Adam Smith
6.867, Fall 2005

Friendship Prediction on Facebook

Project Report

Overview
The Facebook, a popular college social networking website, contains gigabytes of data that can be mined to make predictions about who is a friend of whom. We crawl the facebook to gather this information, and train classifiers to make friendship predictions. The resulting predictions are tested using cross-validation, and a real world experiment.

Primer on the Facebook
The atomic element of information on the website is a profile. Each individual, as illustrated in Figure 2, can list different information about themselves on their profile (see Table 2). Although some of these fields are booleans (either selected or not selected), many of the fields can be missing from any given profile; this condition is shown as a value of "null."

Furthermore, individuals can link themselves in with others in various ways, as shown in Table 1.

Link Type	Example	Number Per Person (Avg)	Description
Friendship (undirected)	Adam and Eve are friends	83.9 friends	Two people mutually agree to be listed as friends
Photo albums	Frank, Don, and Sarah appear in picture #29308	7.2 photo appearances	Pictures can be tagged with the names of the people in them.
Classes	Bob is in 6.867 with 10 other people	4.24 classes	Users can list the classes they are taking.
Groups	"2009 Girls"	12.19 groups	Anyone can create a group and invite others to join.

Table 1. Types of links between people on the Facebook.

Two profiles are shown as examples in Figure 2. The data set is very irregular; some profiles do not contain much data, while others are elaborate.

Friendship links on the website are initiated by one of the two people; person A, after meeting person B, might find person B on the website and request to add her as a friend. Person B then receives an email, and can either accept the undirected friendship, or fail to accept it.

Figure 1. Thumbnails of two profiles. The one on the left is dense, while the one on the right is more sparse.

Field Name	Domain
Graduation Year	{null, 2004, 2005, 2006, 2007, 2008, 2009}
Status	{null, Undergrad, Alumus/Alumna, Faculty, Staff, Grad Student}
Sex	{null, Male, Female}
Major	{null, [44 possibilities]}
Double Major?	{yes, no}
Residence	{null, [64 possibilities]}
Home State	{null, [53 possibilities]}[1]
Looking for Friendship?	{yes, no}
Looking for Dating?	{yes, no}
Looking for Random Play?	{yes, no}
Looking for a Relationship?	{yes, no}
Looking for 'Whatever I can get'?	{yes, no}
Interested in Men?	{yes, no}
Interested in Women?	{yes, no}
Political Views	{null, Very Liberal, Liberal, Moderate, Conservative, Very Conservative, Apathetic, Libertarian, Other}

Table 2. Individual profile data. Note that there are also free text fields in profiles, which were not considered in this project.

The Problem Statement

The goal of the project is to create a friendship predictor. The data corpus contains information about everyone on the website, including both their individual profile data and their relationships to others. Using that corpus, we want to accurately predict, for each person, the people who they are friends with, such that these two people are not currently listed as being friends on the facebook.

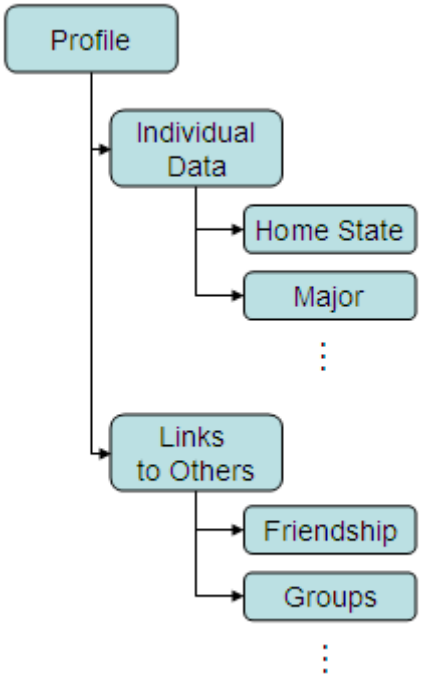

Figure 2. Structure of the information contained in each person's profile.

Challenges with the Data Set

Parameterizing the Data

Data points are taken to be friendships between two people. The first step to be taken, then, is to parameterize the facebook data into feature vectors.

Parameterizing individual profile data is not difficult; profiles contain a single value (or "null") for each field, such as graduation year, and these can be placed directly into a feature vector as an unordered value.

On the other hand, parameterizing the graphs expressed as links to others is more difficult. For example, Figure 3 expresses a friendship graph of the form encoded on the facebook. Powers of adjacency

[1] Including US territories such as Puerto Rico

matrices, or counts of paths of varying finite length from one node to every other, are used to encode the information in the graph.

For example, referring back to Figure 3, placing an arbitrary ordering on the nodes and writing down the adjacency matrix gives us the first matrix of Figure 4. Note that we use zeros along the diagonals, so that a node is not marked as being connected to itself. Also, since links are undirected, the adjacency matrix is symmetric.

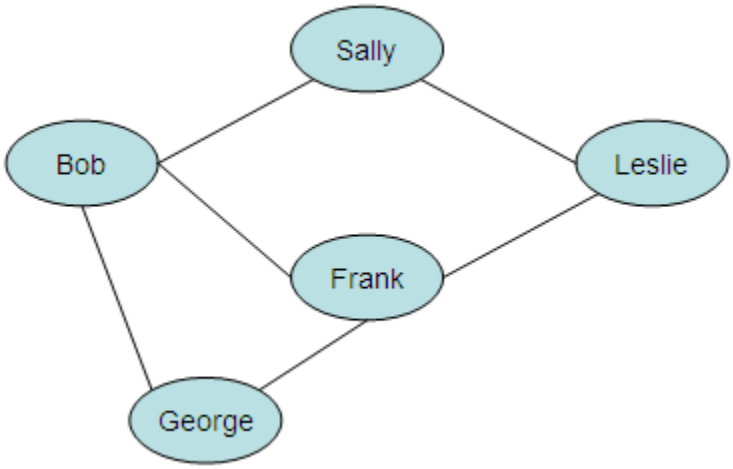

Figure 3. A graph encoding an example of a friendship network.

Now if we square the adjacency matrix, then every entry in the resultant denotes the number of paths of length 2 from i to j. For example, whereas Bob and Leslie were not previously connected, there are two paths of length two.

This process can be repeated for higher powers.

The link types other than friendship are bipartite graphs, however. For example, people are members of classes, so relationships between individuals are only mediated through classes. To parameterize these graphs we remove the intermediate nodes and create cliques containing the set of people who were connected to those intermediate nodes. For example, all people who are listed as in 6.867 are connected to each other in the resultant non-bipartite graph. Because people can be connected to more than one class, photo, or group, the edges can have positive integer weights.

```
x =

     0     1     0     1     1  ← Bob
     1     0     1     0     0  ← Sally
     0     1     0     1     0  ← Leslie
     1     0     1     0     1  ← Frank
     1     0     0     1     0  ← George

x^2 =

     3     0     2     1     1  ← Bob
     0     2     0     2     1  ← Sally
     2     0     2     0     1  ← Leslie
     1     2     0     3     1  ← Frank
     1     1     1     1     2  ← George
```

Figure 4. Example adjacency matrix corresponding to the friendship graph above.

Incomplete Dependent Variable Values

Since our goal is to build a friend predictor, presumably we need examples of “friends” and “not friends.” Unfortunately we do not have explicit examples of the latter case; people only list those that they are friends with. The problem, then, is to turn this unsupervised learning problem into a supervised learning problem.

For the remainder of the paper we will be referencing three states of friendship which are hard to syntactically denote. We will use the following naming convention

- "anti-friends" means that two people are not friends in real life
- "non-friends" means people that are not *listed* as being friends on the facebook. The true nature of their friendship of anti-friendship is not known.

The inexact solution to the non-friendship labeling problem was to consider all cases non-friendship as anti-friendship for the purposes of training a classifier. This approach seems paradoxical; since we will be predicting friendships that are not currently established, it is not intuitive that we should semantically interpret non-friendships to mean anti-friendships.

I will motivate this approach in two separate ways. The first is based on an approximate interpretation based on the sparse nature of friendship, and the second is a perspective from unsupervised learning.

First Approach

First, consider the case of non-friendship. The labeling of the true friendship nature is hidden, but two people in this state are either friends or anti-friends. Denote the probability that two non-friends are actually friends as 'p'. If we semantically interpret non-friends to mean anti-friends, then we will be wrong 'p' fraction of the time.

Empirically 'p' is a small value. Within the network of 5745 people there are 267,403 undirected friendships. These statistics imply that the friendship graph only has 1.6% of the possible number of links.[2] If the ratio of total friendships in the world to friendships expressed in the facebook is 'q', then

$p = (q - 1) * 0.016$ (this equation, p, and q will be referenced throughout the rest of this paper)

Applying domain knowledge, it seems that q = 2 or 3 is a good guess. The average number of friends is in the eighties, and while the distinction of friendship is subjective, based on experience with the facebook it is unlikely that less than one half or one third of friendships are expressed. Using, pessimistically, q = 3, we expect the non-friendship => anti-friendship assumption to be wrong 3.2% of the time, which is insignificant for our concerns.

Second Approach

In unsupervised learning, problems in which there is only one class present are usually solved using supervised learning algorithms and random, uniformly distributed, samples labeled with the opposing class. For example, when trying to find the structure or space that predominantly contains the red instances in Figure 5, the blue data points are added in uniformly and labeled with the opposing class. The resulting problem can then be solved using any supervised learning algorithm. The density of the false data to the real data determines how tightly the resulting boundaries fit.

[2] 267,403 / (5745 choose 2)

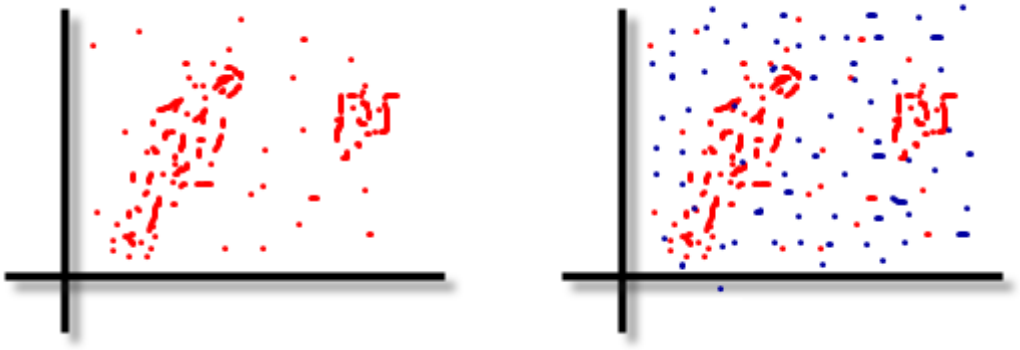

Figure 5. Unsupervised learning problem turned into a supervised learning problem using uniformly distributed false data labelled as a second class

Learning

Given the approaches outlined above, we ran the following two experiments. All learning was done using the Weka toolkit.

Experiment 1 – Cross validation within data set

In this experiment, all expressed friendships were collected along with an equal number of non-friendships (resulting in a data set of roughly half a million data points). Cross-validation was then executed on these sets, with the penalty matrix show in Table 3. We used a variant of the C4.5 decision tree algorithm with the default settings, which grows the decision tree and then prunes it.

		True	
Estimat.		Friends	Non-Friends
	Friends	0	0.95
	Non-Friends	1	0.05

Table 3. Penalty matrix for experiment 1 training. This encodes a belief of p = 0.05, where p is defined as in the last section.

The results are pretty positive. The correctness rate on the test data was consistently 89.x%. Although the decision trees consistently had over 12,000 decision points, one half of the resulting tree was small. That half is shown in Figure 6. Since the graph relationships tend to be higher in the tree than the individual profile data, one conclusion is that peoples' relationships to others is more indicative of friendship than their statistical data.[3]

[3] Intuition says that this relationship is true moreso because profile data is weak than because relational data is strong. This result might explain why dating websites have a hard time matching up partners based on form data, e.g. whether or not they want children.

```
f2hops <= 3
|   f2hops <= 0
|   |   photos2hops <= 0
|   |   |   classes <= 0: 0 (277479.0/5809.0)
|   |   |   classes > 0
|   |   |   |   p0_status = null
|   |   |   |   |   p1_status = null: 1 (19.0/2.0)
|   |   |   |   |   p1_status = 1
|   |   |   |   |   |   p0_lookingFor5 = null: 0 (0.0)
|   |   |   |   |   |   p0_lookingFor5 = 1
|   |   |   |   |   |   |   groups <= 0
|   |   |   |   |   |   |   |   p0_interestedIn2 = null: 0 (0.0)
|   |   |   |   |   |   |   |   p0_interestedIn2 = 1: 0 (21.0/1.0)
|   |   |   |   |   |   |   |   p0_interestedIn2 = 2: 1 (3.0/1.0)
|   |   |   |   |   |   |   groups > 0: 1 (2.0)
|   |   |   |   |   |   p0_lookingFor5 = 2: 1 (2.0)
|   |   |   |   |   p1_status = 2: 1 (0.0)
|   |   |   |   |   p1_status = 3: 1 (0.0)
|   |   |   |   |   p1_status = 4: 1 (0.0)
|   |   |   |   p0_status = 1: 0 (678.0/112.0)
|   |   |   |   p0_status = 2: 0 (0.0)
|   |   |   |   p0_status = 3: 1 (6.0/1.0)
|   |   |   |   p0_status = 4: 0 (0.0)
|   |   photos2hops > 0
|   |   |   photos <= 0: 0 (22.0/8.0)
|   |   |   photos > 0: 1 (4.0)
|   f2hops > 0
```

Figure 6. One half of resulting decision tree from experiment 1. f2hops is the value of the squared friendship adjacency matrix, as previously described. Photos is the adjacency matrix for photo relationships, and photos2hops is the squared version.

In interpreting the classification results to the domain of our problem (finding non-friends who really are friends), the idea is that non-friends that are classified as friends should be our predictions of new friends. In the confusion matrix below, this is the (1, 2) cell. These points, although listed as non-friends in the data sample, are so close to the instances of friends that they are "misclassified," but we interpret this misclassification as a hidden friendship.

The final question to consider for this experiment is the following. How many non-friend data points should we use in the training set for every friend data point? The ratio used was 1:1. The result is that 10% of the non-friend instances are classified as friends, implying a p value (as before) of 6.25. In retrospect, this is higher than we would have liked; if we were delivering a list of "these are your non-listed, true friends" then we should have given the training algorithm a higher ratio of non-friend data points, so that it would predict "friend" less often and we would get a lower p value.

Estimat.		**True**	
		Friends	Non-Friends
	Friends	416075	48341
	Non-Friends	40603	410173

Table 4. Confusion matrix for a single fold of cross validation (others are comparable).

Experiment 2 – Real world evaluation

While we were able to fit a good model to the data in experiment one, it is not clear what those results say about real-world performance.

A second experiment was ran in which a learning algorithm was trained on all instances of friendship in the data set, along with an equal number of randomly chosen instances of non-friendship. The resulting classifier was used by a website that asked people to give positive or negative feedback about each of ten friendship predictions made just for them.

In this case we used AdaBoostM1 and decision stumps as a weak learner as our learning algorithm. The ten chosen candidates for each visitor were the top ten predicted friends sorted on confidence. The C4.5 decision tree algorithm gave much more coarse confidence values for each test point than AdaBoostM1, so we switched algorithms between experiments 1 and 2. In practice, AdaBoostM1 performed slightly worse than C4.5 in experiment 1; its test data misclassification rates were in the 75-80% range instead of 89%. However, these results (and those from other algorithms) are comparable.

Screenshots from three of the four steps on the website are shown in the figures below. A user types in their name, predictions of their friends are computed, and the top ten are displayed asking for feedback. (The fourth page simply thanks the user.) Results were gathered from 52 users, although not all of them submitted feedback on all 10 of the candidates.

Users were separated randomly (with uniform distribution) into three different groups. One group used all available data, another group did not use the friendship graph data (i.e. number of 2-hop paths from person A to person B), and the final group did not use any of the graph data (only each individual's profile data).

Data Set	Number of People	Number Wrong	Number Right	Percent Right
All Data	20	122	56	31%
No Friendship	14	111	20	15%
No Graph	18	166	13	7%

Table 5. Results of experiment 2.

The results are less promising than those of experiment 1. Given that the confidence scores seem homogeneous (and thus not very reliable in sorting out the top ten candidates), the results were probably hurt mostly by the large number of positive test point classifications. In retrospect I should have added more non-friends points to reduce the implied p value of the classifier.

The final question becomes: how good are these results against a strawman? The strawman is choosing ten non-friends at random. Then each of the three data sets did better than the strawman, in expectation, if:

$$r > p$$

where r is the probability that the classifier returns a true friend (empirically, the fraction correct in Table 5). Given our pessimistic estimate of q = 3, the data set without graph data (only individual's profile data) does twice as well as the strawman. The classifier trained on all of the data performs an order of magnitude better.

Figure 7. The welcome page.

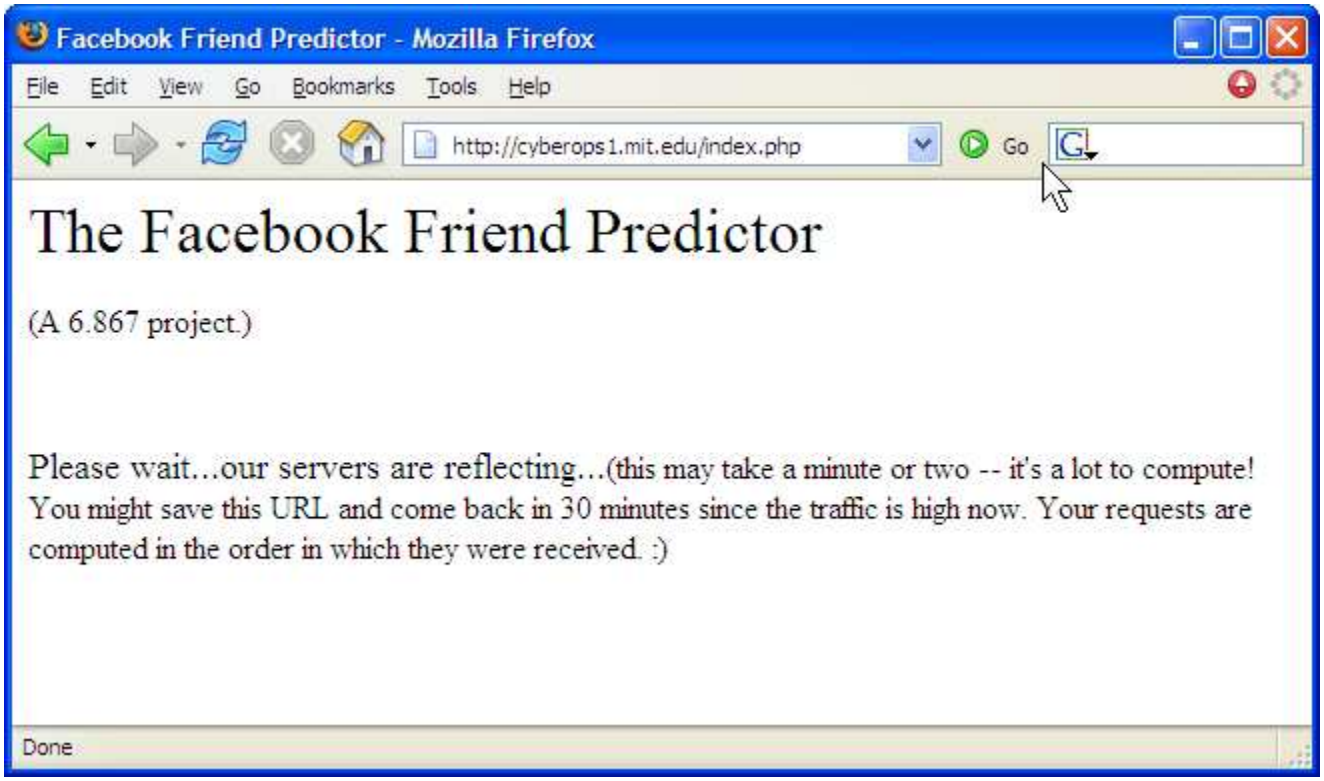

Figure 8. The "computing" page.

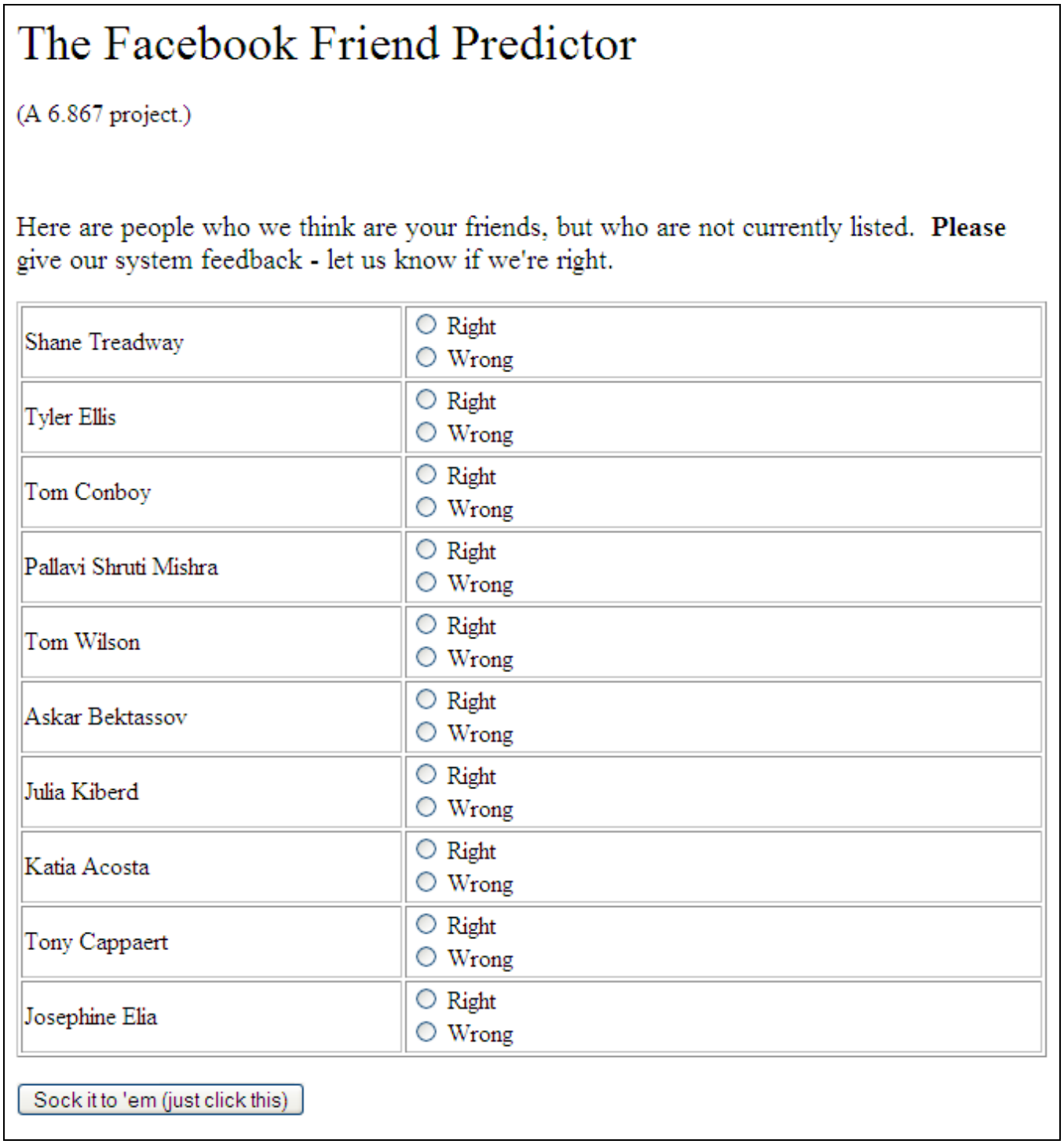

Figure 9. The results page, asking for feedback.

Miscellaneous Reflections

Resources Used

I used a variety of resources for this project. The use of a database (mysql) to manage most of the data was a love-hate relationship. It is very slow, but is useful for random queries and to get at-a-glance information.

I used two dedicated servers to do most of the work, who ran up an estimated 20 hours of computation time. The largest file ever used was over a gigabyte.

Time Spent

I didn't record the time, but retrospectively my estimates are:

- Crawling site (well) – 10 hours (I can talk about why in the interview)
- Parsing site – 2 hours
- Going from parsed data to feature vectors – 20 hours (incl. sparse symmetric matrix multiplication, etc.)
- Generating data sets and debugging feature vector generation (cached matrices, etc.), trying out various algorithms – 8 hours
- Setting up website apparatus – 8 hours
- Writeup – 4 hours

This is fairly close; in reality it should add up to about 55 or 60 hours. The kicker is that the majority of the time was spent on data gathering and preparation!

This project generated over 2000 lines of Java code, not counting database schema and website code.

Future Work

There were many things that I did not do. If I had more time, I would:

- Feed the anti-friends data from experiment 2 into a learning algorithm to see how much it can improve
- Try different ratios of non-friends data points to friends data points to control the implied p value
- Learn different classifiers for different individuals. Of course you also want some global data too, so it would be especially interesting to build a hybrid (vis-à-vis meta learning).