# GradRanking: Online Personalized University Recommendation System

Georgy Megrelishvili
Brown University
Department of Computer Science
Providence, RI
georgy_megrelishvili@brown.edu

## ABSTRACT

Ranking systems are applied in many ways nowadays and have become a commodity to use when it comes to making decisions. Education is one of the most important parts of human life. Choosing a right place is crucial from many perspectives: quality of life, knowledge gained, post-graduation prospects. Making a right choice what institution to attend is based on many factors and is hard to make in a limited amount of time. In our masters project we address this issue and present an online personalized university recommendation system GradRanking.com. We collected information on 1250 universities in the U.S. and over 118,000 student profiles who applied to those institutions. We also use social networks like LinkedIn and GlassDoor as a source of information to understand where students tend to go after graduation: academia or professional career, and depending on the chosen path, help decide which universities provide better opportunities.

## Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval — *Information Search and Retrieval*

## General Terms

Theory, algorithms, web applications

## Keywords

Information retrieval, machine learning, data mining, regression, ranking

## 1. INTRODUCTION

The hypothesis of this project is that a personalized user oriented system can be built on a prior knowledge about the universities and the learning data from student profiles. Matched against each other to build a model for each institution to estimate the chances of being accepted and rejected, and how useful it is for a particular person to attend it. The project is subdivided in several integral parts: data collection, data parsing and ranking based on several widely used machine learning methods.

First we go in description of data collection. Several main data sources were used for this project: we started with collecting the publicly available data provided by universities and aggregated by the yearly US News Report. The ranking provided by company was totally neglected in our ranking schemes. Secondly we collected a large block of data from two major student run sites Edulix.com and TheGradCafe.com. Both provide similar functionality and allow students to report their progress with standardized tests and university applications. The third data source was a professional social network LinkedIn. It provides information on employment and professional growth of graduates. The fourth source of data is a crowdsourced employment information site GlassDoor.com which contains data on salaries and overall happiness of employees in a large number of companies world wide, a majority of which are located in the U.S.

Having all that information in a structured way we are able to build a system to provide a personalized rating of universities based on academic vs. professional preference of a candidate. As an input (query) from a user we collect all possible information asked in an optional input based manner: does she see herself in academia, what are the scores on required tests, location preference, tuition costs depending on location (in-state, out-of-state), scholarship, fellowship probabilities, future salary approximation.

Not all of 1250 U.S. schools are offered in the final ranking, but only those within the user's field of study and that fit in terms of provided query from both bottom and top border of acceptance expectations. After getting the rating user can request detailed information on every school offered in a list. Only some of the data associated with every school is used in ranking algorithms. We keep exuberant data to provide it to the user on demand in order for her to make most informative decisions on finalizing choices which schools to apply to.

## 2. RELATED WORK

Our problem is best solved using regression methods. But since we are looking not for one closest solution but also for a ranking scheme, different approaches were offered in information retrieval and machine learning literature.

The mainstream approach to learning to rank is often called the pairwise approach. There is a variety of other methods [3, 5, 2]. The pairwise method is more widely used and can be described as classification of object pairs to be correctly or incorrectly ranked. Early research describes use of SVM to produce a classification model, like Ranking SVM. Other approaches included Boosting, RankNet. The later used cross entropy as loss function and gradient descent to train a neural network. At the same time learning to rank methods are mostly used in information retrieval problems such as document retrieval using Ranking SVM, document

search on a large web size scale.

Other influential work includes attempts to switch from pairwise to listwise approach in ranking [1, 6] where lists of objects are used as instances in learning. In general that employs a probabilistic approach to ranking. Two probability models are involved: one is permutation probability, another is top $k$ probability, which together define a loss function for learning. Gradient descent and neural network are then used as algorithm and model for ranking.

Work of D. Sculley from Google Research [4] directly touches upon effective combination of regression and ranking, which is in a modified way used in this work. Author establishes two types of metrics and develops a model that is required to achieve best possible results on both of them. First comes from regression part of the problem, second from the ranking. First metrics are such as Mean Square Error; the model is rewarded if $y'$ is predicted well for a given example. The other metrics are such as Area under the ROC curve; the model is rewarded if $y'_1 > y'_2$ is a correctly predicted ranking for a pair of given examples.

# 3. METHODOLOGY

Data collection is an important step in success of an information retrieval or data mining project. In our research we collected different types of data for different purposes. The first one is universities data used in regression to build ranking for the user query. It is obtained from the open sources, partially from the US News Report site, which provides its own universities rating. No rating from any external site was ever used as a reference while own ranking algorithm was developed. The major difference between our site and US News Report as long as many other similar resources is that they provide a static recommendation of best places to study in with no respect to user requirements or abilities. They have a concept of ideal student and ideal study place that satisfies all the needs of a prospective colleger. On the contrary we believe that this scheme does not work for majority of students and needs to be evolved.

In previous steps we collected static data to be matched against while ranking is built, next we collect the learning data. It never shows up in user observed results, but is used to generate a statistical model to understand whether a candidate is likely to be accepted to a university considering his scores in standardized tests, degrees and honors. This data is not used in ranking, but rather in classification problem if particular university should be offered to candidate in the final result table.

Next step brings us to collection of social network data. This block of information leverages the vast data available on the internet in structured form. Concretely we aggregate information on places where graduates of the U.S. universities tend to go immediately after graduation or as a first job afterwards. Having that large list of employers we want to understand which are more privileged and so we collect data regarding salaries in those companies for the positions typically occupied by entry level employees as fresh graduates, and general level of satisfaction of working for a particular employer. This information is used on the last stage when the final ranking is prepared.

The last source of information in our system to date is a large study performed by Business Week that shows percent of students who manage to successfully graduate, get a grant while at school and what is the return of investment estimation from education in each institution. In other words how fast if at all possible student can get back the money invested in the education. All these pieces of data let us proceed to the next step which is ranking.

The ranking part part contains three major steps: building rating of employers from social network data, building a classification model for every university based on learning data, produce rating based on user input and result received from first two stages. Steps one and two are "one time compute" and do not need to be repeated every time a user sends a query.

Developed system uses supervised classification methods and unsupervised learning on stage of data preparation. Due to a noisiness of collected data immediate usage of it would be impossible, final stage uses regression for final ranking output and presents results to a user. Presented results mean that it is likely with high probability to be accepted to all of the observed schools in the list. Schools are not rated which are better or worse, but instead are ranked which are more preferable for the user considering her needs and abilities.

# 4. DATA COLLECTION

In order to collect data we used Python scripting language, Linux scripts, regular expressions and cron jobs to distribute and schedule tasks. Collecting data on large scale is not a trivial task. Sites Edulix.com and TheGradCafe.com on the moment of crawling contained more than 150,000 user profiles. First we developed a crawler and parser to collect user data from the site. The data contains several entries: scores on standardized tests (GRE, GMAT, TOEFL), schools that candidate applied to, schools that he was accepted to and rejected from and what school has he finally chosen to attend. It also contains a text field that contains a list of honors, special degrees and achievements, that user believes are important for schools to make decision on his case. Collected data turned out to be very noisy, some of the form's fields were missing or filled out incorrectly. After leaving only fully filled profiles we remained with about 43,000 user forms with sufficient data suitable for learning. We store data as hashmaps in Python allowing us to get in $O(1)$ time all learning data for any university that we are building model for. We used several widely used Python libraries and extensions in the course of crawling such as UrlLib2 for sending HTTP requests; HTMLParser for getting the payload in HTML text retrieved; htmlentitydefs for getting only the needed parts of HTML payload in particular subsections; pickle for storing a binary representation of dictionaries and quick save and load operations to disk in order for the data to be usable across different processes and machines.

Professional social network LinkedIn contains around 150 million professional profiles with education and employment information available to date. We used an Executive payed plan to have access to as much information as possible at a time. This type of subscription allows to get up to 700 results per search query at a time, grants access to search filters like company size, years of experience, interests, Fortune 1000 and others. This type of subscription also provides full name information, but we would like to make a statement

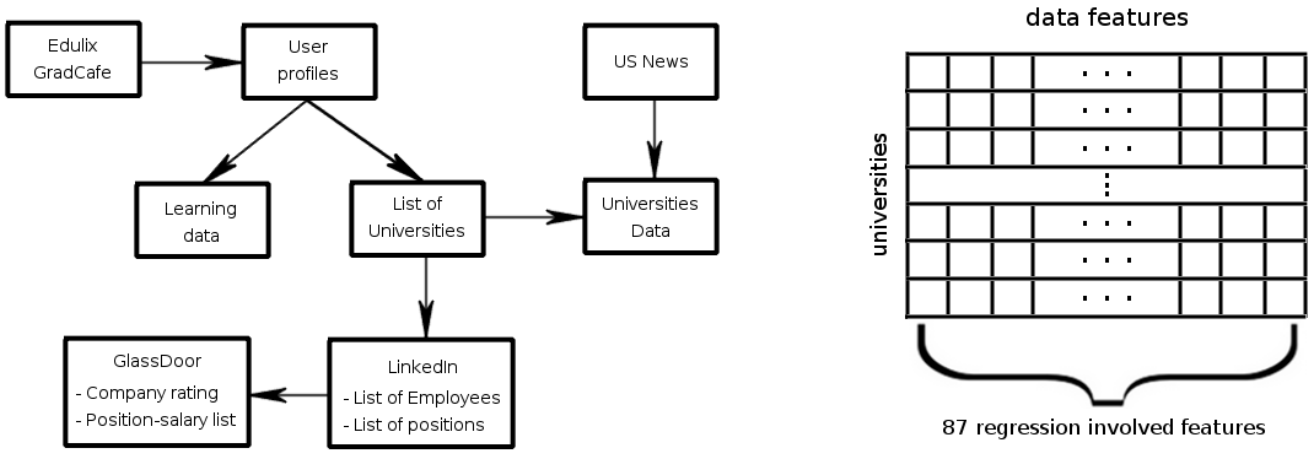


**Figure 1: Data scheme of the GradRanking project. Arrows show the flow of information. The table on the right is the resulting data that is used in ranking**

about privacy of collected information from this and other sites mentioned above and later. Whenever we parsed we immediately removed names and stored information anonymously and mostly in aggregated way. LinkedIn provides two basic ways of getting information not by parsing site's HTML pages, but by using inhouse developed API. First one is REST based second is JavaScript. We used second and during the course of two weeks we were able to collect around 40,000 user profiles having universities from our list as their education place. Amount of data was skewed towards bigger and more popular schools. Some got as few as 40 results in total, while some 1217. On average every school had 354 user profiles, 217 in the median. For every employment entry we collected a tuple information of position within the company.

For every university we extracted a list of employers that were listed on the profiles to be first after graduation from a school we were interested in. We neglected information whether employment happened immediately or few years later, although we understand that in some cases that may be also a signal that we would like to take into consideration. We ended up with a list of 5715 employers, some of them were ranked in Forbes 1000 list which gave an additional boost in understanding a demand for graduates of a particular school, some employers we not discoverable by search engines. This brought us to the need to rank the employers not by use of another rating provided by third party but by ourselves. Thus we came to GlassDoor.

GlassDoor is an insider information company. Users are offered to provide a piece of information about their company anonymously in return of free registration and access to all the site's data on all other companies. Information provided can vary from the number of employees in a company to a salary and position, individual ranking, and such. Data gathered from a source like this can never be considered reliable. In order to make GlassDoor information trustworthy and useful for our cause we removed outliers from the collected data and retained only positions and salaries within $[-2\sigma; 2\sigma]$ distribution as shown on Figure 4. The site does not have an API so we had to parse it by writing Python scripts. There is no limitation on number of requests coming from one profile, but there is a time limit for the frequency of those requests. Robots.txt in the root of the site allows only Google's bot to crawl the site, while all other are disallowed to enter listed sections. For every employer we collected the following information: "positions - salary" chart, aggregated company rating based on employee reviews. We parsed the "position - salary" chart for every company and left only those salary entries that matched positions listed on LinkedIn profiles of school graduates. We also tried another approach to search for "company name position name" iteratively and parse that data, but it turned out that the the recall is higher with just company name and parsing of all the entries that match our positions list. On average we collected 114 entries for 5715 employers. The data is very skewed and while some companies have around 10,000 entries, others have none.

# 5. APPROACH

In order to be able to use all collected features of the data we need to remove the noise in it and primarily that is true for the gaps. Of 87 collected features for 1250 universities 24 did not have full data. Universities less popular among students are not making information about themselves publicly available and it makes it hard to find all the needed data, e.g. endowment size or expenses per faculty member. We had this data features for the majority of school and didn't want to discard them. Figure 2 shows the representation of this situation.

Our approach is to approximate values for missing features having all the rest of fully available data. Average value for the feature calculated with all data in corpus would be not representative since if we look at all features we notice that some schools are far way up in the ranking while the majority is decreasing gradually. If we were to take just an average value we would let the top tier schools contribute to higher ranking of the the bottom once. To avoid this situation we first cluster the universities based on the feature columns that fully available. Then to be minimally biased we take the median value within a cluster for each gap for a particular school.

We use unsupervised learning $K$-means clustering algo-

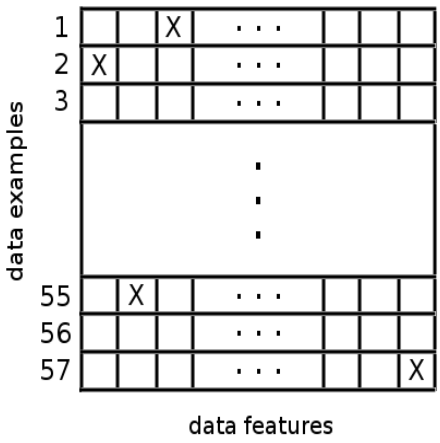


**Figure 2: Gaps in the noisy data as they appear when it is collected. If not removed, data is unusable for regression purposes.**

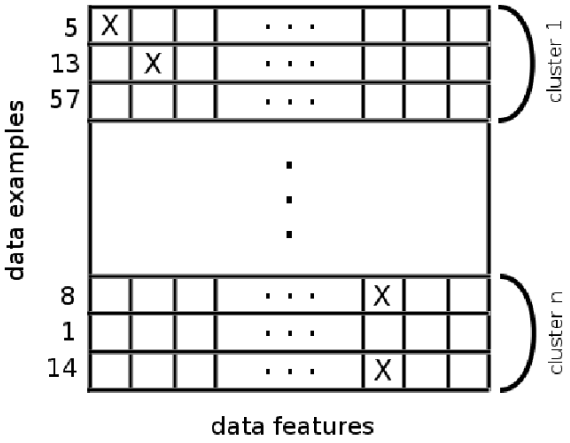


**Figure 3: Data is clustered over the features available for all data point. Missing features are approximated by taking the median within a cluster.**

rithm for partitioning $N$ data points into $K$ disjoint subsets $S_j$ containing $N_j$ data points so as to minimize the sum-of-squares criterion.
Algorithm aims at minimizing an objective function $J$:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \|x_n - \mu_j\|^2 \tag{1}$$

where $x_n$ is a vector representing the $n$ data point and $\mu_j$ is the geometric centroid of the data points in $S_j$.

Before we move on to ranking part we would like to describe the social network integration in the dataset. We should note that we build two separate databases of positions. First describes academia jobs, keywords include: "research", "professor", "postdoctoral" and a few other. Professional positions are considered all the rest that we encounter. Previously we collected information on employers, how happy their employees are judging by reviews and what is the salary for positions that fresh graduates qualify for. Having that information for each of the databases separately we put a prior that those are equally important and contribute proportionally to the rating of a company. We rank companies based on information from each column separately. Than we combine those ratings. Company's final place is a mean of places in two separate ratings as shown in Figure 3. One company can appear in both academia and professional datasets. But since later we use only one of those datasets, depending on user query, it does not add any confusion or rating boost.

Once we have the data to match with our query, we need to scale the features using data normalization. All of the features are on a different scale. From fractions and values less than 1 to billions. We use feature scaling as data normalization technique.

We aims to scale the range in [-1, 1] using the following formula:

$$x' = \frac{x - \bar{x}}{max - min} \tag{2}$$

where $x$ is an original value and $x'$ is the normalized value

Now when all this preliminary work is done (should be repeated from time to time to update ratings), we come to a user query and personalized recommendations. The interface asks user if he is interested in professional or academic career path. Depending on choice made the appropriate dataset is selected. Next we ask if user had classes in his undergraduate education being taught in English. Thus we decide if we also need to take English language exams like TOEFL / IELTS into consideration. To this point we have decided on the features that we are going to use in our recommendation system.

Now we have three steps remaining before user gets her personalized school recommendations:

1. Scale user query.
2. Classify universities based on probability for user to be accepted.
3. Rank those that were classified as positives.

User query is nothing else but a list of features. We set minimal requirement for a query to contain the field of study and GRE / GMAT score (TOEFL if applicable). Although we understand that some schools including Brown University and MIT do not require those, we need some basic input in order to estimate probabilities of being accepted by a particular school (the more input features, the better). Other optional user input features are tuition cost, future salary, ease of employment, location (region).

In order to normalize a query we use the same Formula 2 that was given for the dataset.

Next we classify the universities in two classes. Being likely to accept user or not. We use logistic regression for classification purposes:

Logistic regression works well for binary classification problems where we have labeled training data and examples have $l$ features with values equal to zero or one. We denote concrete example is $\vec{x}$ and the value of the k feature as $x_k$. An additional feature, $x_0 \equiv 1$ is the "bias" feature. The probability of an example to belong to a positive class is

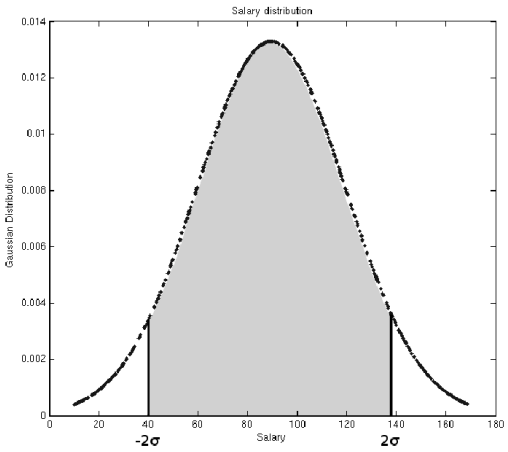


**Figure 4: Salary distribution. Only entries within [-2$\sigma$; 2$\sigma$] were considered. Outliers were descarded from learning data.**

$$p(y = +1|\vec{x}) = g\left(\sum_{k=0}^{l} w_k x_k\right) \tag{3}$$

where $g(z) = \frac{1}{1+e^{-z}}$ is a sigmoid function having values in range [0; 1] which is required for classification purposes.

Logistic regression allows to learn the weight of the features to maximize the likelihood of the data. Having training data $(\vec{x_1}, ..., \vec{x_n})$ and corresponding labels $(y_1, ..., y_n)$ logistic regression maximizes the log likelihood of the data

$$L(\vec{w}) = \sum_{i=1}^{n} \log g(y_i z_i) \tag{4}$$

where $w_k, k \in \{0, ..., l\}$ are weights of features and

$$z_i = \sum_k w_k x_{ik}$$

and since we are having probabilities $1 - g(z) = g(-z)$.

We can learn the weights using two approaches which are used depending on the size of the training set and number of features in data. It is either gradient descent method or solving linear equations. We used the gradient descent for scalability. In order to get the $k^{th}$ weight we calculate

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^{n} y_i x_{ik} g(-y_i z_i) \tag{5}$$

Once this stage is complete we can leave in our final list only universities that are likely to accept the user according to a query.

Next step in our pipeline is the ranking. We intend to rank the schools based on several criteria. Since user is providing us with his preferences for post-graduate expectations and his current academic and financial standing, we can use a regression approach to estimate which universities are more useful and appealing for the user to attend.

There is a not obvious part in this problem, having the normalized data and user input we have dozens of features that have to be also used in regression model but there is no user query criteria to match them against.

We use logistic regression to learn weights of data features. Depending on the career path that user chooses, academia or professional we pick relevant data features and use them in prediction model. Regularization gives an additional boost in precision.

Formula for logistic regression was given in Equation 4. We add regularization term to it to minimize bias from the features. We use a new cost function

$$L = -\sum_{i=1}^{n} \log g(y_i z_i) + \frac{C}{2} \sum_{k=1}^{l} w_k^2 \tag{6}$$

We end up with having a ranking that we show to the user. The chart may contain a large list of entries so we show them by groups of ten with a Next button.

# 6. RESULTS

Here are the results that we had with our dataset on classification problem. We had manually selected user profiles for a test set where we knew the expected data label. The whole dataset is 43,127 user profiles. We split it in three parts: 80% learning set, 10%, cross-validation set and another 10% test set. During the learning and validation parts we rerun the classifier on randomly shuffled data with datasets preserved in the same proportions.

Misclassification error on test set was within 5%. Results of a sample run are presented in Table 1. In this case user query had a tuition payment possibility of $50,000 (which exceeds most non-business school expectations), top 1% GRE and TOEFL exams scores and showed interest in professional career after graduation rather than academic. Considering the data that we have on return of investment per school and the teaching and research quality in listed institutions, without details it can be said that the developed ranking system produces plausible results.

Project that we have been working on does not have competitors to compare results with. Also there is no ground truth that may be unbiased to be used as a reference. The results on a test set and general rationale in sections above is our way to justify the decisions that have made while working on this project.

# 7. CONCLUSION AND FUTURE WORK

Ranking systems are applied in many ways nowadays and have become a commodity to use when it comes to making decisions. Education is one of the most important parts of human life and in this work we started a project that makes it easier to come to a right choice what university to attend. Decisions made by the system are personalized. Recommendations are offered upon considering over 80 creterias.

This project will be hosted on domain GradRanking.com. We collected information on 1250 universities in the U.S. and over 118,000 student profiles who applied to those institutions. We also use social networks like LinkedIn and GlassDoor as a source of information to understand where

**Table 1: Results for a query {possibility to pay high tuition expenses; top procentile GRE, TOEFL exams scores}. The lower the proximity score, the better.**

| Ranking | University Name | City | State | Proximity |
|---|---|---|---|---|
| 1 | Massachusetts Institute of Technology | Cambridge | MA | 0.0 |
| 2 | University of California Berkeley | Berkeley | CA | 0.004 |
| 3 | University of Wisconsin Madison | Madison | WI | 0.012 |
| 4 | Brown University | Providence | RI | 0.012 |
| 5 | Harvard University | Cambridge | MA | 0.016 |
| 6 | University of Pennsylvania | Philadelphia | PA | 0.016 |
| 7 | Stanford University | Stanford | CA | 0.0194883078787 |
| 8 | University of Michigan Ann Arbor | Ann Arbor | MI | 0.02 |
| 9 | Princeton University | Princeton | NJ | 0.02 |
| 10 | Carnegie Mellon University | Pittsburgh | PA | 0.024 |
| 11 | University of California San Diego | La Jolla | CA | 0.032 |
| 12 | University of California Santa Barbara | Santa Barbara | CA | 0.032 |
| 13 | University of Illinois Urbana-Champaign | Urbana | IL | 0.036 |
| 14 | University of Texas Austin | Austin | TX | 0.036 |
| 15 | Cornell University | Ithaca | NY | 0.036 |
| 16 | Duke University | Durham | NC | 0.036 |
| 17 | Rice University | Houston | TX | 0.036 |
| 18 | Johns Hopkins University | Baltimore | MD | 0.040704264472 |
| 19 | Northwestern University | Evanston | IL | 0.044 |
| 20 | University of Minnesota Twin Cities | Minneapolis | MN | 0.044 |
| 21 | Washington University | St. Louis | MO | 0.044 |
| 22 | Georgia Institute of Technology | Atlanta | GA | 0.048 |
| 23 | University of California Los Angeles | Los Angeles | CA | 0.048 |
| 24 | Vanderbilt University | Nashville | TN | 0.048 |
| 25 | University of Maryland College Park | College Park | MD | 0.064 |

students tend to move after graduation: academia or professional career.

We see potential in this project and will continue its development. First step that we plan to make is reinforcing our decision by collecting larger amounts of data. Users visiting our site will be encouriged to take a survey and provide us with additional learning information to make predictions more accurate.

We plan to add more features to the data and experiment with diffeent classification and regression algorithms to achieve maximum performance for this task. It is our goal to make this site useful for people like ourselves, be a source of reliable personalized information on postgraduate and potentially undergraduate education.

# 8. REFERENCES


[1] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. *Proceedings of ICML*, 2007.

[2] K. Crammer and Y. Singer. Pranking with ranking. *Proceedings of NIPS*, 2001.

[3] G. Lebanon and J. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. *Proceedings of ICML*, pages 363–370, 2002.

[4] D. Sculley. Combined regression and ranking. 2010.

[5] A. Shashua and A. Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. *Proceedings of NIPS*, 2002.

[6] F. Xia, T. Liu, J. Wang, and H. Li. Listwise approach to learning to rank - theory and algorithm. *Proceedings of ICML*, 2008.