Text Mining Emergent Human Behaviors for Interactive Systems

Ethan Fast
Stanford University
ethanfast@cs.stanford.edu

Pranav Rajpurkar
Stanford University
pranavsr@cs.stanford.edu

Michael S. Bernstein
Stanford University
msb@cs.stanford.edu

Abstract
People engage with thousands of situations, activities, and objects on a daily basis. Hand-coding this knowledge into interactive systems is prohibitively labor-intensive, but fiction captures a vast number of human lives in moment to moment detail. In this paper, we bootstrap a knowledge graph of human activities by text mining a large dataset of modern fiction on the web. Our knowledge graph, Augur, describes human actions over time as conditioned by nearby locations, people, and objects. Applications can use this graph to react to human behavior in a data-driven way. We demonstrate an Augur-enhanced video game world in which non-player characters follow realistic patterns of behavior, interact with their environment and each other, and respond to the user’s behavior.

Author Keywords
information extraction, fiction, crowdsourcing, data mining

ACM Classification Keywords
H.5.2. [Information Interfaces and Presentation]: Graphical user interfaces

Introduction
Interactive systems aim to support the activities we pursue in our daily life, but the fabric of life is rich and
varied. As a result, while systems can model how people interact with their digital environments — how we use complex applications [1, 4] or write computer programs [2, 6] — few attempt to capitalize on the actions we take in our everyday lives. How do we interact with our friends over dinner? Why might we visit a hospital, a post office, or a cafe? How do we live?

Here we present Augur: a knowledge graph that models human experience by text mining more than one hundred million words of modern fiction from the web. Interactive applications can draw from Augur’s contents to react to user behavior in a data-driven way. We focus on the concrete aspects of everyday human life, like sitting in a cafe, flipping on a car’s turn signal, or exchanging greetings with a friend. Our knowledge graph captures human actions over time, as one action follows another, conditioned by nearby locations, people, and objects.

Human actions often find meaning though the conditions in which they are taken. For example, a patient who requests water in a hospital has a different set of motivations than a customer who asks for water in a restaurant. To capture these relationships, we normalize a large corpus of fiction through part of speech (POS) tagging and lemmatization, extract human actions by parsing out sequences of these tags (e.g., noun, verb, noun), and then use Wordnet’s hierarchy of synonyms to label these relations with conditional information like the location in which they occur (e.g., in a hospital, at a restaurant, on a battlefield) or the kinds of people they involve (e.g., a soldier, a doctor). Finally, we apply a crowdsourced edge-weighting function to our knowledge graph, allowing us to identify higher-precision edges.

To explore the variety of human patterns we can mine from fiction, we’ve used Augur to create video game worlds populated by data-driven people and interactions. These worlds live and breathe on their own, derived from the human activities captured in Augur’s knowledge graph. Non-player characters follow realistic sequences of actions, interact with their environment and each other, and respond to the human player’s behavior. For example, a nurse might call a patient’s name before leading that patient into a room. A waiter might take a customer’s order, then take the menu, then bring food to the customer’s table. These behaviors are represented en masse within Augur’s knowledge graph.

Related Work
Our work is inspired by general techniques for mining user behavior from data. For example, query-feature graphs show how to encode the relationships between high-level descriptions of user goals and specific underlying features of a system [4], even when these high-level descriptions are very different from an application’s domain language [1]. Researchers have applied these techniques to digitally-mediated systems such as Photoshop [1], where there is a clear distinction between the user’s description of a domain and that domain’s underlying mechanics. With Augur, we adapt these techniques to mine real-world human activities that typically occur outside of software.

Our Augur-enhanced video game worlds also extend work in data-driven game design and scene generation. Researchers have studied how users respond to interactive characters [7], crowdsourced the generation of video game narratives and mechanics [8], and synthesized new virtual scenes given example data [3]. Though we focus our applications in video games because they solve the activity sensing and representation problems, Augur provides a broader model of human behavior.
Augur API and game worlds

Augur is a knowledge graph that captures people, actions, objects, locations, and the relationships between them. It derives its contents from a large database of modern fiction. To demonstrate the capabilities of this knowledge graph, we use it to generate video game worlds populated by data-driven people and activities. Here we present a hospital world as an example. We develop this world iteratively through four layers of the Augur API, enabling higher degrees of interactive complexity.

We generate these video game worlds programmatically. We've created a demonstration sprite game engine based on Phaser (http://phaser.io/) that can place characters on a scene tilemap and direct their actions. Two game artists that we hired on the oDesk crowdsourcing market (http://odesk.com) have created character sprites and animations for people and actions in our demonstration worlds. A developer can re-randomize the game world with new characters and actions for a given location, or hand-edit the generated script if she wishes to ensure particular events happen.

Frozen Worlds

Frozen worlds show how Augur’s knowledge graph can populate a world with relevant people and activities. In these worlds, characters are static. Each person is doing something, but only that one thing – forever. Frozen worlds provide the most basic sense of human context. They answer the question: what kinds of people are in what kinds of places? And what are these people doing?

The types and quantities of the characters in this world aren’t accidental. For example, the frozen hospital (Fig. 1) contains more patients than doctors, and more doctors than children. We draw these populations from a distribution Augur has learned via text mining and crowdsourcing. We select actions for these characters in a similar way: the more common an action is, the more likely it is to be chosen.

The most basic layer of the Augur API reports the types of people who appear in a location:

GET /people/hospital

```
[{
  person: 'patient', weight: 40,
  person: 'nurse', weight: 12.5,
  person: 'doctor', weight: 7.5,
  person: 'child', weight: 7.5,
  person: 'officer', weight: 5,
  person: 'baby', weight: 5,
  person: 'receptionist', weight: 5
}```

Given these weights, we can see that patients are five time more common than doctors.

Networked Worlds

Networked worlds add between-character interactions to frozen worlds. Each person is still taking only one action, but now that action may be engaged with another character in the world. For example, in our frozen world an officer might have been “pointing her gun” abstractly. Now, she’s capable of pointing that gun at someone else, some criminal or convict or investment banker. Through interactive worlds, Augur’s knowledge graph answers a new question: how do different kinds of people interact with one another?

Networked worlds (e.g., Fig 2) show that useful context is often embedded in character roles and occupations. Nurses and doctors treat patients. A patient probably isn’t giving orders in a hospital. Using this information allows Augur to make more accurate situational predictions. For example, we can query the Augur API for five doctor actions that involve patients:

GET /interact/doctor/patient/5

```
 Gaston

Figure 1: The frozen hospital contains two doctors, two nurses, four patients, a baby, and a child. Dr. Balram pulls off a cloth while Dr. Aiko unwraps a bandage. The patients Nai and Keir are just arriving, while Aiden and Clay are already lying in bed. Nurse Mae is opening a door, while her coworker Cat is calling a patient’s name. The baby Julie is missing her mommy, and the child Severian is throwing a tantrum.

Figure 2: The Networked Hospital. After Keir arrives at the hospital, the nurse Cat calls his name. Then Dr. Aiko pulls off the cloth that’s binding Keir’s injury. Meanwhile, Dr. Balram is unwrapping a bandage on Aiden as he lies in his hospital bed.
Dynamic Worlds
Dynamic worlds add action transitions to each scene. In these worlds, characters are no longer static, but rather participate in a series of activities. In a frozen world, a king might have been “sitting on his throne.” Now, that king might be sitting on his throne and then receiving a supplicant and then ordering an execution.

These dynamic worlds answer the question: what higher-level patterns can we find among sequences of lower-level actions? Augur captures not just people and actions, but entire chains of behavior. We present one dynamic world in Figure 3. By focusing Augur on contextual elements like a person’s role and location, we can discover higher-level sequences of low-level actions.

Interactive Worlds
Interactive worlds allow our generated characters to react to actions performed by a user. They allow a user to arrive at a bustling scene, take an open-ended action, and see a character react appropriately. Before, a doctor might have looped through a series of actions that relate to serving another patient (e.g., giving a pill, making a diagnosis). In an interactive world, a human player might interrupt her by saying “I’m thirsty,” and she’ll respond by finding the patient a glass of water. For example, we can query the API to find five ways a character might respond to the words, “I love you.”

Augur Knowledge Graph
To build the Augur knowledge graph, we index more than one hundred million words of fiction writing from the Wattpad writing community (http://wattpad.com). This dataset includes stories from a variety of genres, including romance, science fiction, mystery, and urban fantasy. Most of these stories are set in the modern world.

Modeling
We take a condition-driven approach to mining human behavior, so patterns are associated with a condition around which they occur (e.g., when a person is at home, he or she might watch television). These condition-pattern tuples form the edges in our graph.

We detect each condition-pattern edge using two stream processing functions: a condition-finding function that takes a stream of input words and generates a new stream of words annotated with the presence or absence of the given context (e.g. words around a hospital); and a pattern-filtering function that takes a stream of words and captures patterns that represent human actions.

We define condition-finding as a sliding window around sequences of fifty words. WordNet synsets are groups of cognitive synonyms that each describe a distinct concept [9]. When we see a given WordNet synset within our sliding window (e.g., the synsets “building” or “geological location”), then we annotate words within the window with an associated context, like a hospital, school, or battlefield. For pattern-filtering we define a small stream-parsing component with two tokens of lookahead, based on POS tags and WordNet synsets.
Figure 3: A dynamic hospital world. Dynamic worlds allow characters to participate amongst themselves in a sequence of activities. In this scene, a nurse takes instructions from a doctor, a patient receives treatment before leaving the hospital, a child is throwing a tantrum, and a baby is missing her mother.

**Condition-Pattern Edges**

The Augur knowledge graph defines four kinds of patterns and six kinds of condition-pattern edges. These conditions provide hooks that allow Augur to make predictions on environmental priors (for example, we’re likely to be using knives differently on a battlefield and in a kitchen). We extract these patterns from the source text using simple parsers, filters, and the WordNet synset database.

Patterns types: A **Location** is any noun that contains the synsets “building” or “geographical location” among its hypernyms (e.g., house). A **Human-Usable Object** is any noun that contains “instrumentality” among its hypernyms, (e.g., car). A **Person** is any noun that contains the synset “person” among its hypernyms (e.g., doctor). An **Action** is an optional person followed by a verb with the synset “interact” or “act” among its hypernyms (e.g., “Teenager rolls eyes.”)

We combine tuples of these pattern types to extract the different Condition-Pattern edges for Augur’s knowledge graph (e.g., a location-person edge extracts the most
common kinds of people at a given location). The counts for these edges reflect the number of times we observe a pattern within fifty words of a condition. To produce the most relevant edges for a given query, we filter by a frequency count of 2 on all possible edges, then sort the remaining edges by pointwise mutual information (PMI).

Crowdsourced Edge-Weighting
We introduce a final stage to Augur’s pipeline in order to select a high-precision subset of the knowledge graph via paid crowdsourcing. The goals of the crowdsourcing pipeline are to filter out incorrect extractions and to better weight Augur’s graph edges. Unfortunately, people are poor at estimating probabilities, and fall prey to many biases and heuristics (e.g., [10]). To help reduce these biases, the crowdsourcing pipeline focuses not on probabilities but on frequencies [5].

To discover meaningful relations, the Augur pipeline asks three crowdworkers from Amazon Mechanical Turk to answer each question of the general form: “Suppose event A occurred 100 times. For each of the following events B, how many times would you expect B to follow A? Your answers must sum to 100.”

For each possible answer, a crowdworker inputs a number between 0 and 100, which represents the number of people who might be taking that a particular action. All these numbers must sum to 100. We include the the answers “some other action would occur” (to capture missing actions) and “this event is impossible” (to capture inconsistent premises) with every question.

Conclusion
Augur demonstrates that we can mine fiction to build powerful knowledge graphs. By analyzing a large corpus of modern fiction on the web, our system is able to populate itself with a wide variety of scenarios that designers and programmers might never have hand-authored. More broadly, our work suggests a future where applications can use such associative knowledge to help us navigate everyday life.

References