

# Chatty Stochastic Multi-Armed Bandits

Akshay Kumar

April 5, 2014

## **Abstract**

This thesis uses a variant of the classic stochastic multi-armed bandit framework to improve the user experience in an anonymous chat application online by selecting good conversation starters. While the traditional algorithm would converge on the "optimal" conversation starter and use it for every conversation, this novel version of the algorithm attempts to provide new conversation starters for each user while still attempting to maximize the conversation quality. This thesis examines the empirical behavior of such an algorithm in a web application deployed at Princeton University.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What is Tigers Anonymous? . . . . .	3
1.2	Why Multi-Armed Bandits? . . . . .	3
<b>2</b>	<b>Literature Review</b>	<b>5</b>
<b>3</b>	<b>Methods</b>	<b>8</b>
3.1	UCB1-AKSB Algorithm . . . . .	8
3.2	Tigers Anonymous Data Collection Methods . . . . .	8
3.3	Tigers Anonymous Data Format . . . . .	9
3.4	Tigers Anonymous UCB1-AKSB Implementation . . . . .	10
<b>4</b>	<b>Data Analysis</b>	<b>13</b>
4.1	Regret Analysis . . . . .	13
4.2	UCB1-AKSB Algorithm Efficacy . . . . .	16
4.3	Individual User Analysis . . . . .	17
<b>5</b>	<b>Conclusions</b>	<b>20</b>
5.1	Summary of Findings . . . . .	20
5.2	Potential Applications . . . . .	20
5.3	Further Improvements . . . . .	20
	<b>Appendices</b>	<b>21</b>
	<b>A Data Analysis Code</b>	<b>22</b>
	<b>B Conversation Starters</b>	<b>23</b>
	<b>C TA Back-End Implementation</b>	<b>26</b>
C.1	Princeton IP-Address Filtering Functionality . . . . .	26
C.2	User Matching Functionality . . . . .	27
C.3	Web Server Functionality . . . . .	31
C.4	Conversation Metadata Logging Model . . . . .	33

<b>D TA Front-End Implementation</b>	<b>34</b>
D.1 Homepage . . . . .	34
D.2 About Page . . . . .	37
D.3 Chatroom . . . . .	39
<b>Bibliography</b>	<b>42</b>

# Chapter 1

# Introduction

At most college campuses, as students become more settled within their college community, it becomes increasingly harder to branch out and meet people outside their immediate social graph. It was in response to such a problem that Tigers Anonymous (TA) was created. By providing a way to anonymously be matched with, chat with, and potentially meet fellow classmates, TA allows students to make new connections and shake up their social network while also providing a great way to develop and test a new, weakly context-dependent variant of the classic UCB1 multi-armed bandit algorithm (the UCB1-AKSB algorithm, described fully in chapter 3), which is the main focus on this thesis.

## 1.1 What is Tigers Anonymous?

Tigers Anonymous (TA) is the title of a chat application that allows any Princeton student to be matched with another Princeton student. After being matched, the students will be taken to an anonymous chatroom where they are given a conversation starter and have the opportunity to have a conversation. Once both participants have exchanged a pre-determined number of messages, a drop-down menu appears containing two choices (see Figure 1.1 below). If both users click "Yes", the application will authenticate both users via Facebook and reveal each users' identities to the other to facilitate communication outside of TA. For more information on how TA is implemented, see Appendices C and D.

## 1.2 Why Multi-Armed Bandits?

A significant part of the functionality of TA is providing a conversation starter to reduce the awkwardness of the initial interaction with an anonymous stranger online. A naive approach would simply choose conversation starters at random, but this approach would be less than optimal for two reasons. First, students might respond better to some conversation starters than others, so TA should be able to hone in on the best conversation starters and display them in order

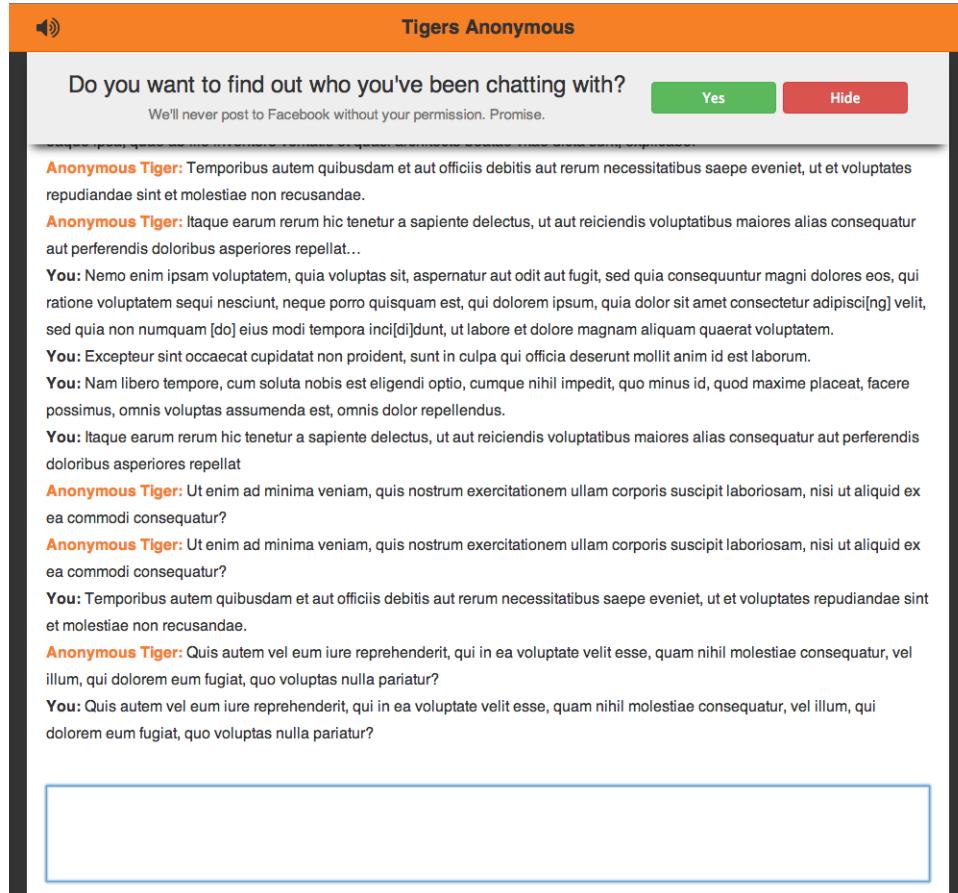


Figure 1.1: TA Drop-Down Menu

to facilitate higher quality conversations. Second, users could potentially see the same conversation starter more than once in a short period of time, which would defeat the purpose of having novel conversation starters.

This is where the multi-armed bandit problem comes in. By modeling conversation starters as "arms" in a classical multi-armed bandit problem and a "success" as a "high-quality" conversation, it should be possible to solve both of the above problems. For more information on the multi-armed bandit problem and the motivation for the UCB1-AKSB algorithm described in chapter 3, see chapter 2.

## Chapter 2

# Literature Review

In its most general form, the multi-armed bandit problem is a sequential decision-making problem where the decision-maker must explore new possibilities while trying to exploit existing knowledge to maximize the received payout. Using the notation in Bubeck and Cesa-Bianchi (2012), there are  $K \geq 2$  arms and sequences  $X_{i,1}, X_{i,2}, \dots$  for each arm  $i = 1, \dots, K$ . At each time step,  $t = 1, 2, \dots$ , the decision-maker picks  $I_t \in \{1, \dots, K\}$  and receives a payout  $X_{I_t,t}$ .

The common way of benchmarking the performance of a decision-making algorithm in this context is to take the distance between the expected payout of the algorithm and the optimal payout by play  $n$ . In the literature, optimality can be defined at two levels of granularity: play-by-play or by expectation. Using the first notion of optimality yields the expected regret, which is defined in Equation 2.1.

$$\mathbb{E}R_n = \mathbb{E} \left[ \max_{i=1, \dots, K} \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right] \quad (2.1)$$

A slightly weaker notion of regret comes from the second, more broad definition of optimality: optimality in expectation. Using this notion of optimality yields the pseudo-regret, which is defined in Equation 2.2.

$$\bar{R}_n = \max_{i=1, \dots, K} \mathbb{E} \left[ \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right] \quad (2.2)$$

It is important to notice here that in Equation 2.1, the algorithm is competing with the best possible random draw at every time-step, whereas in Equation 2.2, the algorithm is only competing with the arm with the highest expected value. This is the intuition for why  $\bar{R}_n \leq \mathbb{E}R_n$ .

Now that the general form of the multi-armed bandit problem has been introduced, there are three main sub-problems outlined by Bubeck and Cesa-Bianchi (2012) with further assumptions about the reward distributions. In the stochastic version, the sequences of plays  $X_{i,1}, X_{i,2}, \dots$  are I.I.D samples drawn

from distributions  $\nu_i \in [0, 1]$  for each arm  $i = 1, \dots, K$ . In the adversarial version, at each time step  $t$  an adversary selects a gain vector  $g_t = (g_{1,t}, \dots, g_{K,t}) \in [0, 1]^K$  such that  $X_{i,t} = g_{i,t}$ . In the Markovian version, the reward processes are neither i.i.d (as in the stochastic version) or chosen by an adversary (as in the adversarial version). Instead, each arm represents a Markov process with a state space  $S$ . Each time an arm  $i$  is chosen in state  $s$ , a reward is drawn from a probability distribution  $\nu_{i,s}$  and the state of the reward process for arm  $i$  changes in a Markovian fashion, based on an underlying stochastic transition matrix  $M_i$ .

FIXME REWORD.

With these three sub-problems in hand, we can turn to the problem at hand: choosing conversation starters for a pair of TA users. Under the simplifying assumption that Princeton students react similarly to different conversation starters, at first glance it makes sense to use the stochastic formulation of the multi-armed bandit problem because the rewards (in this case, Facebook de-anonymization) could reasonably be assumed to be I.I.D. Additionally, the UCB1 algorithm in Auer et al. (2002) is elegant, computationally easy to implement, and has an upper logarithmic bound on cumulative regret when applied to a stochastic multi-armed bandit problem.

While formulating TA conversation starters as a stochastic multi-armed bandit and using the UCB1 algorithm solves the first problem outlined in section 1.2 (i.e. picking the conversation starter that students would respond best to), it would actually exacerbate the second problem (i.e. seeing the same conversation starter repeatedly). Since the stochastic multi-armed bandit version of the problem assumes the reward distributions are I.I.D, it will simply hone in on the distribution with the highest expected value, and in the long term, will just pick the optimal arm at every play. In the context of TA, this would result in a single conversation starter always being displayed in the long term. This defeats the purpose of having a variety of conversation starters in the first place.

Since the arms chosen need to depend on the context (i.e. who the users are), a logical first choice would be to look at a contextual bandit algorithm formulation for some help. One example of such a personalized recommendation algorithm can be found in Li et al. (2010). Their algorithm observes the current user  $u_t$  and a set  $A_t$  of arms of actions together with their feature vectors  $x_{t,a}$  for  $a \in A$ . The vector  $x_{t,a}$  summarizes information of both the user  $u_t$  and arm  $a$ , and will be referred to as the context. However, such a model would rapidly get out of hand for the case of TA for a variety of reasons. First, there isn't just one user, but rather a pair of users, so using the LinUCB algorithm suggested by Li et al. (2010) would FAIL BECAUSE BLAH CONTEXTUAL BANDITS AREN'T A GOOD SOLUTION. - DEFEATS THE PURPOSE OF THE SIMPLIFYING ASSUMPTION THAT PRINCETON STUDENTS ARE SIMILAR -

Another possibility would be to model BANDITS WITH SIDE INFORMATION.

So this raises an important question: how do we take advantage of the simplifying assumption that Princeton students will respond similarly to any given conversation starter (i.e. the rewards are I.I.D) while still enforcing the

invariant that a user doesn't see the same conversation starters repeatedly? This was the motivation for me to create a UCB algorithm somewhere between the context-free UCB1 and the context-dependent LinUCB. This new algorithm (which I have named UCB1-AKSB) is weakly context dependent, in that it applies the context-free UCB1 algorithm but dynamically filters the arms over which the algorithm to only include arms that neither user has already seen. This intuitive explanation is formalized in chapter 3 below.

# Chapter 3

## Methods

### 3.1 UCB1-AKSB Algorithm

The multi-armed bandit algorithm used by Tigers Anonymous is a novel variant of the well-known UCB1 algorithm (Auer et al., 2002). The new algorithm is outlined below:

Before explaining the algorithm, it will be useful to introduce notation. Let the users be represented as the set  $U$  and the bandit arms as the set  $X$ . Let the set of arms that have already been played for user  $u \in U$  be represented by the set  $X^u \subset X$ . The goal of the UCB1-AKSB algorithm is to pick some arm  $x \in X$  given the pair of users  $u, v \in U$ . In this specific application, the goal is to pick the optimal conversation starter  $x \in X$ .

The UCB1-AKSB algorithm proceeds as follows: For each pair of users  $u, v \in U$ , we pick the conversation starter  $x$  such that

$$x = \operatorname{argmax}_{x \in (X^u \cup X^v)^c} f(x) \quad (3.1)$$

where

$$f(x) = \begin{cases} \bar{x} + \sqrt{\frac{2 \ln n}{n_x}} & : n_x > 0 \\ \infty & : n_x = 0 \end{cases} \quad (3.2)$$

In Equation 3.2,  $n_x$  is the number of times that conversation starter  $x$  has been played and  $n$  is the total number of conversation starters that have been shown. Note that ties are broken arbitrarily.

### 3.2 Tigers Anonymous Data Collection Methods

The complete data-collection method used for this thesis is outlined below:

1. Two users visit [www.tigersanonymous.com/chat](http://www.tigersanonymous.com/chat) from a Princeton IP address.
2. The users are directed to the chat server and are matched on a first-come, first-served basis.
3. A conversation starter is selected based on the UCB1-AKSB algorithm described above.
4. After either of the users disconnects, a 10-tuple representing the chat session is logged in a database (see Data Format section below for more details).

### 3.3 Tigers Anonymous Data Format

The data that will be collected can be represented by the vector of 10-tuples  $(x_i, y_i, t0_i, t1_i, q_i, b_i, c1_i, c2_i, m1_i, m2_i)$  where  $x_i$  and  $y_i$  represent the pseudonymous user ids of the two participants in the chat,  $t0_i$  and  $t1_i$  represent the start and end times of the conversation,  $q_i$  represents the conversation starter,  $b_i \in (0, 1)$  represents whether the drop-down menu was displayed (i.e. both chat participants exchanged more than a predefined number of messages),  $c1_i, c2_i \in (0, 1)$  represent whether users  $x_i$  and  $y_i$  opted to de-anonymize the conversation respectively and  $m1_i, m2_i \in (0, 1)$  represent the number of messages that user  $x_i$  and  $y_i$  sent respectively. The subscript  $i$  is unique for each conversation.

A sample of this data is shown below:

---

```
[{ "userID1" : "9a675a6f581fd1dfa0b982826e75b4f5", "userID2" :
  "a8262bb13e641e2bf5dcb3985b2061be", "question" : "Do you believe in
  love at first sight?", "startTime" : 1390873110621, "endTime" :
  1390873162944, "buttonDisplayed" : false, "user1Clicked" : false,
  "user2Clicked" : false, "user1MessagesSent" : 1,
  "user2MessagesSent" : 0, "_id" : "52e70a4ac43b6d020079e52d", "__v"
  : 0 },
{ "userID1" : "a8262bb13e641e2bf5dcb3985b2061be", "userID2" :
  "9a675a6f581fd1dfa0b982826e75b4f5", "question" : "Do you believe in
  soul mates?", "startTime" : 1390873219878, "endTime" :
  1390873263469, "buttonDisplayed" : false, "user1Clicked" : false,
  "user2Clicked" : false, "user1MessagesSent" : 1,
  "user2MessagesSent" : 2, "_id" : "52e70aaafc43b6d020079e52e", "__v"
  : 0 },
{ "userID1" : "27ac4f2d7e40b5249c2edcba19e21fb8", "userID2" :
  "370f85e443ad3ee24a879b1ce5a2b54b", "question" : "What is one thing
  you miss about being a kid?", "startTime" : 1390876198530,
  "endTime" : 1390876307059, "buttonDisplayed" : false,
  "user1Clicked" : false, "user2Clicked" : false, "user1MessagesSent"
  : 1, "user2MessagesSent" : 0, "_id" : "52e71693c43b6d020079e52f",
  "__v" : 0 },
```

---

```
{
  "userID1" : "370f85e443ad3ee24a879b1ce5a2b54b", "userID2" :
    "6e0fe76fcfa80cf2920bd5fc7717cf6dd", "question" : "What's one thing
    that you learned this week?", "startTime" : 1390881063228,
    "endTime" : 1390882681992, "buttonDisplayed" : true, "user1Clicked"
    : true, "user2Clicked" : true, "user1MessagesSent" : 33,
    "user2MessagesSent" : 37, "_id" : "52e72f79c43b6d020079e531", "__v"
    : 0 },
...
]
```

---

### 3.4 Tigers Anonymous UCB1-AKSB Implementation

This is the code on the Tigers Anonymous server that implements the UCB1-AKSB algorithm.

---

```
var questions = require('./questions').list;

// Used in lieu of positive and negative infinity
var largePositiveNumber = 1000000000;
var largeNegativeNumber = -1000000000;

// UCB1 function to pick opening question
exports.getQuestion = function(collection, user1, user2, callback) {
  var questionAsked = {
    $or: [
      {$eq: ["$userID1", user1.id]},
      {$eq: ["$userID2", user1.id]},
      {$eq: ["$userID1", user2.id]},
      {$eq: ["$userID2", user2.id]}
    ]
  };
  var outputFormat = {
    _id: "$question",
    plays: {$sum: 1},
    wins: {$sum: {$cond: [{ $and: ["$user1Clicked", "$user2Clicked"]}], 1,
      0}}},
    timesShown: {$sum: {$cond: [questionAsked, 1, 0]}}
  };

  // Aggregate conversation data and call UCB callback
  collection.aggregate().group(outputFormat).exec(function(err, data) {
    if (err) console.log(err);
    UCB1(data, callback);
  });
}
```

```

// Helper function to get a random question
var getRandomQuestion = function() {
    var randomIndex = Math.floor(Math.random() * questions.length);
    return questions[randomIndex];
};

// Helper function to invoke callback on the data item with the max UCB
// value
var UCB1 = function(data, callback) {
    var finalData = {};

    // If there's no data, return a random question
    if (data.length === 0) {
        callback(getRandomQuestion());
        return;
    } else {
        // Otherwise, get all the available data for the questions and run
        // UCB
        var questionStats = {};
        var totalPlays = 0;

        // For each entry in data, sum the total number of plays and
        // populate the questionStats table with the corresponding question
        for (var i = 0; i < data.length; i++) {
            var entry = data[i];
            questionStats[entry._id] = {
                plays: entry.plays,
                wins: entry.wins,
                shown: (entry.timesShown > 0 ? true : false)
            };
            totalPlays += entry.plays;
        }

        for (var i = 0; i < questions.length; i++) {
            var question = questions[i];
            // If there's no data for this question, then it hasn't been
            // displayed yet, so assign it an arbitrarily large UCB value
            if (!questionStats[question]) {
                finalData[question] = largePositiveNumber;
            } else if (questionStats[question].shown) {
                continue;
            } else {
                // If the question hasn't been shown and there's data for it,
                // compute the UCB value
                var probabilityEstimate =
                    questionStats[question].wins / questionStats[question].plays;
                var UCBoundEstimate =
                    Math.sqrt(2 * Math.log(totalPlays /
                        questionStats[question].plays));
                finalData[question] = probabilityEstimate + UCBoundEstimate;
            }
        }
    }
}

```

```
        }
    }

    if (Object.keys(finalData).length > 0) {
        // Find question with max UCB value
        var bestValue = largeNegativeNumber;
        var bestMatch = null;
        for (var question in finalData) {
            var currentValue = finalData[question];
            if (currentValue >= bestValue) {
                bestMatch = question;
                bestValue = currentValue;
            }
        }
        callback(bestMatch);
    } else {
        callback(getRandomQuestion());
    }
}
};
```

---

# Chapter 4

# Data Analysis

## 4.1 Regret Analysis

Recall from chapter 2 that the cumulative regret of the UCB1 algorithm is proportional to  $\log(n)$ , where  $n$  is the number of plays. Thus, we should expect the UCB1-AKSB algorithm to result in a cumulative pseudo-regret that is approximately logarithmic in the number of plays. This empirical pseudo-regret,  $\hat{R}_n$ , was calculated using Equation 4.1 below under the assumptions that the long-term average de-anonymization proportion was optimal.

$$\hat{R}_n = \mu^* n - \sum_{t=1}^n X_{I_t, t} \quad (4.1)$$

In Equation 4.1,  $\mu^* n$  is the expected optimal number of de-anonymizations by play  $n$  (i.e.  $\max_{i=1,\dots,K} \mathbb{E}[\sum_{t=1}^n X_{i,t}]$  from Equation 2.2). The plot of  $\hat{R}_n$  as a function of  $n$  for the UCB1-AKSB algorithm is shown below in Figure 4.1.

The regret looks approximately logarithmic for the second half of the dataset, but the first half of the data gives a steadily negative cumulative regret. This is due to the fact that the I.I.D assumption of conversation de-anonymization is most-likely false. Instead, there were probably different regimes in which people perceived Facebook de-anonymization differently. This is most analogous to the Markovian bandits in Bubeck and Cesa-Bianchi (2012), where each conversation starter (i.e. bandit arm) is associated with a Markov process with a discrete set of distributions.

Because of the clear split in the data, it seems like there were two discrete distributions from which de-anonymizations were drawn. The first distribution occurred in the initial stages of TA's launch, where users were more likely to de-anonymize a conversation simply because of the novelty of doing so. This is supported by looking at the initial cumulative Facebook de-anonymization statistics (see Figure 4.2), where the Facebook connect rate was almost double the long-term average. The second distribution most likely occurred as users became more used to the idea of Facebook de-anonymization, the rate of de-

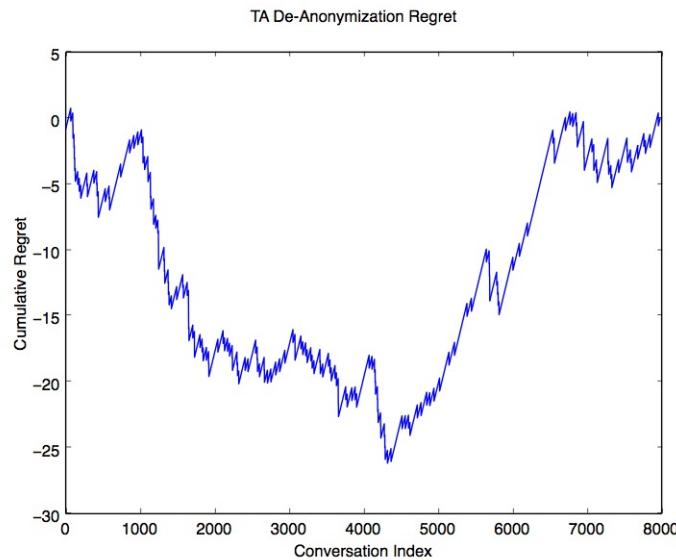


Figure 4.1: TA De-Anonymization Regret Analysis

anonymization drifted slowly back towards the long term average (around the second half of the data set). The existence of multiple regimes explains the two polar opposite segments of regret data.

ADDITIONALLY, DO THE SECOND HALF REGRET GRAPH HERE!!!  
FIXME

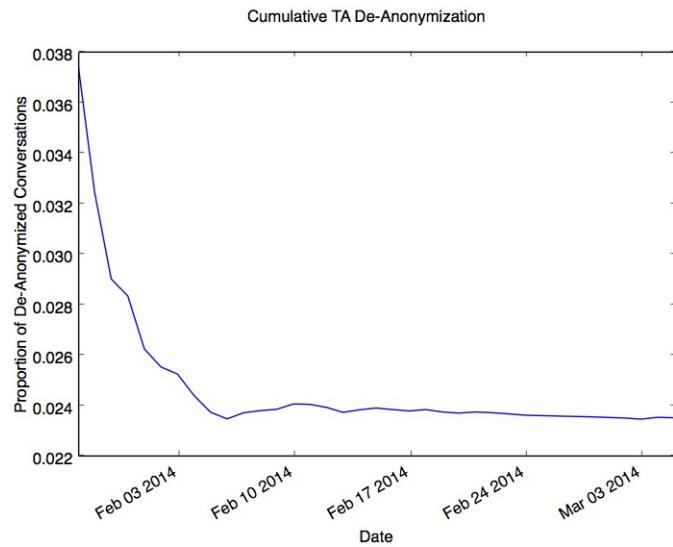


Figure 4.2: TA Cumulative De-Anonymization Data

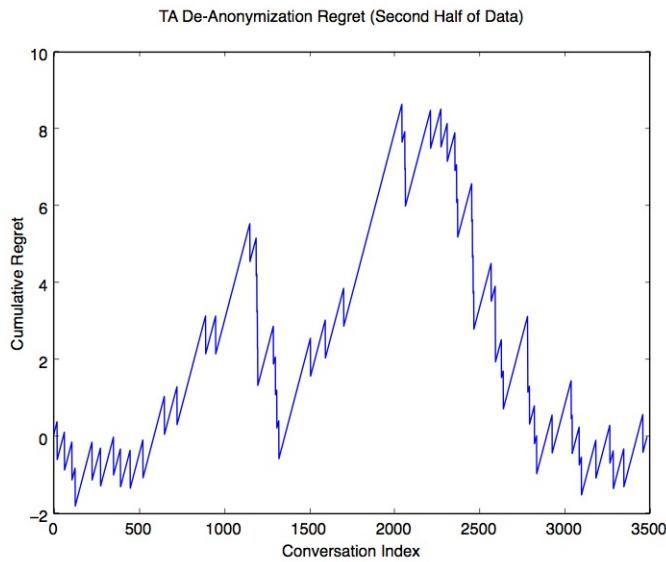


Figure 4.3: TA Cumulative De-Anonymization Data (For Second Half of Data Set)

## 4.2 UCB1-AKSB Algorithm Efficacy

Since the original goal of the UCB1-AKSB algorithm was to improve conversation quality (for which conversation de-anonymization served as a proxy), the efficacy of this algorithm will be assessed by the long-term proportion of conversation de-anonymization. ~~FIXME DAILY GRAPH AND REFERENCE CUMULATIVE GRAPH~~

In addition to using conversation de-anonymization as a metric of success, it makes sense to use other proxies for conversation quality (such as participation rates and average conversation length), because they are more fine-grained than the binary variable of conversation de-anonymization. The plots are below.

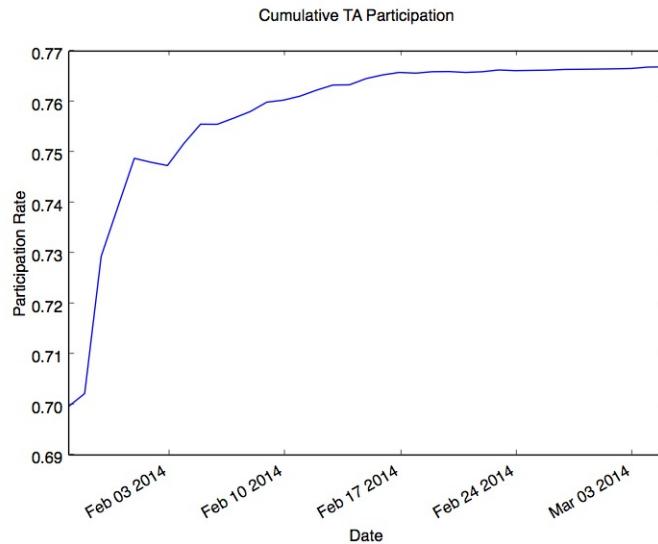


Figure 4.4: TA Cumulative Participation Rate

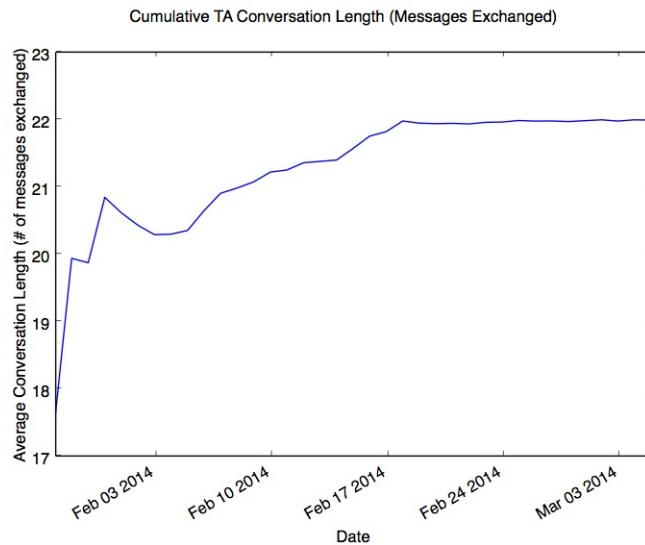


Figure 4.5: TA Cumulative Average Conversation Length

#### INTERPRET GRAPHS

### 4.3 Individual User Analysis

Another way of examining the data is to look at how individual users interacted with the site over the course of their site use.

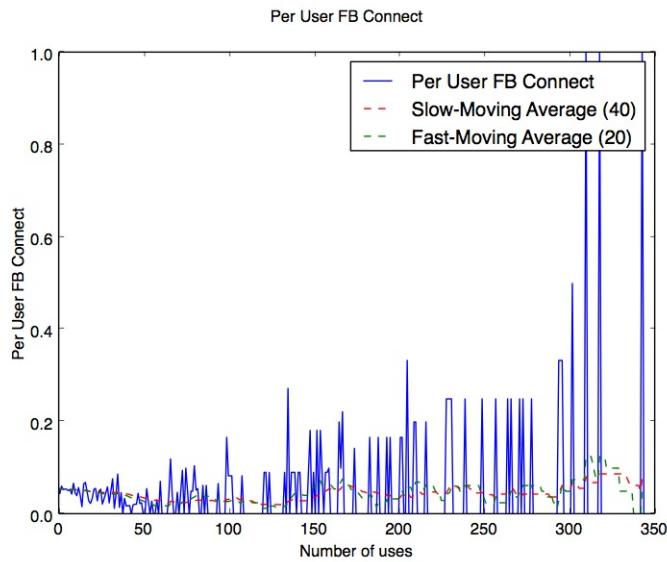


Figure 4.6: Per User De-Anonymization Proportion

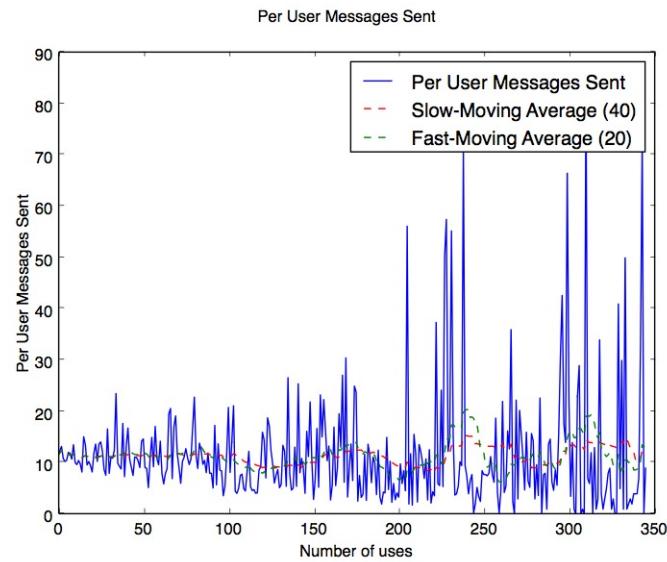


Figure 4.7: Per User Messages Sent

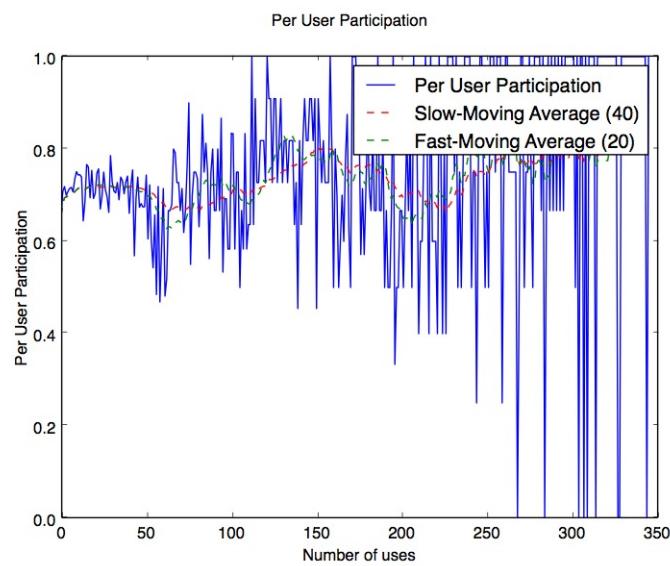


Figure 4.8: Per User Participation

# **Chapter 5**

# **Conclusions**

## **5.1 Summary of Findings**

## **5.2 Potential Applications**

A weakly contextual bandit that requires less computation Could be used for recommendation algorithms that need to recommend newly generated content (such as blog posts, news websites, etc.)

## **5.3 Further Improvements**

The methodology of this study could be improved by increasing the user space

For a fixed arm space (i.e. Tigers Anonymous conversation starters), a possible improvement to the UCB1-AKSB algorithm would be to have a moving time-window, so that both users would be guaranteed to see a conversation starter that they haven't seen in at least 5 uses or 5 days (i.e. number of uses or a fixed time length)

# Appendices

# Appendix A

## Data Analysis Code

---

FIXME: ADD CODE HERE

---

# Appendix B

## Conversation Starters

The following is the array of conversation starters used in Tigers Anonymous.

---

```
exports.list = [
    "What animal is your Patronus?",
    "If you ruled the world, what laws would you make?",
    "What was your last dream about?",
    "What would you do if you won the lottery?",
    "What does your dream house look like?",
    "What was your favorite vacation?",
    "If you could go back in time to change one thing what would it be?",
    "What's the greatest invention of all time?",
    "Have you ever been admitted to the hospital?",
    "Have you ever had any brushes with the law?",
    "What's the best practical joke you've played on someone?",
    "What's the best practical joke someone's pulled on you?",
    "What is your best achievement to date?",
    "If you could live anywhere, where would it be?",
    "What's your favorite song?",
    "What's your favorite word (inappropriate or otherwise)?",
    "What's the longest period of time you've gone without sleep?",
    "Do you have any scars?",
    "If you could change anything about yourself what would it be?",
    "Would you rather trade some intelligence for looks or looks for
        intelligence?",
    "Have you ever had a secret admirer?",
    "If you could ask your future self one question, what would it be?",
    "Are you a good liar?",
    "What's your favorite joke?",
    "What's the worst present you've ever gotten?",
    "What's your favourite saying?",
    "Have you ever accidentally injured anyone?",
    "What cartoon character would you love to see in 21st century life?",
    "What's the word you use most often?",
```

"What's your dream job?",  
"Which song annoys you the most?",  
"What's your first thought when you wake up?",  
"If you could steal one thing in the world, what would it be?",  
"What's your favorite Pokemon character?",  
"When did you stop believing in Santa?",  
"What's your favorite Disney movie?",  
"What's your life motto?",  
"What's the most unusual thing you've ever eaten?",  
"Do you collect anything?",  
"What thing would you like to bring back into fashion?",  
"What makes you nervous?",  
"What's the worst pickup line you've ever heard?",  
"What do you do when you forget someone's name immediately after  
they've introduced themselves?",  
"Have you ever been in a fight?",  
"Have you ever started a rumor?",  
"What's the most memorable rumor you've heard about yourself?",  
"Is there anything about the opposite sex you just don't understand?",  
"If you had a warning label, what would yours say?",  
"Which fictional character do you wish was real?",  
"Who was your first crush?",  
"Do you believe in destiny or free will?",  
"What's the best decision you've made so far?",  
"Who would you want to be with on a desert island?",  
"If you had to pick a new name for a week, what would it be?",  
"What is your first memory?",  
"Where did you go on your first ride on an airplane?",  
"Who was your first best friend?",  
"What was your first detention for?",  
"What would be the name of your debut solo album?",  
"What's something you get compulsive about?",  
"Have you ever stolen anything?",  
"What was the last social faux pas you made?",  
"What makes you nostalgic?",  
"What's the scariest thing you've ever done?",  
"What fairy tale character do you most associate with?",  
"What's the craziest thing you've ever done for someone?",  
"What's the best piece of advice anyone has ever given you?",  
"Do you have a Princeton bucket list?",  
"What's your favorite memory at Princeton?",  
"What building would you donate to Princeton?",  
"What is one thing you always wanted as a kid, but never got?",  
"What is the nicest thing someone else has done for you?",  
"If you could time travel, what would you do?",  
"If you went to a psychiatrist, what would he/she say you suffer  
from?",  
"What one thing annoys you most at a restaurant?",  
"What do Princeton students do too much of today?",  
"What would you like to spend more time doing?"

"If you could dis-invent one thing, what would it be?",  
"How would you dispose of a dead body?",  
"What's the most recent dream you can remember?",  
"What's something about you that people wouldn't expect?",  
"If you could change one thing about the world, what would it be?",  
"What's your favorite genre of music?",  
"If you could eat lunch with one famous person, who would it be?",  
"How are you feeling right now?",  
"What do you think about the most?",  
"Do you sing in the shower?",  
"Before Princeton, what did you want to be when you grew up?",  
"What is your best childhood memory?",  
"What's something embarrassing that happened to you?",  
"If you could live in any city in the world, where would it be?",  
"Where do you want to travel to?",  
"What's something spontaneous that you've done?",  
"If you could only eat one food for the rest of your life, what would it be?",  
"What's your biggest pet peeve?",  
"What was the happiest moment in your life?",  
"What quality about yourself do you value most?",  
"What are you most proud about in your life?",  
"What is your biggest concern about the future?",  
"What is the biggest lesson you've learned in life thus far?",  
"Do you think people can control their own destinies?",  
"How is your relationship with your parents?",  
"If you could go back and relive a day in your life, what would you change?",  
"What is the weirdest thing about you?",  
"If you could have any superpower, which one would you pick?",  
"What is the last thing you do before you go to sleep?",  
"Whats the first thing you notice when you meet someone new?",  
"Whats one of your worst habits?",  
"If your house was on fire, what's the one thing you'd want to take with you?",  
"If money was no object, what would you be doing with your life?",  
"What does your vision of a utopian society look like?",  
"If you only had one day left to live, what would you do?",  
"What's one thing that you learned this week?",  
"What was the last thing you thought about last night?",  
"What were you like as a kid?",  
"What is one thing you miss about being a kid?",  
"Do you believe in soul mates?",  
"Do you believe in love at first sight?",  
"What's one thing you'd like to change about Princeton?",  
"How was your RCA during your freshman year?"  
];

---

# Appendix C

# TA Back-End Implementation

The following pieces of code implement the back-end and front-end functionality of Tigers Anonymous unrelated to the UCB1-AKSB algorithm.

## C.1 Princeton IP-Address Filtering Functionality

---

```
var range_check = require('range_check');

// Pre-defined Princeton IP address blocks
var princetonIPs = [
    "128.112.0.0/16",
    "140.180.0.0/16",
    "204.153.48.0/22",
    "66.180.176.0/24",
    "66.180.177.0/24",
    "66.180.180.0/22"
];

// Check to ensure that the user's IP is a valid Princeton IP
var isValidIP = function (userIP) {
    if (userIP === "127.0.0.1" || // for debugging
        range_check.in_range(userIP, "192.168.0.0/16") ||
        range_check.in_range(userIP, "10.0.0.0/8")) {
        return true;
    }
    for (var i = 0; i < princetonIPs.length; i++) {
        if (range_check.in_range(userIP, princetonIPs[i])) {
```

```

        return true;
    }
}
return false;
}

exports.isValidIP = isValidIP;

```

---

## C.2 User Matching Functionality

---

```

var mongoose = require('mongoose');
var Conversation = mongoose.model('Conversation');
var ucb = require('./ucb');
var mailer = require('./mailer');

function User(socket, userID) {
    this.socket = socket;
    this.id = userID;
    this.partner = null;
    this.conversation = null;
    this.buttonClicked = false;
    this.messagesSent = 0;
    this.name = null;
    this.fbLink = null;

    var user = this;
    this.socket.on('disconnect', function() {
        if (!user.conversation) return;

        if (!user.conversation.endTime) {
            user.conversation.chatLog.push({
                date: new Date(),
                user: '',
                text: '*** ' + user.pseudonym + ' disconnected ***'
            });
        }

        user.conversation.endTime = new Date();
        user.conversation.save();
        user.partner.socket.emit('finished');
        user.partner.socket.disconnect();
    });
}

this.socket.on('chat message', function(data) {
    if (!user.conversation) return;

```

```
user.conversation.chatLog.push({
  date: new Date(),
  user: user.pseudonym,
  text: data.message
});

user.messagesSent++;
user.socket.emit('chat message', {
  name: 'You',
  message: data.message
});

var userName = user.conversation.revealed ? user.name : 'Anonymous
Tiger';
user.partner.socket.emit('chat message', {
  name: userName,
  message: data.message
});
});

this.socket.on('dropdown displayed', function(data) {
  if (!user.conversation) return;

  user.conversation.buttonDisplayed = true;
});

this.socket.on('identity', function(data) {
  if (!user.conversation) return;

  user.conversation.chatLog.push({
    date: new Date(),
    user: '',
    text: '*** ' + user.pseudonym + ' accepted dropdown ***'
  });

  user.name = data.name;
  user.fbLink = data.link;
  user.buttonClicked = true;

  if (user.partner.buttonClicked) {
    user.socket.emit('reveal', {
      name: user.partner.name,
      link: user.partner.fbLink
    });
    user.partner.socket.emit('reveal', {
      name: user.name,
      link: user.fbLink
    });
    user.conversation.revealed = true;
  }
});
```

```

        user.conversation.chatLog.push({
            date: new Date(),
            user: '',
            text: '*** Facebook identities revealed ***'
        });
    }
});

this.socket.on('typing', function() {
    if (!user.conversation) return;

    user.partner.socket.emit('typing');
});

this.socket.on('not typing', function() {
    if (!user.conversation) return;

    user.partner.socket.emit('not typing');
});
}

function ConversationWrapper() {
    this.user1 = null;
    this.user2 = null;
    this.startTime = new Date();
    this.endTime = null;
    this.question = null;
    this.buttonDisplayed = false;
    this.revealed = false;
    this.chatLog = [];

    var self = this;
    this.save = function() {
        new Conversation({
            userID1: self.user1.id,
            userID2: self.user2.id,
            question: self.question,
            startTime: self.startTime,
            endTime: self.endTime,
            buttonDisplayed: self.buttonDisplayed,
            user1Clicked: self.user1.buttonClicked,
            user2Clicked: self.user2.buttonClicked,
            user1MessagesSent: self.user1.messagesSent,
            user2MessagesSent: self.user2.messagesSent
        }).save();

        if (process.env.NODE_ENV === 'production') {
            mailer.sendMail(this);
        }
    };
}

```

```
}

var queue = new Array();
exports.connectChatter = function(socket, userID) {
    var user = new User(socket, userID);

    user.socket.emit('entrance');
    user.socket.emit('waiting');

    if (queue.length === 0) {
        queue.push(user);

        user.socket.on('disconnect', function() {
            var index = queue.indexOf(user);
            if (index !== -1) {
                queue.splice(index, 1);
            }
        });
    } else {
        var conversation = new ConversationWrapper();
        conversation.user1 = user;
        user.conversation = conversation;
        user.pseudonym = 'Origin';

        var partner = queue.shift();
        user.partner = partner;
        partner.partner = user;
        conversation.user2 = partner;
        partner.conversation = conversation;
        partner.pseudonym = 'Black';

        ucb.getQuestion(Conversation, user, partner, function(question) {
            user.conversation.question = question;
            user.socket.emit('matched', {
                question: question
            });
            partner.socket.emit('matched', {
                question: question
            });

            conversation.chatLog.push({
                date: new Date(),
                user: '',
                text: question
            });
        });
    }
};
```

---

### C.3 Web Server Functionality

---

```

var express = require('express'),
    app = express(),
    server = require('http').createServer(app),
    io = require('socket.io').listen(server);
mongoose = require('mongoose'),
princeton = require('./server/princeton'),
conversation = require('./server/conversation'),
chatter = require('./server/chatter');

var port = process.env.PORT || 5000;
server.listen(port);

var mongoUrl;
io.configure('development', function() {
    mongoUrl = 'mongodb://localhost/test';
});
io.configure('production', function() {
    mongoUrl = process.env.MONGOHQ_URL;
});
mongoose.connect(mongoUrl);

var connectedUsers = {};

app.get('/count', function(req, res) {
    var count = Object.keys(connectedUsers).length;
    res.send(count.toString());
});

io.configure('production', function() {
    io.set('log level', 1);
    io.set('transports', ['websocket']);

    io.set('authorization', function(handshakeData, callback) {
        // Check if Princeton IP
        var ipAddr = getClientIP(handshakeData);
        var isValidIP = princeton.isValidIP(ipAddr);
        if (!isValidIP) {
            callback('Sorry, this site is only for Princeton students!',
                    false);
            return;
        }

        // Check if already connected to server
        if (ipAddr in connectedUsers) {
            callback('Sorry, you can only chat with one person at a time!',
                    false);
        }
    });
});

```

```

        return;
    }

    callback(null, true);
});

// Needed to get the client's IP on Heroku for socket.io
function getClientIP(handshakeData) {
    var forwardedIps = handshakeData.headers['x-forwarded-for'];
    if (forwardedIps) {
        return forwardedIps.split(', ')[0];
    } else {
        return handshakeData.address.address;
    }
}

function getValueFromCookie(name, cookie) {
    if (cookie) {
        var pairs = cookie.split('; ');
        for (var i = 0; i < pairs.length; i++) {
            var pair = pairs[i].split('=");
            if (pair[0] === name) {
                return pair[1];
            }
        }
    }
}

io.sockets.on('connection', function(socket) {
    var userID = getValueFromCookie('userID',
        socket.handshake.headers.cookie);
    if (userID) {
        // Add user to list of connected users
        var ipAddr = getClientIP(socket.handshake);
        connectedUsers[ipAddr] = true;
        socket.on('disconnect', function() {
            delete connectedUsers[ipAddr];
        });

        chatter.connectChatter(socket, userID);
    } else {
        socket.disconnect();
    }
});

```

---

## C.4 Conversation Metadata Logging Model

---

```
var mongoose = require('mongoose');

var conversationSchema = new mongoose.Schema({
  userID1: String,
  userID2: String,
  startTime: Date,
  endTime: Date,
  question: String,
  buttonDisplayed: Boolean,
  user1Clicked: Boolean,
  user2Clicked: Boolean,
  user1MessagesSent: Number,
  user2MessagesSent: Number
});

mongoose.model('Conversation', conversationSchema);
```

---

## Appendix D

# TA Front-End Implementation

### D.1 Homepage

The homepage (shown below in Figure D.1) is implemented with the code shown at the bottom of this section.

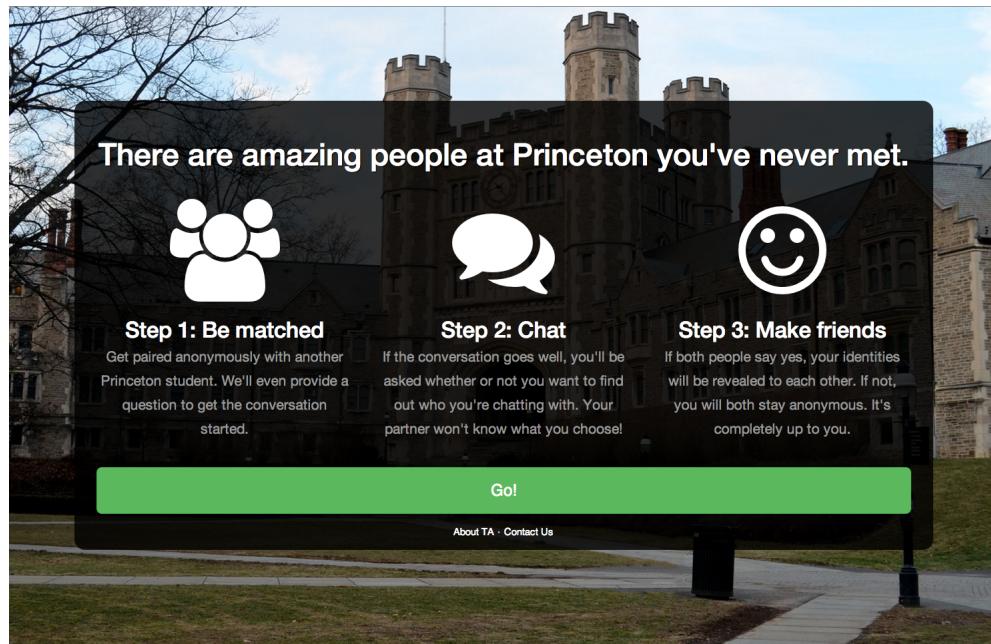


Figure D.1: Tigers Anonymous Homepage

---

```

<!DOCTYPE html>
<html>
  <head prefix="og: http://ogp.me/ns#">
    <title>Tigers Anonymous</title>
    <link rel="icon" href="img/favicon.ico" type="image/x-icon">
    <meta name="viewport" content="width=device-width,
      initial-scale=1.0, user-scalable=no">
    <meta property="og:title" content="Tigers Anonymous">
    <meta property="og:description" content="There are amazing people at
      Princeton you've never met.">
    <meta property="og:url" content="http://www.tigersanonymous.com">
    <meta property="og:image"
      content="http://www.tigersanonymous.com/img/ta1024.png">
    <meta property="og:image:type" content="image/png">
    <meta property="og:image:width" content="1024">
    <meta property="og:image:height" content="1024">
    <link rel="stylesheet"
      href="//netdna.bootstrapcdncdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
    <link
      href="//netdna.bootstrapcdncdn.com/font-awesome/4.0.3/css/font-awesome.css"
      rel="stylesheet">
    <link href="css/index.css" rel="stylesheet" type="text/css"
      media="all">
    <script>
      (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
        (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
        Date();a=s.createElement(o),
        m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
      })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
      ga('create', 'UA-23357698-2', 'tigersanonymous.com');
      ga('send', 'pageview');
    </script>
  </head>
  <body class="cover">
    <div class="wrapper">
      <div class="container">
        <div class="row text-center">
          <div class="col-md-12">
            <h1 class="hook">There are amazing people at Princeton you've
              never met.</h1>
          </div>
        </div>
        <div class="row how-it-works">
          <div class="col-md-4">
            <div class="row text-center padded-icon">
              <i class="fa fa-users large-icon"></i>
            </div>
            <div class="row text-center padded-text">

```

```
<h2>
  Step 1: Be matched<br>
  <small>Get paired anonymously with another Princeton
        student. We'll even provide a question to get the
        conversation started.</small>
</h2>
</div>
</div>
<div class="col-md-4">
  <div class="row text-center padded-icon">
    <i class="fa fa-comments large-icon"></i>
  </div>
  <div class="row text-center padded-text">
    <h2>
      Step 2: Chat<br>
      <small>If the conversation goes well, you'll be asked
            whether or not you want to find out who you're
            chatting with. Your partner won't know what you
            choose!</small>
    </h2>
    </div>
  </div>
<div class="col-md-4">
  <div class="row text-center padded-icon">
    <i class="fa fa-smile-o large-icon"></i>
  </div>
  <div class="row text-center padded-text">
    <h2>
      Step 3: Make friends<br>
      <small>If both people say yes, your identities will be
            revealed to each other. If not, you will both stay
            anonymous. It's completely up to you.</small>
    </h2>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-md-12">
    <a href="/chat" class="go-btn btn btn-success btn-xlg
      btn-block">Go!</a>
  </div>
</div>
<div class="row text-center">
  <div class="col-md-12 footer">
    <a href="/about">About TA </a>
    &#8901;
    <a href="mailto:originblack609@gmail.com"> Contact Us</a>
  </div>
</div>
</div>
```

---

```
</div>
</body>
</html>
```

---

## D.2 About Page

The About page (shown below in Figure D.2) is implemented with the code shown at the bottom of this section.

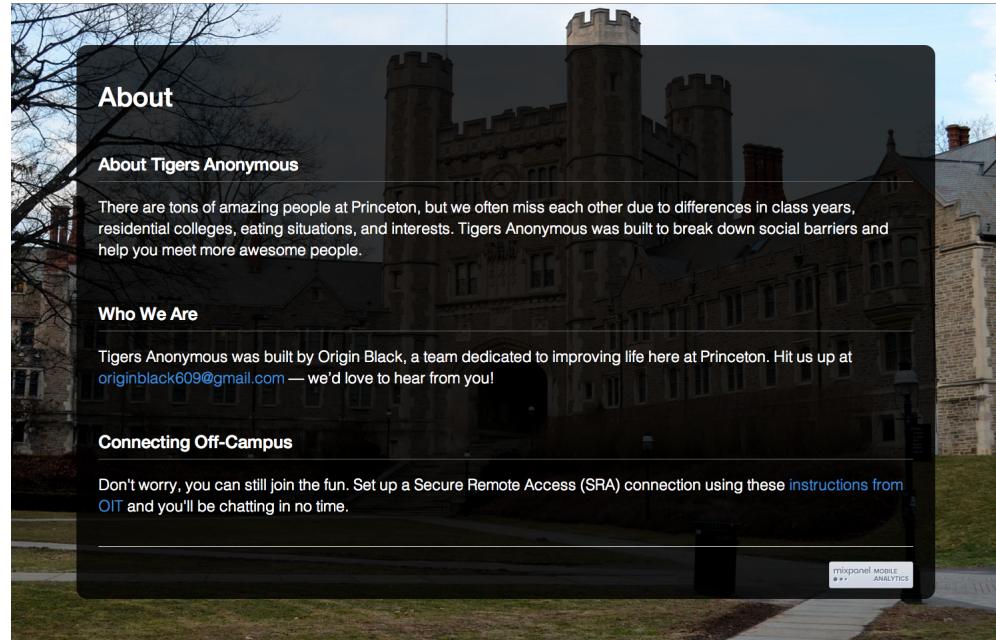


Figure D.2: Tigers Anonymous About Page

---

```
<!DOCTYPE html>
<html>
<head>
    <title>About - Tigers Anonymous</title>
    <link rel="icon" href="img/favicon.ico" type="image/x-icon">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0, user-scalable=no">
    <link rel="stylesheet"
        href="//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
    <link href="css/index.css" rel="stylesheet" type="text/css"
        media="all">
    <script>
        (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
```

```

(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','','//www.google-analytics.com/analytics.js','ga');
ga('create', 'UA-23357698-2', 'tigersanonymous.com');
ga('send', 'pageview');

</script>
</head>
<body class="cover">
<div class="wrapper">
<div class="container">
<div class="row">
<div class="col-md-12">
<h1>About</h1>
<div class="header" id="about">
<h3>About Tigers Anonymous</h3>
</div>
<p class="lead">
There are tons of amazing people at Princeton, but we often
miss each other due to differences in class years,
residential colleges, eating situations, and interests.
Tigers Anonymous was built to break down social barriers
and help you meet more awesome people.
</p>
<div class="header" id="whoweare">
<h3>Who We Are</h3>
</div>
<p class="lead">
Tigers Anonymous was built by Origin Black, a team dedicated
to improving life here at Princeton. Hit us up at <a
href="mailto:originblack609@gmail.com">originblack609@gmail.com</a>
&#8212; wed love to hear from you!
</p>
<div class="header" id="offcampus">
<h3>Connecting Off-Campus</h3>
</div>
<p class="lead">
Don't worry, you can still join the fun. Set up a Secure
Remote Access (SRA) connection using these <a
href="http://helpdesk.princeton.edu/kb/display.plx?ID=6023">instructions
from OIT</a> and you'll be chatting in no time.
</p>
</div>
</div>
<div class="row">
<div class="col-md-12 text-right">
<hr>
<a href="https://mixpanel.com/f/partner"></a>

```

```
</div>
</div>
</div>
</div>
</body>
</html>
```

---

### D.3 Chatroom

The Tigers Anonymous chatroom (shown below in Figure D.3) is implemented with the code shown at the bottom of this section.

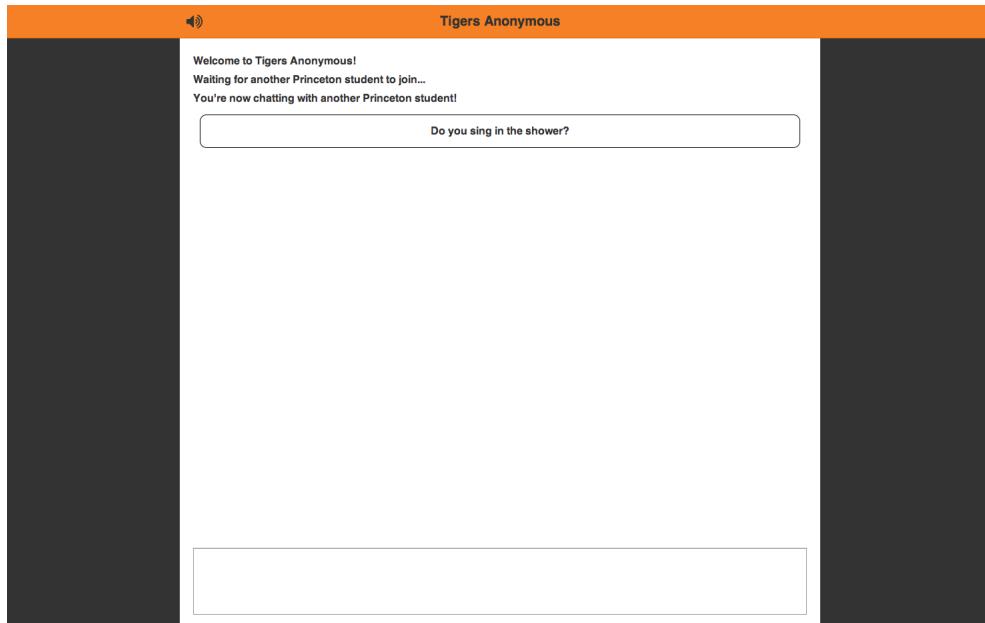


Figure D.3: Tigers Anonymous Chatroom

---

```
<!DOCTYPE html>
<html ng-app="pom">
  <head ng-controller="TitleCtrl">
    <title ng-bind="getTitle()">Tigers Anonymous</title>
    <link rel="icon" href="img/favicon.ico" type="image/x-icon">
    <meta name="viewport" content="width=device-width,
      initial-scale=1.0, user-scalable=no">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <link
```

```

        href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css"
        rel="stylesheet">
<link href="css/chat.css" rel="stylesheet" type="text/css"
      media="all">
</head>
<body ng-controller="ChatCtrl">
<div id="fb-root"></div>
<div class="nav">
  <div class="nav-container">
    <span class="brand" href="/">Tigers Anonymous</span>
    <a class="volume" ng-click="playSound = !playSound" ng-cloak>
      <i class="fa fa-volume-up" ng-show="playSound"></i>
      <i class="fa fa-volume-off" ng-show="!playSound"></i>
    </a>
    <a class="circle-down" ng-show="dropdown.shouldShowMinimized()
      && state == 'chatting'" ng-click="dropdown.show()" ng-cloak>
      <i class="fa fa-chevron-circle-down"></i>
    </a>
  </div>
</div>
<div class="chat-container">
  <div class="dropdown" ng-show="dropdown.shouldShowFull() && state
    == 'chatting'" ng-cloak>
    <div class="question">
      Do you want to find out who you've been chatting with?<br>
      <span class="promise">We'll never post to Facebook without your
        permission. Promise.</span>
    </div>
    <div class="options">
      <button type="button" class="yes-btn"
             ng-click="dropdown.accept()">Yes</button>
      <button type="button" class="hide-btn"
             ng-click="dropdown.hide()">Hide</button>
    </div>
  </div>
  <div class="chatroom" pom-scroll-glue>
    <ul ng-cloak>
      <li ng-repeat="message in messages" ng-class="message.type"
          ng-switch="message.type">
        <div ng-switch-when="chat">
          <span ng-class="{userName: !message.isPartner, partnerName:
            message.isPartner}">{{message.name}}:</span>
          <span ng-bind-html="message.text | linky |
            linkyNewlines"></span>
        </div>
        <div ng-switch-when="system">
          <div ng-switch="message.template" ng-class="{important:
            message.important}">
            <div ng-switch-when="entrance">
              Welcome to Tigers Anonymous!
            </div>
          </div>
        </div>
      </li>
    </ul>
  </div>
</div>

```

```
</div>
<div ng-switch-when="waiting">
  Waiting for another Princeton student to join...
</div>
<div ng-switch-when="matched">
  You're now chatting with another Princeton student!<br>
  <div class="question-box">
    {{message.question}}
  </div>
</div>
<div ng-switch-when="selfRevealed">
  Your partner's identity will be revealed if they also
  want to discover yours.
</div>
<div ng-switch-when="partnerRevealed">
  Congratulations! You get to find out your partner's
  identity!<br>
  You've been chatting with: <a
    href="{{message.partnerLink}}"
    target="_blank">{{message.partnerName}}</a>
</div>
<div ng-switch-when="fbError">
  Sorry, there was an error connecting to Facebook.
  Please try again.
</div>
<div ng-switch-when="fbFake">
  Sorry, it looks like you're using a fake Facebook
  account.
</div>
<div ng-switch-when="finished">
  {{partnerName}} has disconnected. Refresh the page to
  start another chat!<br>
  What do you think about Tigers Anonymous? <a
    href="https://docs.google.com/forms/d/1NI2nuAoYRZzYcawLrbWPKHsc43EdvbS5mU5d0A4cM2"
    target="_blank">Let us know!</a>
</div>
<div ng-switch-when="disconnected">
  You have been disconnected.
</div>
<div ng-switch-when="error">
  Sorry, we're unable to connect you. Please check the
  following:
<ol>
  <li>
    You need to be using a computer connected to
    Princeton's network.<br>
    If you're off-campus, <a href="#offcampus">follow
    these instructions.</a>
  </li>
  <li>You can't already be chatting with a user.</li>
</ol>
</div>
```

```
<li>You need to be using a modern web browser that
    supports WebSockets.</li>
</ol>
</div>
<div ng-switch-default>
    {{message.text}}
</div>
</div>
</div>
</li>
<li class="typing" ng-show="partnerTyping && state ==
    'chatting'">
    {{partnerName}} is typing...
</ul>
</div>
<div class="input-wrapper">
    <textarea
        tabindex="1"
        pom-focus-on-chat
        ng-disabled="state != 'chatting'"
        ng-model="message"
        ng-keydown="sendMessage($event)"
        ng-change="updateTyping()"></textarea>
</div>
</div>
<audio pom-play-on-message src="audio/notification.wav"></audio>
<script src="/socket.io.js"></script>
<script
    src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.6/angular.min.js"></script>
<script
    src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.6/angular-sanitize.js"></script>
<script
    src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.6/angular-animate.js"></script>
<!-- build:js js/app.js -->
<script src="js/app.js"></script>
<script src="js/controllers.js"></script>
<script src="js/directives.js"></script>
<script src="js/services.js"></script>
<script src="js/filters.js"></script>
<!-- endbuild -->
</body>
</html>
```

---

# Bibliography

- Peter Auer, Nicoló Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.
- Sébastien Bubeck and Nicoló Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.
- Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.