WeQuest: Scalable Alternate Reality Games Through End-User Content Authoring

Andrew Macvean[†], Sanjeet Hajarnis[‡], Brandon Headrick[‡], Aziel Ferguson[‡], Chinmay Barve[‡], Devika Karnik[‡], and Mark O. Riedl[‡]
[†] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, Scotland, UK
apm8@hw.ac.uk
[‡] School of Interactive Computing, Georgia Institute of Technology, Atlanta, Georgia USA
{sanjeet, brandonheadrick, aziel.ferguson, cbarve, karnik, riedl}@gatech.edu

ABSTRACT

Alternate Reality Games (ARGs) are interactive narrative experiences that engage the player by layering a fictional world over the real world. Mobile ARG stories are often geo-specific, requiring players to visit specific locations in the world. Consequently, mobile ARGs are played infrequently and only by those who live within proximity of the locations that the stories reference. In this paper, we describe an ARG platform, WeQuest, that addresses the geo-specificity limitation through end-user content generation. An authoring tool allows end-users to create new ARG stories that can be executed automatically on geo-location aware mobile devices, leading to greater numbers of available stories to be played. An intelligent process called *location translation* makes geo-specific ARGs playable anywhere in the world.

Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: General—*Games*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*

General Terms

Design, Human Factors

Keywords

Alternate Reality Games, End-User Content Generation, Authoring Tools, Location Translation

1. INTRODUCTION

Alternate Reality Games (ARGs) have recently emerged as a new genre of games. ARGs are interactive narrative experiences that engage the player by layering a fictional world over the real world; as players act in the real world their actions influence the state of the fictional world. With the advent of geo-location aware mobile devices, ARGs make use of the actual, physical world as the environment for which the game plays out. By bringing the player out into the real world, the player can also benefit from the type of rich social experiences and physical activity not possible playing a console game [8]. ARGs to date require a significant amount of human effort to run. Games must be authored—a game instance tells a story, which may be a linear sequence of events or may contain branching user-decision points. A *Game Master* runs the game and monitors players from remote in order to make adjustments to the narrative arc or trigger branching points as necessary. Many ARGs additionally utilize confederate actors planted throughout the physical world to interact with players in real time.

The ARG genre is limited in two significant ways, preventing it from becoming a mainstream form of entertainment. First, supporting an ARG is effort-intensive on the part of human Game Masters and confederates. Second, ARG stories can be *geo-specific*—they can reference real world geographical locations and landmarks requiring visits to these places to advance the narrative. Consequently, a particular ARG story is fixed to a specific region of the real world; a story set in New York City cannot be played in London without substantial re-authoring. Taken together, the scalability limitations result in a situation where ARGs are played infrequently and can only be played by those who live within proximity of the region in which the game story is set.

In this paper, we present the WeQuest platform, which has been designed to overcome the above limitations through *end-user content authoring* and automated execution of ARG stories. End-user content generation is an increasingly common means of increasing the amount of game content available to players. Thus, highly motivated players can use an authoring tool to create and share ARG stories that are set in places they want to play in. Automation of the story execution process eliminates the need for a human Game Master and replaces confederate actors with Non-Player Characters (NPCs) that appear on the mobile device screen.

While end-user authoring of new ARG stories has the potential to dramatically increase access to playable content, it is only part of the solution; the geo-specificity of newly authored ARG stories inherently limits sharing with other players in other regions. Thus, to help scale up sharing of end-user authored content, WeQuest uses an artificial intelligence process called *location translation* to break the geo-specificity constraints of ARG stories and make it possible to play stories anywhere in the world.

Short presentation, ACE'2011 - Lisbon, Portugal

2. BACKGROUND AND RELATED WORK

As mobile geo-location aware devices improve, people have become interested in playing *pervasive games*—games that blur the line between the virtual world of the game and the real world of the player in order to bring a more immersive and entertaining game experience. With mobile geo-location aware devices it is possible to create a wide variety of game playing experiences—both collaborative and competitive [1, 2, 3]. However, unlike other forms of pervasive games that rely on more traditional gameplay mechanisms, ARGs are distinguished by their emphasis on story as the primary enjoyment mechanism, with social aspects enhancing the primary mechanic.

Alternate Reality Games are interactive, fictional narrative experiences that unfold the real world. The first ARGs were played on the Internet; but there has been a recent push to move ARGs into real physical spaces. Current ARGs are about blurring and breaking through the boundaries of traditional games by expanding along the social and spatial axes, allowing players to play unexpected games in unexpected scenarios at multiple locations simultaneously [9]. Social expansion offers opportunities of forming collaborative communities where players can share content in the form of games. Spatial expansion allows games to be played out in the real world and away from the console.

There are numerous examples of commercial ARGs that utilize mobile technology but are constrained to a specific time and place. The fact that most people are not in the right place at the right time to participate in an ARG suggests the need to overcome scalability limitations inherent to ARGs. How can one reduce the need for human confederate actors and game master? The use of virtual character agents can reduce the need for human confederate actors. Lim and Aylett [6] and Stock et al. [12] implemented virtual tour guides for a museum that can adapt their presentations of existing exhibits. However, such systems do not overcome geo-specificity limitations; agents can only perform in the vicinity of fixed landmarks.

Efforts are under way to automate game mastering as well. The Spyfeet ARG [10] uses a rule-base implemented in the popular interactive fiction programming language, Inform7™, to control interactions with virtual characters. The Spyfeet story does not reference specific geographical locations and instead requires certain activities such as finding an NPC that has been mapped to an arbitrary geo-location. The Backseat Playground [4] is a mobile ARG system that triggers story elements based on features of the local environment as one rides in the back seat of a car. Backseat Playground story content also does not make specific reference to location or landmark. WeQuest also automates the Game Master and confederate actor roles, although with a story representation that is easier for end-users to author than rules. Further, WeQuest allows ARG stories to reference *specific* geographical landmarks and uses location translation to make the game playable to people in other areas.

End-user content generation has the potential to increase the number of games and the diversity of places that the games have been authored for; it has been a prominent way of expanding content of desktop computer games. To that end, authoring tools can be devised to allow motivated players to create new games for the places that they live. Authoring tools can be provided for ARG platforms as well, as long as the story representation is easy to understand and the tools are easy to use. For example, authoring tools have been developed for audio tour guides (cf., [13]); such techniques could be extended to ARGs. Our platform, WeQuest, provides an end-user authoring tool as the first step toward addressing the scalability of the ARG genre. However, without *location translation*, end-user authored ARG stories will only be accessible to those who live in the vicinity of the place that the story is set. To our knowledge, WeQuest is the first system to address geo-specificity limitations of the ARG genre through the combination of end-user authoring and location translation.

3. THE WEQUEST SYSTEM

Story-based games do not have a high replay value, motivating the need for end-user authoring and location translation as means of providing greater amounts of unique content and making that content more accessible to a greater number of people. WeQuest is a platform designed to scale up accessibility of ARGs through the use of three components:

- A game engine that runs on a geo-location aware mobile device and can download and execute single- and multi-player ARG stories.
- An authoring tool that supports end-user authoring of new geo-specific stories.
- A location translation process that adapts ARG stories to new areas, allowing them to be played anywhere.

The remainder of this section overviews the story representation used by WeQuest and game engine execution loop.

In WeQuest, ARG stories are represented by a dependency graph, a directed, acyclic graph (DAG) where the nodes correspond to story events and arcs impose constraints on story event visitation order. Inspired by classic Role-Playing Games (RPGs)—a metaphor that works well with ARGs [10]—story events involve engaging in dialogue with virtual Non-Player Characters (NPCs) and using or acquiring virtual inventory items. Story event nodes reference specific GPS coordinates that a player is required to be within a certain radius of for the interaction to occur. Arcs between nodes represent dependencies that must be fulfilled for a particular event to fire. A dependency graph is a basic technique for managing lock-and-key style game play; for an event to occur, it must be "unlocked" by completing all other events it depends upon. Unlike finite state machines, dependency graphs can support branching stories (e.g., choose-your-own-adventure), partial ordering of events, and and parallel multiplayer events. The dependency graph defines the logical progression of the story and NPC actions, enabling the game engine to perform the roles of Game Master and confederate actors.

A game dependency graph is represented in XML format. Figure 1 shows several event nodes representing contact from an NPC, two options in a dialogue tree, and an inventory item. The XML references images and movies on a server, which are only downloaded when needed. As described in Section 4, ARG story authors never need to edit XML or manage image and movies on the server; these tasks are managed through an authoring tool interface.

Figure 2 shows the game engine in action, presenting an NPC interaction to the player. Players communicate with NPCs through *dialogue trees*, branching structures where

```
<node id="27" location="33.777455,-84.390096" dependencies="">
  <subnode type="dialogue" image="27.jpg">
    <line speaker="Detective">Welcome...</line>
  </subnode>
</node>
<node id="28" location="" dependencies="27">
  <subnode type="option" id="29">I accept</subnode>
  <subnode type="option" id="30">No thanks</subnode>
</node>
...
<node id="41" location="33.777203,-84.39774" dependencies="29,36">
  <subnode type="inventory" image="41.jpg">
    <inventory type="global">Receipt</inventory>
  </subnode>
</node>
```

Figure 1: A fragment of dependency graph XML.

Figure 2: The game engine showing an NPC interaction.

players are presented with dialogue options to select from and the NPC responds accordingly. Dialogue trees can be implemented with a dependency graph (see Figure 3).

Because the logic of story progression is encoded into the dependency graph structure, the Game Engine execution loop is simple. Once a game instance dependency graph has been downloaded, the Game Engine periodically searches through the list of nodes to see if it there are any nodes in which the following conditions hold: (a) the mobile device is within a certain radius of the node, and (b) all dependencies of the node are unlocked. When these conditions hold, the Game Engine follows the instructions in the node for how to interact with the user: show a picture of an NPC, show a movie, play an audio file, display a line of dialogue from an NPC, provide a set of dialogue options for the player, or present a virtual inventory item.

The Game Engine supports both single and multi-player gameplay. Dependencies are synchronized between players of the same game so that team members can unlock different parts of the story independently of each other, creating opportunities for cooperative problem-solving and also competition. The dependency graph story representation can be used to implement classical ARG mystery and conspiracy stories, role-playing adventure games, tours, and races between teams of players. To support different types of games, parameters can disable checkpoint flags and other players' location on the game engine's map screen. In general, we believe our dependency graph representation to be flexible enough to afford end-user authors a high degree of creativity when it comes to what and how ARGs should play out.

4. AUTHORING TOOL

Key to ARG scalability is the idea that the end-users are capable of authoring their own games, rather than being reliant on a few expert game designers. To lower barriers to ARG story creation with the WeQuest platform, we provide an authoring tool that hides the complexities of authoring an XML dependency graph. The WeQuest authoring tool can be accessed through a Javascript™ enabled web browser (including tablet PCs which can be used to author in-situ). To facilitate authoring of geo-location material, the authoring tool is integrated with Google Maps™. As shown in Figure 3, authors can directly manipulate the dependency graph and see the events superimposed on a map.

The dependency graph is portrayed to the user as location nodes, specifying places where things can happen in the game, with embedded event nodes. Location nodes are not part of the dependency graph representation—they are a visualization convenience to the author to break locations out as a separate concept from story event nodes. Because of the modular nature of ARGs [7], we believe that it is easier for authors to think about locations and the events that happen at those locations as separate concepts. The visual representation is automatically converted to the dependency graph representation, which stores the GPS coordinates of locations in the story nodes.

New locations can be created by querying the map interface, which utilizes the Google Maps™ API. Event nodes specify what happens to the players when they arrive at a particular place; authors can specify the instructions by selecting and parameterizing different types of event nodes. Figure 3 shows part of a story for a multiplayer game. The first few events create a dialogue tree with interleaved NPC dialogue nodes and player response options. Later events can occur in un-specified order, or in parallel if multiple players spilt up.

Although game players generally have a good idea of what games they enjoy, and there are objective measures of ARG story quality (cf., [7]), it is not necessarily the case that ARG game players make effective game authors. To understand the effectiveness of end-user content authoring for ARGs, two questions must be answered. First, are end-users of an ARG platform interested and likely to generate new game content? Second, how well does the WeQuest authoring tool support amateur end-user authoring of new ARG stories? We ran a small-scale preliminary study to determine the extent to which amateurs were able to author their own games and how they perceive the process.

We recruited 9 participants to create ARG stories from scratch using the WeQuest authoring tool. Of the 9 participants, 1 participant had never played nor had previously authored computer game content, 3 were "infrequent" or casual gamers, 3 described themselves as regular gamers who had never created their own games, and 2 were regular gamers who had also previously created their own desktop video games. All participants were computer scientists at Georgia Tech and Heriot-Watt universities.

Participants were given a tutorial story to author and then asked to use the WeQuest authoring tool to create their own ARG adventure. All participants were provided with a common set of constraints so that participants' stories could be compared: participants were instructed to use at least 5 locations, to use all of the story event types (NPC dialogues, player options, and inventory items), and to author at least one branch in the story. Aside from meeting the constraints of the study, participants were free to explore the tool at their leisure and utilize the available functionality as they

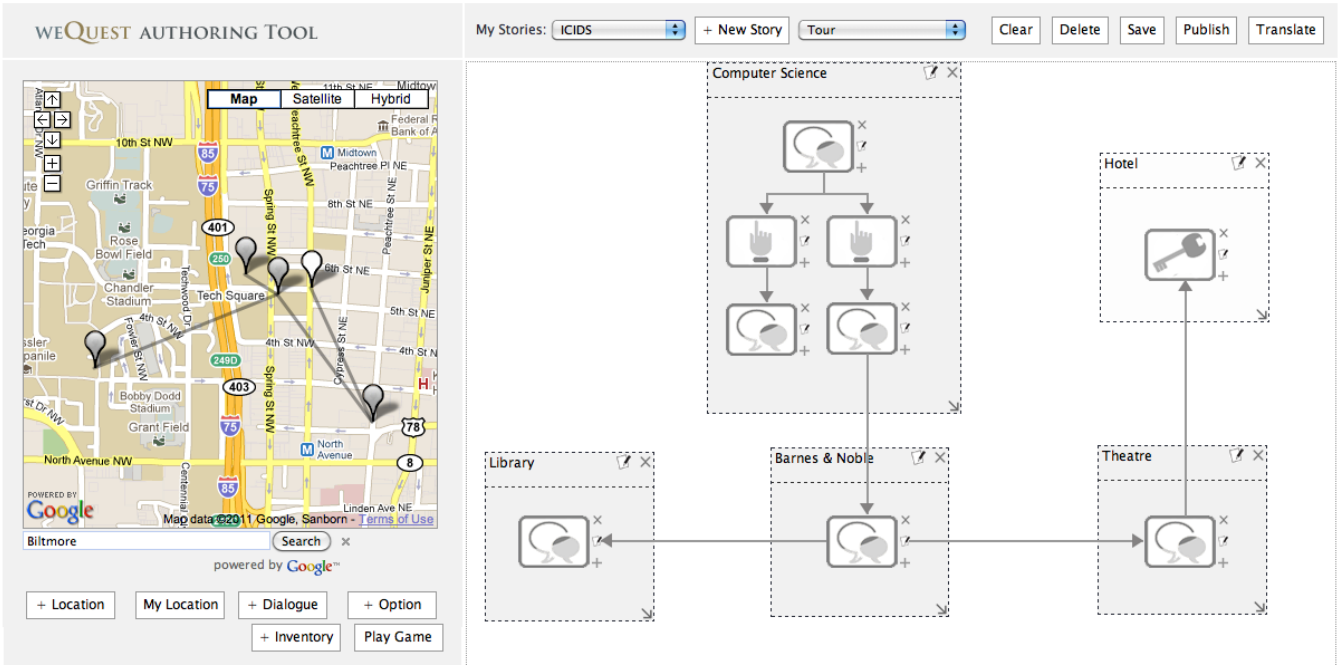
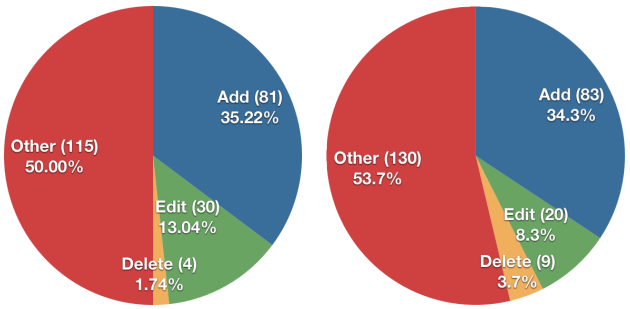


Figure 3: The authoring tool with a simple branching story involving character dialogues at five locations.



(a) The non-gamer author actions.

(b) The experienced gamer author actions.

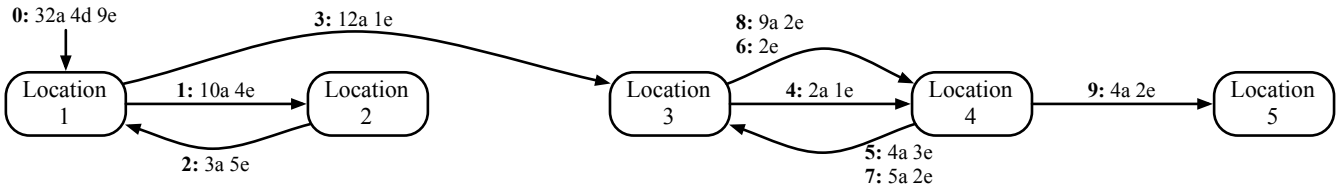
Figure 4: Case study action breakdown.

saw fit. On completion of the authoring, participants were provided with a questionnaire asking them to assess the authoring process. The WeQuest authoring tool logged all actions the participants took, allowing us to reconstruct a picture of the authoring processes used by each participant.

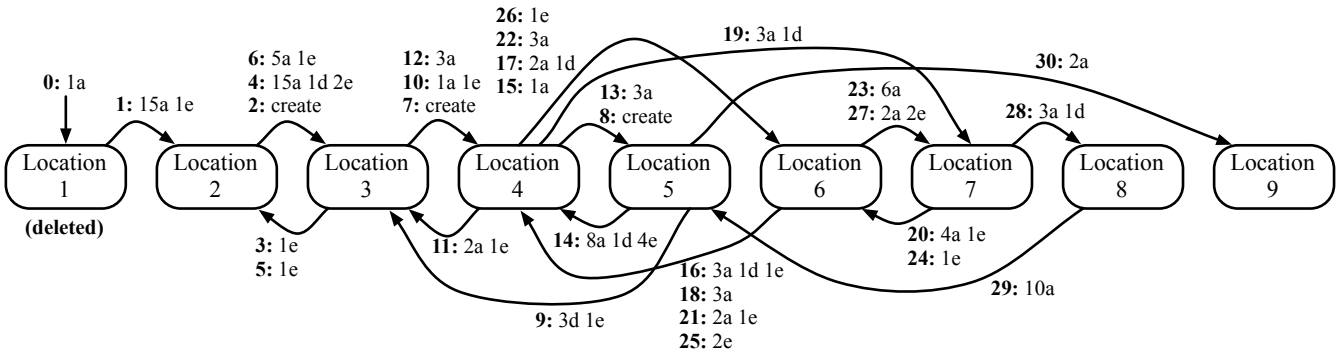
To author a new ARG story from scratch with the above constraints, participants took between 43 minutes and 1.5 hours. Using Likert scales, marked from 1 to 5, we asked the participants to rate how much they enjoyed authoring their own ARG using WeQuest, whether they would like to use the authoring tool again, and whether they would like to play through a user authored game. The mean response for enjoyment of the authoring process was 4.3 ($sd = 0.71$), the mean desire to author again was 4.2 ($sd = 0.83$), and the mean desire to play a WeQuest user authored game was 4.1 ($sd = 0.93$). Despite the varying background of our participants, it was positive to note the consistently high marks and low deviation within our results.

We present two case studies that illustrate how people use the authoring tool to create ARG stories. From the nine participants in our study, we selected one case to represent the authoring patterns of (a) non-gamers, and the other case to represent (b) experienced gamers who have designed and built games before. These cases represent the extreme ends of the expertise spectrum in our data. For each case study, we present a pie chart showing the action breakdown (Figure 4). In the charts, actions are characterized as follows: (a) *adding* new elements such as locations, NPCS dialogue entries, and inventory items; (b) *editing* existing elements to make changes; (c) *deleting* existing elements; and (d) *other* actions that do not affect game content such as saving the game or manipulating the interface. As can be seen from Figure 4, both cases have very similar usage patterns despite having very different backgrounds. The large percentage of "other" actions suggests there may be user interface inefficiencies. In particular, we note that screen real-estate can be quickly consumed, requiring frequent node re-arrangement in the interface.

When analyzing the traces of authoring actions, we see two very different authoring styles. Due to the modular nature of ARG games [7], stories are constructed from a number of 'location' nodes where the main parts of the narrative take place. We can therefore analyze the order in which authors create and populate these location nodes to get a picture of their approach to authoring. Figure 5 show the patterns adopted by our two case study participants. The boxes in the figure represent the different locations authored and the arcs represent shifts of attention. Arcs are numbered in the order the attention shifts occurred and are annotated with the number of added story events, deleted story events, and edits to content that occurred once attention shifted to that location. We can therefore construct a picture of how linearly/iteratively our case-study authors



(a) The non-gamer pattern.



(b) The expert gamer pattern.

Figure 5: Contrasting authoring patterns adopted by our two case study participants. Arcs denote attention shifts annotated with adds, deletes, and edits that occurred while attention was focused on each node.

created their game content and the order in which they created locations and story nodes.

As illustrated by Figure 5(a), our inexperienced author chose to work relatively linearly from the first location to the last, with very little revisitation of perviously created locations. This participant created a new location, populated it with content, and then moved onto the next location to appear in their game. In contrast, our experienced author (Figure 5(b)) took a far more iterative approach, following strands of the narrative to other locations before revisiting previously made locations in order to progress other paths through their story. While Figure 4, show very similar overall usage, we see from Figure 5 the tool supports very different authoring styles, showing a flexibility which allows people of all backgrounds to create games. We note that our other participants fell between these two polar authoring styles, anecdotally we observed that the degree of iterative authoring is relative to the experience level and background of the participant.

Having identified that the tool affords the flexibility for differing authoring approaches, future work will involve a larger scale study to test hypotheses about the relationship between non-linear authoring and expertise , to determine whether there are categories of authoring patterns that can be learned, and to analyze end-user authored stories with respect to existing ARG enjoyment metrics [7] in order to establish design principles in support of end-user authoring of enjoyable ARGs.

5. LOCATION TRANSLATION

While end-user authoring has the potential to increase the amount of content available for players, location translation seeks to make all content accessible to all players regardless of geographical contstraints. Location translation maps locations in a game story to analogous locations in a new city where the user intends to play.

To formalize the problem, consider an original story set in one area as a number of locations L derived from a dependency graph. For each location L_i in the original story, there can be n_i analogous candidate locations in the vicinity of the target area, denoted $M_{i,j}$ for $j = 1...n_i$. The goal of the translation agent is to select one location $M_{i,j}$ for each L_i such that: (a) the analogical similarity between any locations in the original and translated graphs is maximized, irrespective of geography, and (b) the difference in distances between adjacent locations in original story and translated story is minimized when geography is considered.

5.1 Translation Search Algorithm

Our location translation process searches for the optimal candidate $M_{i,j}$ for each location L_i, given a dependency graph. Viewing game instance translation as an optimization problem, we observe that the *optimal substructure* property holds—the optimal solution to a problem can be obtained through the combination of optimal solutions to its subproblems. That is, the optimal choice of candidate for a particular location L_i is a function of the optimal choice of candidates for locations immediately adjacent to it in the dependency graph. We solve the location translation problem with *dynamic programming* (DP), an optimization algorithm specifically designed to exploit the optimal substructure property through an inductive process that runs in $O(n_{\max} * |L|)$. The solutions to subproblems are cached to avoid repetitious computation. Our DP implementation determines the suitability of any given candidate $M_{i,j}$ for original location L_i by computing the cost of $M_{i,j}$ given the optimal solutions for locations prior to $M_{i,j}$ in the dependency graph. Because dependency graphs can branch arbitrarily, we extend dynamic programming to account for multiple subproblems. Our location translation algorithm is

Input: A list of locations L, origin and target cities $city_1, city_2$, and a set of similarity matricies.
Output: A list of locations S that is analogous to those in L.
let $S \leftarrow C \leftarrow M \leftarrow \emptyset$
for i = *1 to number of locations* $|L|$, *consistent with the dependency graph* **do**
  let $M_i \leftarrow$ `candidates`(L_i, $city_1$, $city_2$, `type`(L_i))
  for j = *1 to number of candidates* $|M_i|$ **do**
    let $cost \leftarrow 0$
    for l = *1 to number of parents of* $M_{i,j}$ *in dependeny graph* **do**
      $cost \leftarrow cost$ + length difference of edge between $M_{i,j}$ and S_l
    $cost \leftarrow cost + (k/$`sim`$(L_i, M_{i,j}, city_1, city_2,$ `type`$(L_i)))$
    if $cost < C_i$ **then**
      $S_i \leftarrow M_{i,j}$;
      $C_i \leftarrow cost$;

Figure 6: Modified dynamic programming for location translation.

shown in Figure 6.

A cost function evaluates a candidate location $M_{i,j}$ based on similarity of $M_{i,j}$ to the original location L_i plus the difference in distances between the candidate and its dependency graph predecessor as compared to the original dependency graph when locations are positioned geographically. Specifically, the cost of candidate $M_{i,j}$ is computed as: $\sum_{d \in Dep(M_{i,j})} \left(\left| length(edge_{d,j}) - length(edge_{\text{orig}}) \right| \right) + \frac{k}{sim}$ where $Dep(M_{i,j})$ returns the nodes that candidate $M_{i,j}$ are dependent on according to the dependency graph, $edge_{d,j}$ is a edge in the new graph between the current candidate location and the candidate selected as the solution to a subproblem, $edge_{\text{orig}}$ is the corresponding edge in the original dependency graph, and sim is the probability ([0..1]) that two locations in two different cities are similar. Thus, as similarity decreases, cost increases exponentially. The constant k penalizes dissimilarity relative to edge difference and can be tuned manually to shift emphasis between similarity and geographical distance. Our DP implementation selects candidates in the target city that minimize cost.

5.2 Location Similarity

How do we compute the analogical similarity between locations in different cities? Analogical reasoning is a difficult problem typically requiring large amounts of well-formed common-sense knowledge. Since there is no readily-available knowledge-base of semantic features of locations, shops, and landmarks,[1] we use an alternative approach to finding analogies that finds statistical correlations based on information about locations retrieved from the World Wide Web. Websites such as Citysearch™ and Yelp™ allow their users to write reviews of restaurants, shops, and other landmarks. We make the assumption that the words people use to describe their experiences at these locations captures some latent (e.g., hidden) semantic information [5]. That is, some words in the user reviews of places capture salient features of the place and algorithms can analyze and compare word usage to derive similarity between places by inferring, with some probability, the existence of common, salient features. When this assumption holds, term-frequency vector similarity techniques can be used to compute the distance between texts. Our approach to identifying similar locations based on reviews is inspired by the phrase-similarity computation technique of Sahami and Heilman [11], which compares term-frequencies vectors between documents retrieved from Google™. Our technique, however, uses web-retrieved reviews as a document corpus instead of the entire Web Wide Web, and compares locations instead of phrases.

Location translation begins with a pre-processing phase in which a *similarity matrix* is built that captures the probability that locations in disparate cities are analogous. We further specialize the similarity matrices by *type* of location (e.g., restaurant, park, salon, etc.). Thus, each similarity matrix represents a combination of $City \times City \times Type$. For each location of each type in each city, we download all reviews from Citysearch™ through their API. Reviews are merged into a single document representing the location and stop-words, words that are not nouns (according to Wordnet), and common proper nouns (such as the names of credit card companies) are removed. Removing non-noun words from reviews is to avoid relating two places based on similar sentiment. While sentiment analysis is useful for product recommendation, we require an objective account; noun-only similarity is a simple form of feature-only comparison under the assumption that nouns identify salient features of a place.

Review documents are converted into term frequency vectors where each dimension in the vector is a term and the value for each term is computed by Term-Frequency Inverse-Document Frequency (TFIDF), a common measure of term importance based on term uniqueness across documents. The Cosine similarity measure is used to determine similarity of document vectors by measuring the angle between each pair of vectors. Applied to all pairs of locations from two cities, the result is a similarity matrix with columns representing places in one city, rows representing places in the other city, and cells containing the probability that the two places are the same. Figure 7 illustrates the process of computing a similarity matrix for two cities. Repeating this process for all pairs of cities and all types of locations produces $|City \times City \times Type|$ similarity matrices. This process must only be done occasionally to incorporate new locations and new reviews. Given an origin city, target city, and location type, the candidates for a story location L_i are generated by extracting an entire column or row of the appropriate similarity matrix.

5.3 Results and Evaluation

In practice, we find ARGs are subjective experiences and that players do not differentiate between game instances as long as the locations are reasonably chosen. Figure 8 shows an example of an ARG story and its translation to a different part of the same city (disallowing self-matching), which allows us to visually inspect the results. The similarity matrices were built from a relatively small selection of 185 locations of different types in the city. The original story locations are (1a) a fresh-faire pan-asian cafe, (1b) Barnes and Noble bookstore, (1c) a NY-style pizza restaurant, (1d) the Fox Theatre, and (1e) the World of Coke museum. The story required a minimum of 2.1 miles to traverse. The translated story locations are (2a) an organic foods cafe, (2b) an independent bookseller, (2c) a bar with a number of pizza selections, (2d) the Center for Puppetry Arts (which has theatre shows), and (2e) the Art Museum. The translated

[1]There are ontological knowledge-bases such as Freebase™, but these resources tend to be incomplete and/or too sparsely populated for our needs.

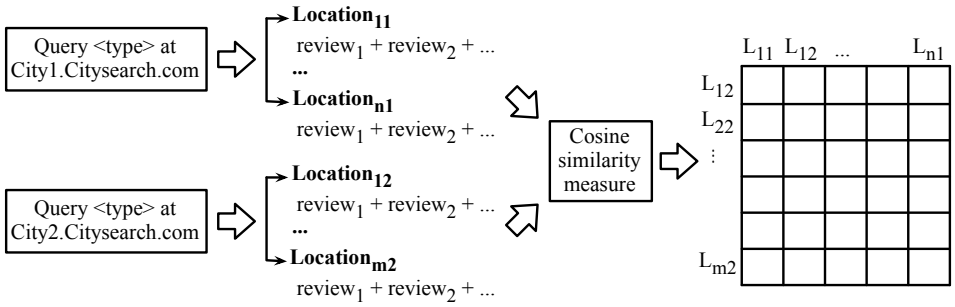


Figure 7: The similarity matrix construction process for a pair of cities and a type of location.

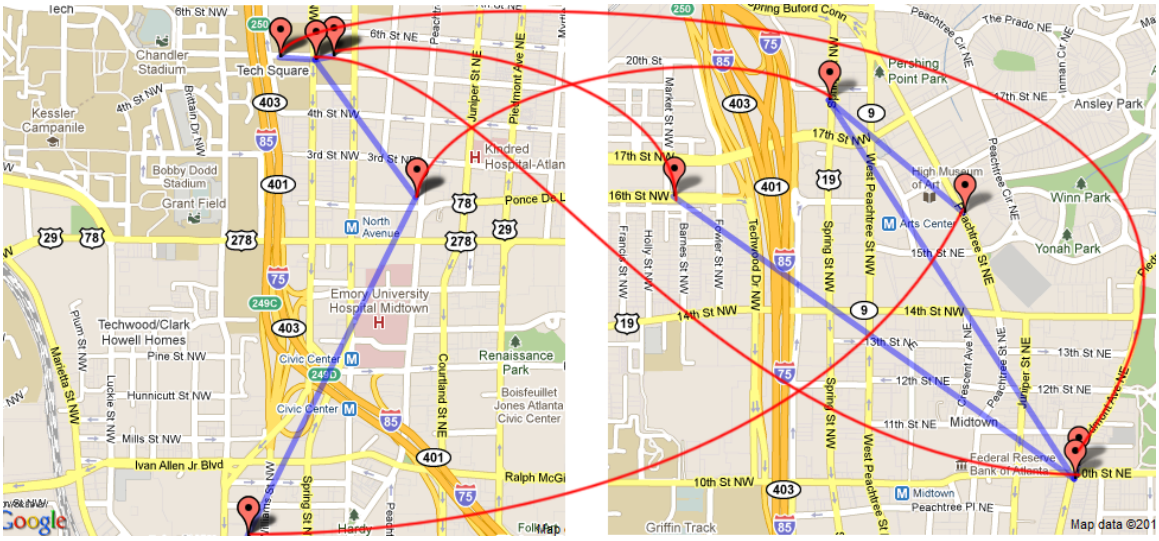


Figure 8: An ARG story translated from one part of a city to another. Blue lines are dependency arcs. Red lines show analogical matches between locations.

story required a minimum of 3.1 miles to traverse. Note that the quality of translation is dependent on the quality of reviews from Citysearch™, the coverage of the target city in terms of the number of locations that we retrieve, and the particular geometrical layout of each city.

Reliable location translation requires accuracy in the ability of our system to capture meaningful similarities between locations. To evaluate the quality of the similarity computations, we randomly sampled 10 *source* restaurants from our dataset of locations. For each source restaurant, we randomly sampled 10 *target* restaurants against which to evaluate similarity. We asked 5 participants familiar with the city to sort each list of targets based on their judgement of similarity to the source. From participant data, we computed a gold standard ranking as follows. Treating each participant's trial as a competition amongst target restaurants to be the most similar to the source restaurant, we use the ELO tournament rating method to determine a total order of target restaurants for each source restaurant. The ELO rating for a target restaurant is the aggregate number of other restaurants ordered below it by participants. We then used similarity matrix lookups to generate an ordered list of targets for each source (geography was ignored). Thus, humans and WeQuest performed the same ranking tasks.

To compare the WeQuest similarity matrix ranking against the gold standard, we used the Kendall's Tau rank correlation coefficient to assess the association between ranked lists. We calculated an average τ of 0.533 (where 1.0 indicates perfect agreement) across the 10 source restaurant comparisons. The Tau value is significant at $p = 0.0318$, indicating that that the gold standard and generated rankings tend to be highly associated. Not all source restaurants resulted in statistically significant results; we observed a spread from $\tau = 0.2$ to $\tau = 0.822$. We conclude that, on average, WeQuest-generated and gold standard rankings statistically describe the same ranking. We note that, anecdotally, human participant ranking becomes increasingly arbitrary when actual similarity between locations is low, making the gold standard ELO values for low-similarity restaurants unreliable. Thus, looking at the tops of the rankings, WeQuest's top pick concurred with the gold standard's top-pick 60% of the time, was in the top two 80% of the time, and was in the top three 100% of the time. Thus, accuracy is highest when human-rated similarity is also high, which is significant considering that the optimization search must balance maximizing similarity with minimizing distances; it doesn't always pick the most similar location. Further improvements potentially may be achieved through more informed feature selection and through alternative document similarity measures, such as Latent Semantic Analysis [5].

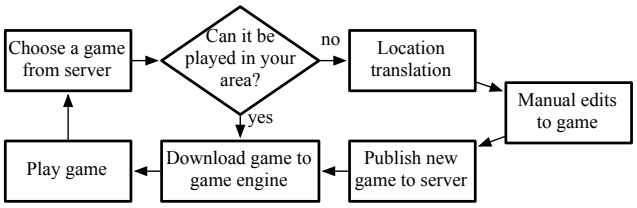


Figure 9: The flow of game play, end-user authoring, and location translation.

Results will improve with a larger corpus of greater numbers of locations and more/longer reviews per location.

5.4 Semi-Automated Translation

The optimization search process, in its attempt to balance graph size differences and analogical similarity, may choose candidates that are too far away, not analogical enough, or are in unsafe neighborhoods. We also cannot rule out the possibility of a poor analogy, especially for locations that are not well-reviewed by users. Consequently, we provide the ability for the end-user to edit the translated story before play occurs, as shown in Figure 9. We assert that editing an existing, translated story is easier than writing a new story from scratch or completely manually translating an existing story. Translated stories are pushed back to a central server, allowing other people in the same area to immediate begin using the stories. We believe this to be a key element to achieving ARG scalability.

6. CONCLUSIONS

The ability to play through stories in the real world coupled with the opportunities for socialization during game play are some of the reasons that ARGs have gained in popularity. Despite the positive qualities of ARGs, the geo-specificity of stories coupled with the human effort required to create and run games have been critical factors in preventing the ARG genre from becoming a mainstream form of entertainment. We believe that ARGs can be scaled to the point that they become a form of game that everyone can play. To achieve this, human Game Masters and confederate actors must be eliminated, greater amounts of story content must be available, and the geo-specificity of stories must be mitigated.

The WeQuest attempts to overcome scalability limitations in three ways. First, by providing a game engine that runs on geo-location aware mobile devices, the execution of ARG stories is automated. This is accomplished through the use of the dependency graph story representation, which is flexible enough to account for a wide variety of ARG experiences but precise enough to implement non-player character dialogues. Second, an authoring tool allows end-users to generate their own stories, thus providing availability of content for where ever there are individuals motivated to make their own games. We find the authoring tool to support a spectrum of authoring practices. While end-user content generation is good means of acquiring large amounts of content, the geo-specific nature of ARG stories still must be overcome to facilitate greater accessibility to existing content. Thus, location translation allows players to quickly and easily convert any ARG story into one that can be played in their locality.

7. REFERENCES

[1] L. Barkhuus, M. Chalmers, P. Tennent, M. Hall, M. Bell, S. Sherwood, and B. Brown. Picking pockets on the lawn: The development of tactics and strategies in a mobile game. In *Proceedings of the 7th International Conference on Ubiquitous Computing*, 2005.

[2] S. Benford, R. Anastasi, M. Flintham, A. Drozd, A. Crabtree, C. Greenhalgh, N. Tandavanitj, M. Adams, and J. Row-Farr. Coping with uncertainty in a location based game. *IEEE Pervasive Computing*, 2(3):34–41, 2003.

[3] S. Benford, D. Rowland, M. Flintham, A. Drozd, R. Hull, J. Reid, J. Morrison, and K. Facer. Life on the edge: supporting collaboration in location-based experiences. In *Proceedings of the 2005 Conference on Human Factors in Computing Systems*, 2005.

[4] A. Gustafsson, J. Bichard, L. Brunnberg, O. Juhlin, and M. Combetto. Believable environments: Generating interactive storytelling in vast location-based pervasive games. In *Proceedings of the 2006 ACM International Conference on Advances in Computer Entertainment*, 2006.

[5] T. Landauer and S. Dumais. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

[6] M. Y. Lim and R. Aylett. Narrative construction in a mobile tour guide. In *Proceedings of the 4th International Conference on Virtual Storytelling*, 2007.

[7] A. Macvean and M. O. Riedl. Evaluating enjoyment within alternate reality games. In *Proceedings of the 38th International Conference on Computer Graphics and Interactive Techniques*, 2011.

[8] C. Magerkurth, A. Cheok, R. Mandryk, and T. Nilsen. Pervasive games: bringing computer entertainment back to the real world. *ACM Computers in Entertainment*, 3(3), 2005.

[9] M. Montola. Exploring the edge of the magic circle: Defining pervasive games. In *Proceedings of the 2005 Digital Arts and Culture Conference*, 2005.

[10] A. Reed, B. Samuel, A. Sullivan, R. Grant, A. Grow, J. Lazaro, J. Mahal, S. Kurniawan, M. Walker, and N. Wardrip-Fruin. A step towards the future of role-playing games: The SpyFeet mobile RPG project. In *Proceedings of the 7th Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[11] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International World Wide Web Conference*, 2006.

[12] O. Stock, M. Zancanaro, P. Busetta, C. Callaway, A. Kruger, M. Kruppa, T. Kuflik, E. Not, and C. Rocchi. Adaptive, intelligent presentation of information for the museum visitor in PEACH. *User Modeling and User-Adapted Interaction*, 17(3):257–304, 2007.

[13] K. Tsuruoka and M. Arikawa. An authoring tool for urban audio tours with animated maps. In *Proceedings of the 2008 ACM International Conference on Advances in Computer Entertainment*, 2008.