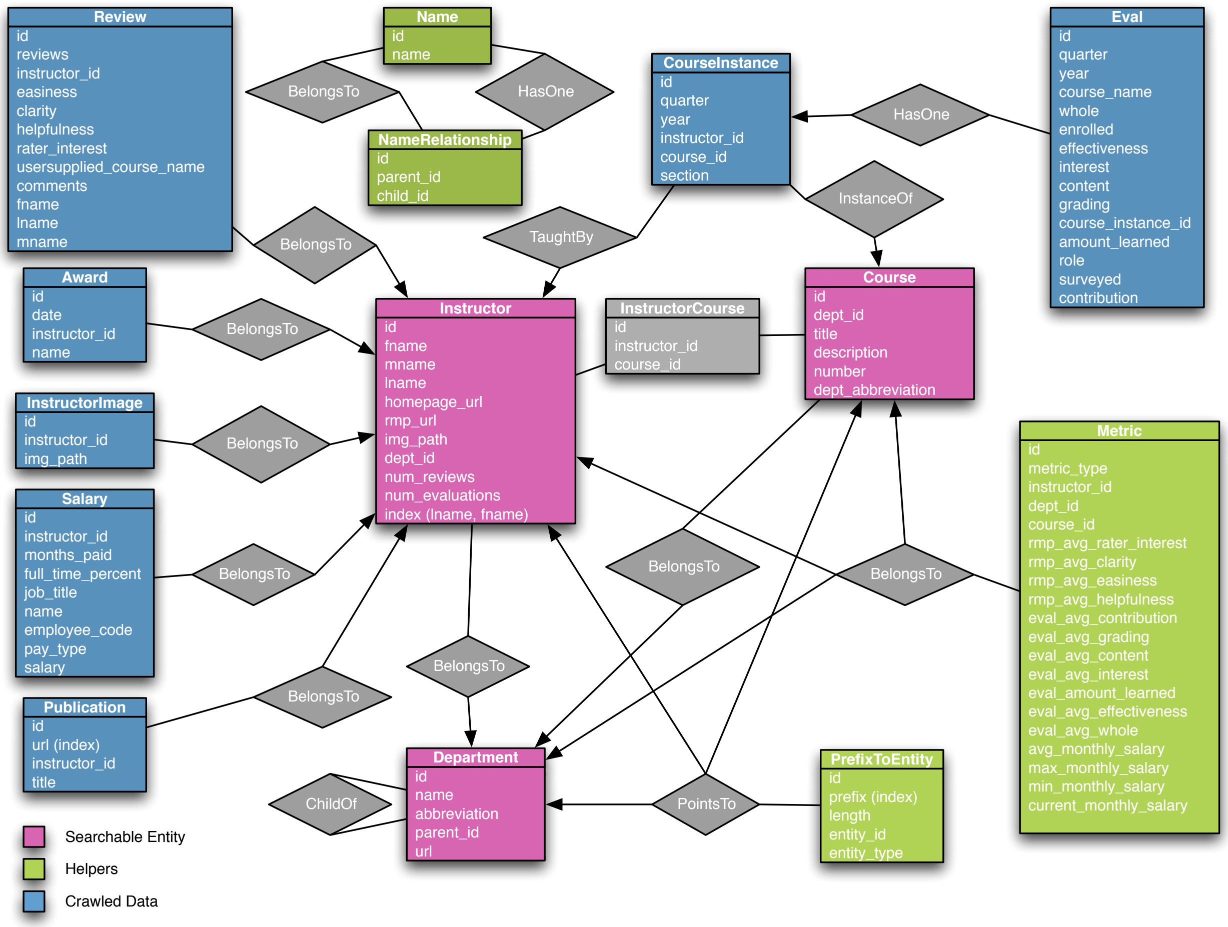


Figure 18: Relational Database Schema

Our relational database is an intricate part to the execution of our program, and every page of our application uses the database to present information in some way.

Search

We decided that the benefit of some sort of full text search would not be very useful to our users. Instead we should provide search that quickly finds results based identifiers such as names, and relationships between entities. Our search is able to accomplish this by using complicated SQL statements that weight results on the fly through the very fast SQL database query manager. While the semantics of how we accomplish this in SQL are uninteresting, what is interesting is what attributes we search on and how we assign weights.

To find relevant courses the highest weighted situation is where a course has a number and department abbreviation that show up in the search query. The next highest weighted situation is if a course has ever been taught by an instructor with a first name or last name that can be found in the query terms; this situation is weighted five times less than the first, so it would take five of these types of situation to match a course to matches the first situation. The last two situations are that a course has matching department abbreviation or number, which are given much lower weighted values. Searching for instructors and departments is almost identical except for instructors matching first name and last name is the highest weighted situation and for department matching name is the highest weighted situation. Now that the weights are clear it is important to understand that these situations are cumulative. An instructor can have multiple courses that they have taught that have matches in the query terms. This is handled in our SQL queries because we sum all the matches we find and Group By id so all the matches weights for each instructor, course, or department are summed together to create a final score which is used to determine the order of our results. These scores are independent amongst the different types of entities in our system because our results are separated by type.

Autocomplete

Originally we developed our autocomplete by taking advantage of SQL's Like operator. However, speed became an issue as the number of rows in our database grew larger. To make this autocomplete fast again we decided to do some pre processing. We initially thought that building a trie (a tree where the nodes contain letters, and walking the tree gives all possible words for some dictionary), but we decided to also try an implementation that could still take advantage of the quick disk access provided by a database. Our database implementation uses a table with columns (prefix, entity_id, entity_type), where prefix is a string that represents an autocomplete query, entity_id is the identifier of a course, instructor, or department, and entity_type identifies what type of entity to refer to. We then fill this table with all possible prefixes of all the possible things we would expect our autocomplete to handle, like instructor names, department names, etc. Then when the user types into the search bar we make a JavaScript request and ask for all entities in our prefix table who match the query. This strategy eliminates the need for the Like operator, but allows us to use the benefits of a database, and was our fastest solution to the problem.

The autocomplete is still a little slow on Instroogle.com. There are a few external factors to this, including a relatively slow host (DreamHost's cheapest account), and a slow database (SQLite3). Ideally, we would create a dedicated service for autocomplete that holds a trie in memory for an extremely fast autocomplete a la Google Suggest.

Evaluation and Experiments

We evaluated our site in two general ways; first through quantitative measurements of several of our modules used to discover and classify content, and second through user feedback in an informal user study.

Quantitative Analysis

Most of our crawlers were extracting information from well structured sources. Only the homepage crawler used some form of classification algorithm, the others just used document structure and layout, discovered manually. For these crawlers, we list several statistics on their data extracted, usually relating to the amount of data covered by that crawler.

Time Schedule

The time schedule data is, in many ways, the central point for all data in the application. It contains all courses and provides most of the instructor names. As previously mentioned, some instructors had duplicated names, such as 'Reges, S' vs 'Reges, Stuart Thomas'. Some of these duplications had a one-to-one mapping between short form names and long form, whereas others had multiple long names matching one short name. One-to-one mappings can be reduced to just one instructor with a fairly high degree of confidence, but many-to-one mappings are unsolvable with just the given data. We could make an intelligent guess by comparing classes taught, but this method cannot be foolproof.

Table 1: Time Schedule Summary

	Occurrences	Percentage of relevant data
Courses	69432	100%
Solvable Duplicate Names	2281	26.7%
Unsolvable Duplicate Names	395	4.62%

Course Evaluations

On rare occasions, the course evaluation data would disagree with the time schedule data. In these cases, we went with the evaluation data, as it is more recent. More common was that evaluation data was not present for a course. This happens because course evaluation is only available for one year, whereas the schedule goes back many years.

Table 2: Course Evaluation Summary

	Occurrences	Percentage of relevant data
Disagrees with time schedule	522	0.7%
Courses without evaluations	62737	90.4%

Awards

Because the awards page goes very far back, there are many instructors listed that are not present in other sources.

Table 3: Awards Summary

	Number of awards
Awards matched to instructors	103
Awards not matched (instuctor not present)	144

Names Crawler

This is the one crawler which does not interact with other areas of the database.

Table 4: Names Summary

	Number of names
Total names discovered	4144
Distinct names	2818

Publications Crawler

We did not find publications for many instructors, but the distribution of publications found has a fairly long tail, with some instructors having as many as 30 publications found. See figure 20 for a visual representation of the distribution curve for publications. This figure does not contain the frequency of instructors with zero publications found, as that would have made the remainder of the graph very small. The number of instructors without publications is 7124, or 83.3% of instructors.

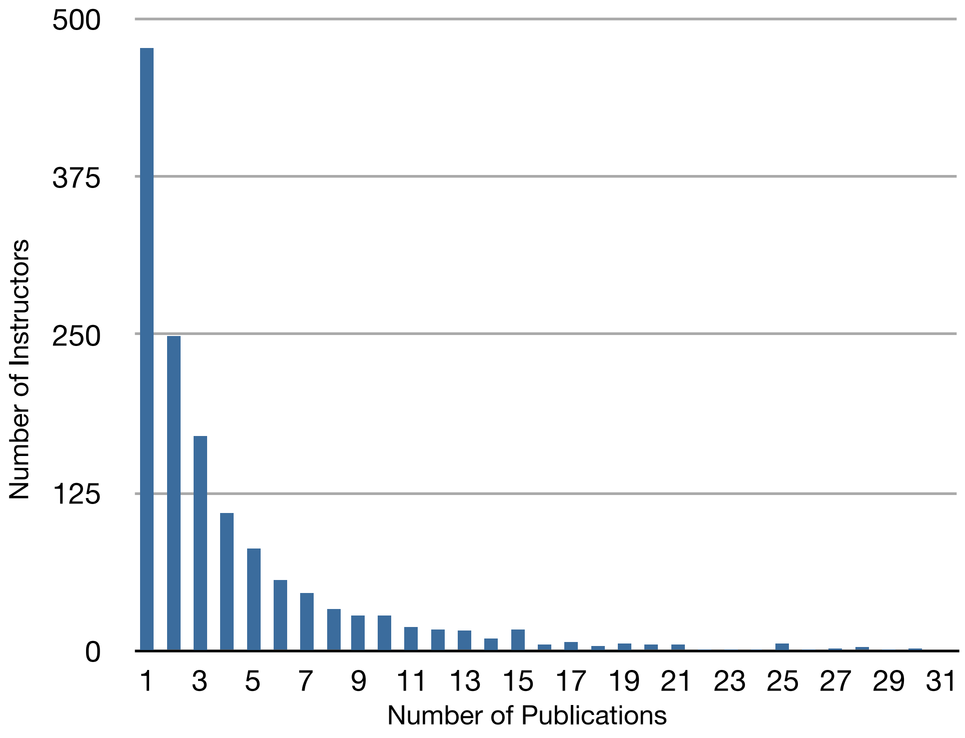


Figure 20: Distribution of publications

Homepage Crawler

Unfortunately, our homepage crawler tended to produce a large number of false positives. These false positives turned up as random noise, and increasing the classification threshold did not improve results. To measure the precision and recall, we took a random sample of 50 instructors, and manually checked the homepage listed, if present, or searched for a homepage if it was not listed. The precision and recall calculated from this testing is listed in table 5 below.

Table 5: Homepage classifier statistics

	Percentage
Precision	42.6%
Recall	87.0%

RateMyProfessors Crawler

Again, the main problem here was data duplication. Because the instructor names are from an external source, we expected a larger amount of variation than from UW sources. As in the publications crawler, most instructors did not have reviews. The distribution curve is also similar, slowing declining from one, but with a long tail. See figure 21 for a graph of the distribution curve.

Table 6: RateMyProfessors summary

	Occurrences	Percentage of relevant data
Instructors with reviews	1046	12.2%
Entries matching existing instructors	8808	65.3%

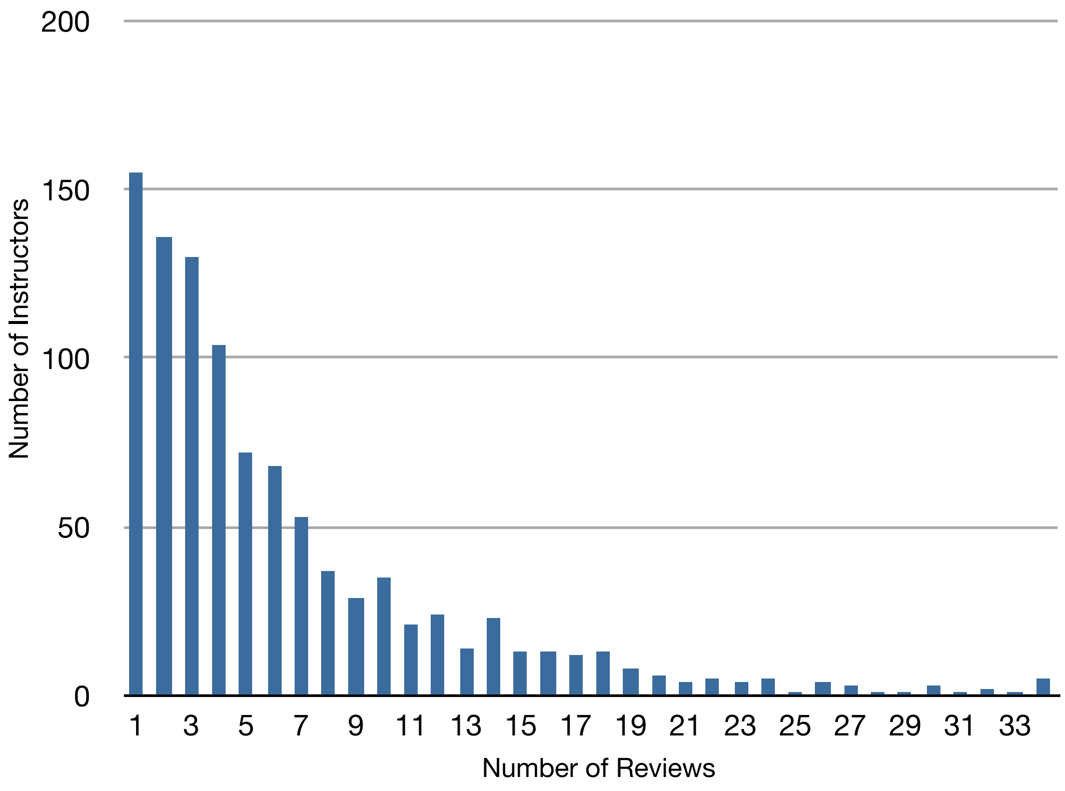


Figure 21: Distribution of reviews

Salaries

Salary data is very structured, but again, matching names was an occasional problem. In addition, salary data is not available for some people classified as instructors, such as many TAs.

Table 7: Salaries summary

	Occurrences	Percentage of relevant data
Instructors without salaries	1046	12.2%

Informal User Evaluations

We conducted a user evaluation with four questions in mind:

- Can people quickly find information about instructors and is the formatting of the information clear?
- Can people understand and use our tools for comparison between entities on our site?
- Once users have found information about instructors, is the information useful?
- What information, or changes to the site could be made to make the site more useful?

Our study was done with a total of seven participants. Five were male and two were female, all were undergraduate students at the University of Washington. Five of the participants used tools like RateMyProfessor.com to help make decisions about what instructor would be best to take a course from, one thought such reviews were unhelpful, sighting metrics like appearance as immature, and one had never heard of any such tools. None of our participants knew that course evaluations and salary information was available online. Most of our participants also mentioned friends and hearsay as important factors in deciding whether to take a course with an instructor.

We performed our testing in one of the project rooms in the Paul Allen Center that provided an isolated, distraction free environment. We did all seven tests separately, and each test lasted roughly 45 minutes. First we had the participants sign our consent form, informed them that we were testing the interface and not them, and described briefly that Instroogle.com was a site that provided information and comparison tools for instructors at the University of Washington. Our tasks were chosen so as to help us answer our four questions stated previously, as well as to allow for use of all of our application's features. For our first task we asked the participants to find information about a specific instructor. We chose a different instructor each time, to eliminate any bias that could be created by asking for only one. Next we asked the participants to choose which instructor they would prefer to take course x from, where x was a different course each time. Lastly we asked them to find an exceptional instructor within their department and find an interesting course that this instructor taught.

Results

We had each user run through the three tasks, but used a separate ordering each time, to avoid any biases a certain ordering might produce. We gave minimal help to the users, only assisting when it was clear they had misunderstood the task. After each user was done, we asked them for any general comments or suggestions on the interface, and then thanked them for their time.

We were generally pleased with the results of the tests as were the participants with our site. The general feeling among the participants was that the site was much quicker and more satisfying than searching the UW time schedule and tools like RateMyProfessor.com. Most were also surprised and excited to see salaries and evaluation data which they assumed was private data that the University did not release to students.

Finding and Understanding Information

All of our participants were able to search and find information very quickly. However, there were a couple of small usability issues that slowed them down. The first issue was discovered when a participant tried to search for the Mechanical Engineering department by typing ME in the search bar. The autocomplete returned no results and neither did the search results page. As it turns out the actual abbreviation for the Mechanical Engineering department is “M E” (separated by a space) and the autocomplete and search were not smart enough to eliminate the space in between the words. This problem could be solved by making our search and autocomplete smarter, possibly just by adding this case.

Four participants got stuck on a department profile page where they were looking for courses offered by the department. Finding the list of courses was no problem but each course is listed with only its department abbreviation and number, which forced participants to click back and forth looking up the titles of each course on the course profile page. This issue could be easily solved by just adding the title to the link for each course on the department profile page.

Along the same lines as the last issue two participants complained because there was no link between a course and its department. The interface forced them to do a new search for the department rather than click a link. This too is an easy fix but an important problem with the flow and navigation of the site.

On top of minor issues with finding information, participants also discovered some issues with the format of some information that confused them. Five of our participants at one point asked us if the scores we showed them were out of five, because there was no indication of the maximum score. Another point of confusion was what the scores and salaries on the course page actually meant. Interestingly this was not an issue on the department page, but people did not immediately understand that we could also aggregate salaries and scores from all teachers that had taught a course. Both of these issues seem minor and are easily fixed by adding a small amount of information in the right places.

Overall participants were able to quickly find information.

Using the Comparison Tools

The user study was arguably most helpful in pointing out short comings with our comparison tools. The first issue participants found frustrating was not being able to compare all the instructors that taught a course without adding each one individually. Upon understanding this problem we also realized that being able to add all the courses offered by a department is a similar issue that should also be addressed, although none of our participants pointed this out. Fixing this is more complicated than most of the other issues because it isn’t clear the best action to take. One idea is to have a button on the course profile page that reads “Compare All Instructors for Course X”. Another solution would be to allow the course to expand once it is on the compare page with all of the instructors. Clearly more user studies would be needed to make this process work smoothly.

Another issue that would take some time to solve is that there is no undo or trash can on the compare page. While only one user in our study complained about the lack of this feature it seems useful and important if people will be using the comparison page for more than a minute or two.

The last issue we found with our comparison tools was that users did not understand the feedback we provided after they pressed the “add to compare list” button. Three participants thought that nothing happened after clicking the button and that our site was broken. This problem is easier to solve, we plan on making each button turn grey and disabled if the entity it adds to the compare list is already in the compare list. This would give more visual feedback after pressing the button.

While there were issues with our comparison tools they were not related to the concepts behind using such a system. All our participants were able to quickly understand how to use the tool and understand more deeply the relationships between the entities they compared.

Usefulness of the Information

To understand how well our interface achieved this goal we couldn't simply look at flaws in our interface but rather how effectively and convincingly were people able to answer the questions we gave them. While this is difficult to impossible to gauge aside from listening to the opinions of our participants, we feel their explanations were very helpful. The general feeling we got from or participants was that the best way to know how much they will like a teacher aside from having the teacher in a previous course is hearing about the professor from friends and peers. This sounds a lot like the information we provide from RateMyProffesor.com, but our participants brought up that we only provided numbers, and that descriptions would be helpful as well. Along these same lines we came to understand that the participants seemed to want a more social atmosphere to discuss instructors that allows users to generate content.

Conclusions

While we are very pleased with the artifact we have created, we realize that Instroogle has much more potential to be explored. We can already imagine another rich feature set that would fit perfectly on top of Instroogle.

Goals

These were the original goals from our project proposal:

"Steam will build a website that lets students search and compare interesting information about professors. This includes graphs of course evaluation histories, salary comparisons based on departments, best reviewed professor for a given course, number of publications relative to salary, etc."

Achievements

We overwhelmingly feel like we accomplished our initial goals and even exceeded some of them. Based on the feedback we've received from our peers and friends, our artifact is impressive, useful, and fun. Instroogle is definitely on the right track to becoming a fully-fledged instructor search engine, and we're really excited to see what the future brings us.

Future Work

Right now Instroogle does not support the ability to have department-specific leaderboards of courses and instructors or course-specific leaderboards of instructors. We have the technology to implement said features, but we chose to delay the implementation of these features until after we finished our larger features. Once the quarter was coming to an end, we could not afford to budget time for these features. We plan to implement them in the near future.

Instroogle does not graphically display any data or trends in its current build. We originally wanted to plot data about instructors over time and use these visual aids to assist users in their interpretation of the data. We want to add this capability in the time to come.

One area we originally wanted to explore but never budgeted time for was our own engine for user feedback of courses and instructors. At the moment we rely on data from UW's course evaluation catalog and RateMyProfessors.com to rank and compare instructors, courses, and departments. MetricMaker currently ranks and compares courses and departments based on aggregated data of individual instructors who have taught in said courses and departments. While this approach is acceptable at the moment, we believe students should be able to review courses more generally, such as "favorite topic discussed in the course" or perhaps a discussion of whether or not the course is typically graded on a curve. This type of feedback system does not exist anywhere else.

If we had our own feedback interface, we could structure it exactly the way we wanted it and explore many more metrics not previously covered, such as "average number of hours spent per week working for this class." We would also have a "comments" field that would allow users to write detailed comments directly on our site instead of through RMP.

Instroogle also has the potential to fit nicely into a larger suite of applications. For example, with our structured data set of instructors, reviews, ratings, evaluations, and our ability to load UW's time schedule data, Instroogle would fit perfectly into a schedule planner. Students could use Instroogle's technology to plan their course schedule each year, always knowing the reputation of the instructors of the courses they are considering taking.

Additionally, one day Instroogle could very well be abstracted to work with any university. There is so much rich data on the web about instructors, courses, and reviews. Our relational database is relatively school independent (besides course evaluations which are UW specific). Data from about any university could be parsed and our service would operate successfully. We can imagine Instroogle being the internet's leading source for college students looking to learn more about the courses they are taking and the instructors who teach them.

If We Could Go Back...

If we could start the crawlers from scratch, we would first develop a way to download the crawled content that was no longer being updated. For example, the UW time schedule data from Spring 2006 to Spring 2009 is unlikely to be updated anymore. Had we downloaded the content of these pages, we would have seen a significant increase in speed of the time schedule crawler which currently requests each page over the internet. Reading these files locally would be significantly faster and drastically reduce the network activity of our application.

We also should have developed a system to keep everyone's data in sync throughout the development phase. Most tables in our database have foreign-key restraints into the instructors table. Thus every time we ran the time schedule crawler, we had to run all other crawlers again to use the new data we gathered. Unfortunately we had to run the time schedule crawler dozens of times throughout the quarter because new features were constantly asked to be extracted from the data.

A logger of the crawler's activity would have allowed us to reuse some of the crawled data from RateMyProfessors.com and other sources that critically depended on the time schedule crawler's updated data.

Instroogle in many ways is quite fragile. Because we rely on screen-scraping to parse data from UW's time schedule, RateMyProfessors reviews, and more, our crawlers would need constant tuning if these services ever changed their HTML code (which eventually will happen). In the future we would like to begin relationships with the services we are crawling to gain access to some sort of API or XML feed of their data. If that were to happen, Instroogle would be able to adapt to site formatting changes and ultimately have more secure, confirmed data.

Overall, this was a great experience and we all learned a lot from it. We are extremely proud of the artifact we created and have already used it in our personal course planning. Instroogle development is just beginning: we have a full bug and feature list that needs to be worked on.

Appendix

Roles

Patrick

Patrick wrote the time schedule crawler and the RMP reviews crawler for Instroogle. He also wrote the entire Leaderboards engine and contributed to the development of the three types of profile pages Instroogle has.

Matt

Matt wrote the salaries crawler and the course evaluations crawler. He also was the webmaster and set up our SVN code repository, our web server and is ultimately responsible for Instroogle.com. Matt's biggest and most impressive contribution is the styling you see when visiting Instroogle.com. Matt was our lead designer and is the reason our site looks so impressive.

Daniel

Danny's main contribution to the project was the search engine. Written entirely from scratch, Danny implemented both autocomplete search results and full search results when autocomplete did not suffice. Danny also wrote the names crawler and the homepage crawler.

Tim

Tim wrote the homepage classifier used by Danny's homepage crawler, and also wrote the publications crawler of Instroogle. Tim designed and implemented the comparison page which features sorting on different metrics. Tim also owned the session logic which allowed users to have a "comparison cart" throughout their visit to Instroogle.com.

Group

The entire group worked together to design the artifact from the beginning to the end. From the initial database schema to the per-pixel design of our homepage, the group worked together to produce the best product possible.

The group worked quite well together. It was invaluable that each member of the group had some experience working in Ruby on Rails before beginning this project. Each student had also taken Databases which was a big help too.

The biggest problem our group had throughout the quarter was finding times to meet to work. Every one of us was extremely busy with our heavy course load and interviews for after graduation. There was a three week stretch towards the end of the quarter where we each had 4 or 5 interviews per week!

Consent Form

The Instroogle application is being produced as part of the coursework for the University of Washington Computer Science course "CSE 454: Advanced Internet and Web Services". Participants in experimental evaluation of the application provide data that is used to evaluate and modify the interface of Instroogle. Data will be collected by interview, observation, and questionnaire.

Participation in this experiment is voluntary. Participants may withdraw themselves and their data at any time without fear of consequences. Concerns about the experiment may be discussed with the researchers (Patrick Healy, Matthew York, Daniel Swisher, or Tim Crossley) or with Professor Dan Weld, the instructor of CSE 454:

Daniel Weld

Computer Science & Engineering

University of Washington

dan at danweld dot com

Participant anonymity will be provided by the separate storage of names from data. Data will only be identified by participant number. No identifying information about the participants will be available to anyone except the researchers and their supervisors.

I hereby acknowledge that I have been given an opportunity to ask questions about the nature of the experiment and my participation in it. I give my consent to have data collected on my usage and opinions in relation to the Instroogle experiment. I understand I may withdraw my permission at any time.

Name ____

Date ____

Signature____

Witness name ______________________________________

Witness signature___________________________________

RateMyProfessors.com Rating Categories

Easiness

- Some students may factor in the easiness or difficulty of the professor or course material when selecting a class to take. Is this class an easy A? How much work do you need to do in order to get a good grade?

Clarity

- A professor's organization and time management skills can make a great difference on what you get out of the class. How well does the professor teach the course material? Were you able to understand the class topics based on the professor's teaching methods and style?

Helpfulness

- Helpfulness is defined as a professor's helpfulness and approachability. Is this professor approachable, nice and easy to communicate with? How accessible is the professor and is he/she available during office hours or after class for additional help?

Rater Interest

- There is always that one class everyone recommends taking before graduating. As a student, how interested were you in the class, BEFORE taking it? Or how interested were you in taking this course from this specific professor.

Outside Code

Dexagogo - a javascript sortable table using the prototype javascript library

Prototype - a javascript library that comes stocked with Ruby on Rails.

Scriptaculous - a javascript animation library built on top of Prototype. It also comes stock with Ruby on Rails

Hpricot 0.6.164 – HTML parser. We needed this specific version to run our crawler due to some poorly formatted html. This version works some magic.

Sanitize – HTML sanitizer. Removes tags from the html to get just the text.

will_paginate - A pagination plugin for Ruby on Rails.

ruby-imagespec - Reads metadata from image files

Sites Crawled

BehindTheName Crawler: http://www.behindthename.com

Time Schedule Crawler: http://www.washington.edu/students/timeschd/

Course Evaluation Crawler: http://www.washington.edu/cec/

RateMyProfessors Crawler: http://www.ratemyprofessors.com/SelectTeacher.jsp?sid=1530

Salaries Crawler: http://lbloom.net

Publications Crawler: http://citeseer.ist.psu.edu/

Homepage & Images Crawler: Google search API on washington.edu domain

Awards Crawler: http://www.washington.edu/uaa/teachingacademy/awards-past.html

Instructions

Browser Requirements: Instroogle works best in Safari, Firefox, or Chrome

If you would like to run it from scratch, you will need all of the projects listed in Outside Code and you will need to run the crawlers. Instructions can be found by running "`rake -T`" from the root application directory.