Murder Mystery Generation from Open Data

Gabriella A. B. Barros[1], **Antonios Liapis**[2] **and Julian Togelius**[1]
1: Tandon School of Engineering, New York University, New York, USA
2: Institute of Digital Games, University of Malta, Msida, Malta
gabriella.barros@nyu.edu, antonios.liapis@um.edu.mt, julian@togelius.com

Abstract

This paper describes a system for generating murder mysteries for adventure games, using associations between real-world people mined from Wikipedia articles. A game is seeded with a real-world person, and the game discovers suitable suspects for the murder of a game character instantiated from that person. Moreover, the game discovers characteristics of the suspects which can act as clues for the player to narrow down her search for the killer. The possible suspects and their characteristics are collected from Wikipedia articles and their linked data, while the best combination of suspects and characteristics for a murder mystery is found via evolutionary search. The paper includes an example murder mystery generated by the system revolving around the (hypothetical) death of a contemporary celebrity.

Introduction

Computational creativity focuses on discovering artifacts in creative domains such as art, music and digital games, or even mathematics and engineering. Sometimes, the results of creative processes are created *in vitro*, without a basis in the real world. This is possible, for instance, in automated theorem discovery (Colton 2002) where a model of finite algebra suffices for proving a generated theorem. However, the real world often influences the creative process in some way, acting as a training set (Eigenfeldt and Pasquier 2012), as a seed (Krzeczkowska et al. 2010; Hoover et al. 2012), as an evaluation (Correia et al. 2013; Martins et al. 2015) or as a mapping between dissimilar modalities (Johnson and Ventura 2014; Veale 2014). There is interesting research in transforming textual data into images (Krzeczkowska et al. 2010), poems (Colton, Jacob, and Veale 2012) or even games (Cook and Colton 2014).

This paper describes a system for creating digital adventure games using open data (primarily Wikipedia articles). Starting from a designer- or player-specified real-world person, the game produces a murder mystery where said person has been killed and the player must find the killer among several suspects. The player can pinpoint the killer by eliminating suspects based on certain characteristics they do not share with the killer. Suspects and their characteristics are collected from Wikipedia articles; the suspects are associated with the victim, while the characteristics may be shared among some but not all of the suspects. From this broad range of suspects and types of characteristics (e.g. date of birth, affiliation, awards), the best combination for use in a murder mystery is discovered via evolutionary search, which ensures that the mystery is solvable while the characteristics are diverse within the chosen set of suspects.

The WikiMystery system described here is the next step from our previous work on "Data Adventures", i.e. generating adventure games from open data (Barros, Liapis, and Togelius 2015; 2016). The system presented hereby enriches the experience by adding multiple paths between a victim and possible culprits (thus allowing the player to explore the game in a non-linear fashion). Moreover, the addition of clues (which help the player eliminate suspects) and puzzles (which block progression on specific plot lines) increases the richness of interaction. More importantly, WikiMystery is unique among attempts to generate games from data in that the storyline and affordances (possibly even the difficulty in terms of solvability) are affected directly by the data.

Related Work

As a highly creative domain, storytelling has received considerable attention in computational creativity research. Such research has often focused on generating complete stories in textual form: examples include stories written around a specific theme such as betrayal in BRUTUS (Bringsjord and Ferrucci 2000), a specific caste of heroes such as the Knights of the Round Table in MINSTREL (Turner 1993), or whether the story achieves an intended goal state (Riedl and Young 2010). On the other hand, games as interactive media (Aarseth 1997) can do away with several of the requirements of full story generation and the challenges posed e.g. by the natural language processing needed for narration (Montfort and Pérez y Pérez 2008). Examples of a narrative structure transformed (computationally) to and from interactive experiences include the work of Laclaustra et al. (2014) which uses a game map and game characters to create a story from their interactions, and the work of Robertson and Young (2015) which transforms a story plan into a level structure, instantiates game characters and creates actions for the player to issue to their avatar.

Automatically transforming stories into games and vice versa is an example of computational creativity used for *transformation* or *reinterpretation* of data from one medium

to another. Other examples include the transformation of images (Johnson and Ventura 2014), game levels (Lopes, Liapis, and Yannakakis 2015), or text (Thorogood and Pasquier 2012) into soundscapes, news articles into games (Cook and Colton 2014) or collages (Krzeczkowska et al. 2010), etc. For the purpose of conciseness, we will focus on how data has been transformed into playable experiences.

The automated game creator Angelina is a prime example of a system transforming data into games: Angelina (Cook, Colton, and Pease 2012) generates platformer levels and decorates them in an audiovisual theme. The theme is derived from the text body of news articles, while the sounds, images, backgrounds are derived from parsing the text and searching online databases using these keywords. Game-o-matic (Treanor et al. 2012) uses human-authored concepts and their associations to generate consistent games and their rules, while relying on web-based images of the authored concepts for their in-game visual representation. Open data, on the other hand, have also been transformed into playable experiences: examples include the simple physics game BarChartBall (Togelius and Friberger 2013) which uses UK census information to form the terrain of a ball-rolling game, or OpenTrumps (Cardona et al. 2014) which instantiates *Top Trumps* (Winning Moves 1999) card decks with countries' statistics (from United Nations and World Development Indicator databases). Games that feature content generated from open data are often referred to as *Data Games* (Friberger et al. 2013).

Similarly to the aforementioned projects, WikiMystery and its overarching project of "Data Adventures" (Barros, Liapis, and Togelius 2015) use online data to produce playable games featuring real-world people (both alive and dead) and locations. Similar to Angelina and Game-O-Matic, they decorate the game's visuals with images collected from Wikimedia Commons. Unlike some other data games, however, the data form a core part of the gameplay and deeply affect the experience. Compared to Game-O-Matic or Angelina where the online data is used to theme an already playable game, WikiMystery relies on the data to form the plot of the game, the locations the player can visit, the non-player characters (NPCs) and their relationships (which act as clues in the mystery). This reliance on data (which, due to the crowd-sourced nature of Wikipedia and human fallibility can be erratic or incomplete) poses a grand challenge to Data Adventures. To guarantee playable adventure games relying exclusively on such data requires an almost human-like intelligence and creativity. The current steps taken by the authors with this paper and its predecessors (Barros, Liapis, and Togelius 2015; 2016) can — and often do — create absurd storylines and unintuitive associations between non-player characters. It should be noted, however, that absurdity is inherent to the grounding of data (and a desirable artifact of freedom of online speech) and hiding or curating it would remove the core strength (and evidence) of the open data-driven nature of the game. The world is absurd, and this is reflected in some of the results of a data-based game generator.

Overview of WikiMystery

WikiMystery draws inspiration from several adventure games, a broad and rich genre that has gained a resurgence of popularity over the past years. Due to the diversity of games within this genre, there is a large variety of mechanics and gameplay styles in adventure games that can also be found in other genres. In this work, we refer to adventure games as games with the following characteristics: they are story-driven, their core mechanics revolve around puzzle-solving, interaction with the game world is mostly done through object manipulation, the player controls a character in the world and is motivated to explore the interactions that the space around her provides (Fernández-Vara and Osterweil 2010).

The plot of the game revolves around a crime: someone was killed and it is up to the player, a detective, to find out who did it by gathering clues to arrest the culprit. The plot is constructed from Wikipedia articles and their links. The player begins in the house of the deceased, knowing who died and finding a list of possible suspects. With this list, the detective can travel between locations, talk to NPCs and interact with items. Her goal is to pinpoint who, within that list, is the culprit and prove so by selecting the right options in an arrest warrant template. This template has a series of possible characteristics, such as "residence", "nationality" or "awards received", and each characteristic type has a series of possible values. By selecting the right combination of values, the player can differentiate between suspects.

Initially, the user inputs a person's name to the system, ("Justin Bieber" in this paper's example) who will be, for plot purposes, killed. The system queries DBpedia (Auer et al. 2007), a structured version of Wikipedia, to find possible suspects: people linked to the victim. A genetic algorithm evaluates possible suspects and the relations between them, to optimize the suspect pool and guarantee playability. In other words, to guarantee that it is possible to pinpoint the culprit among the suspects, by assigning characteristics such as "genre: Trip_hop" or "homeTown: Stratford".

Once suspects are selected, the system finds a path between the victim and each suspect. These paths are used to populate the system with game objects: cities, buildings, NPCs, items and dialogue. The process of selecting suspects and paths is exemplified in Figures 1 and 2. Finally, for each object in the game, images are obtained from Wikimedia Commons using Spritely (Cook and Colton 2014).

Choosing Suspects and their Characteristics

In a crime-based story, such as the Sherlock Holmes series (BBC 1965) or the game *"Where in the world is Carmen Sandiego?"* (Brøderbund 1985), the hero/player is often asked to identify the culprit of a crime by solving puzzles and finding clues. In a game where characters and puzzles are created from open data, the challenge of selecting data suitable for suspect and clue creation arise. In a design sense, we want suspects that are related to the victim so that we can establish a motive. Additionally, we want the game to be solved by eliminating suspects when the player discovers that the killer does not have characteristics possessed

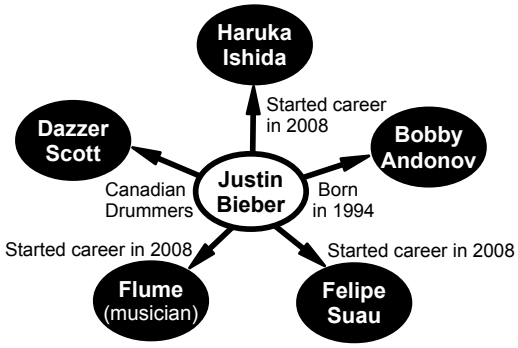


Figure 1: Simple representation of the process of selecting suspects and finding relations between the victim and the suspects. Initially, the system only has a single node: the victim (white node). Suspects related to the victim are selected with a genetic algorithm (black nodes), and paths between the victim and the suspects are created from DBpedia (see Fig. 2). All suspects share common characteristics with the victim, shown on the arrows: in this example three of the suspects started their careers on the same year as the victim.

by other suspects. The killer must therefore have enough unique characteristics not shared by at least one other suspect. In the current design paradigm, each clue should suffice to eliminate one innocent suspect: thus for X suspects (among which one is the culprit), $X-1$ clues (and therefore types of characteristics) are needed.

Creating a pool of possible suspects: It is necessary to create a pool of possible suspects for the genetic algorithm to choose from throughout evolution. To do this, we use DBPedia, which structures information from Wikipedia in the form of tuples[1]. All suspects must share a direct link to the victim based on DBPedia: i.e. suspects must have one or more common characteristic types (predicates) and characteristic value (object) with the victim. For instance, both the victim and a suspect may have lived in the same town, or have the same age (see Fig. 1 for sample shared characteristics). Given these suspects, we generate a list of all possible characteristics types (e.g. "homeTown", "associatedBand", etc.) that one or more suspects have. We omit the value "Living_People", which simply indicates that the person is alive, as well as website-related types (e.g. thumbnails, links for redirected pages, etc) in order to avoid highly abstract relationships as well as a vast, unmanageable search space for the genetic algorithm.

Evolving combinations of suspects: Once we collect all possible suspects, we need to select a number of X suspects among those and $X-1$ types of characteristics (e.g. "placeOfBirth" or "yearsActive"), used to pinpoint the culprit. The suspects' types of characteristics must be varied but common among most suspects; each suspect however must have a unique combination of characteristic values so that we can identify the culprit. A genetic algorithm (GA) is used to find this combination of suspects and types of characteristics. Mutation alone is used as a genetic operator, and in our experiments evolution runs for 100 generations on a population of 100 individuals. The genotype is a vector of size $2X-1$ (X suspects and $X-1$ characteristics). The initial population is generated by selecting random suspects, without repetition, and selecting characteristics within the sub-pool of these suspects' characteristics. The mutation operator changes a few of these elements (i.e. suspects and characteristics). Since we do not have a crossover operation, mutation is mandatory: it will always change at least a few elements. Additionally, if the element mutated is a person, then all types of characteristics undergo a validation check: any characteristic type not possessed by at least one suspect is replaced by a new one.

Fitness function: Our fitness function takes two major concerns into account: diversity and solvability. Diversity measures the distribution of characteristics' types and values for all suspects, and favors characteristics for which values differ more between suspects. For example, a set of suspects living in the same city and owning the same kind of car is less diverse than a set where most suspects live in different cities and drive different cars (but still have the "city" and "car" characteristics). The diversity fitness (f_D) is calculated as the total entropy of each type, multiplied by the number of suspects that have that characteristic type:

$$f_D = \sum_{i=0}^{P} \left(Q_i \times \left(-\sum_{j=0}^{V_i} p_{ij} \left(log_2 p_{ij} \right) \right) \right) \qquad (1)$$

where P is the number of characteristic types, V_i is the number of values for type i, and Q_i is the number of people that have type i. We multiply Q_i to encourage that more suspects have that type. p_{ij} is calculated as the number of people that have a certain value j in characteristic i, divided by Q_i.

Solvability, on the other hand, guarantees that it is possible to pinpoint the killer among the suspects. As discussed above, WikiMystery operates under the assumption that each discovered clue should eliminate one suspect: the clue in this case specifies which value of a characteristic does **not** belong to the culprit, but belongs to an innocent individual instead. To do so, we need to pair the suspects and types uniquely, such that each suspect can be identified from the killer. This fitness is calculated via a form of Depth-First Search (DFS): for each person in a genome, we choose a potential culprit and let the remaining people be suspects. For each one of the $X-1$ characteristic types in the gene, add that type to a 'clue' list, if and only if: 1) the killer has a value for that type of characteristic; 2) at least one of the suspects has that type of characteristic; 3) the value of the killer for that characteristic is different than that of the suspect. This is done to avoid characteristic types that can, single-handedly, allow the player to pinpoint the culprit.

Having chosen a potential culprit, the algorithm creates a 'clue' list for each innocent suspect (s). For each characteristic type in the pool, if s has a different value from that of

[1] An article in Wikipedia may be represented in DBpedia by a collection of ⟨Subject,Predicate,Object⟩ tuples (e.g. ⟨Nikola Tesla, Birth Date, 1856-07-10⟩).

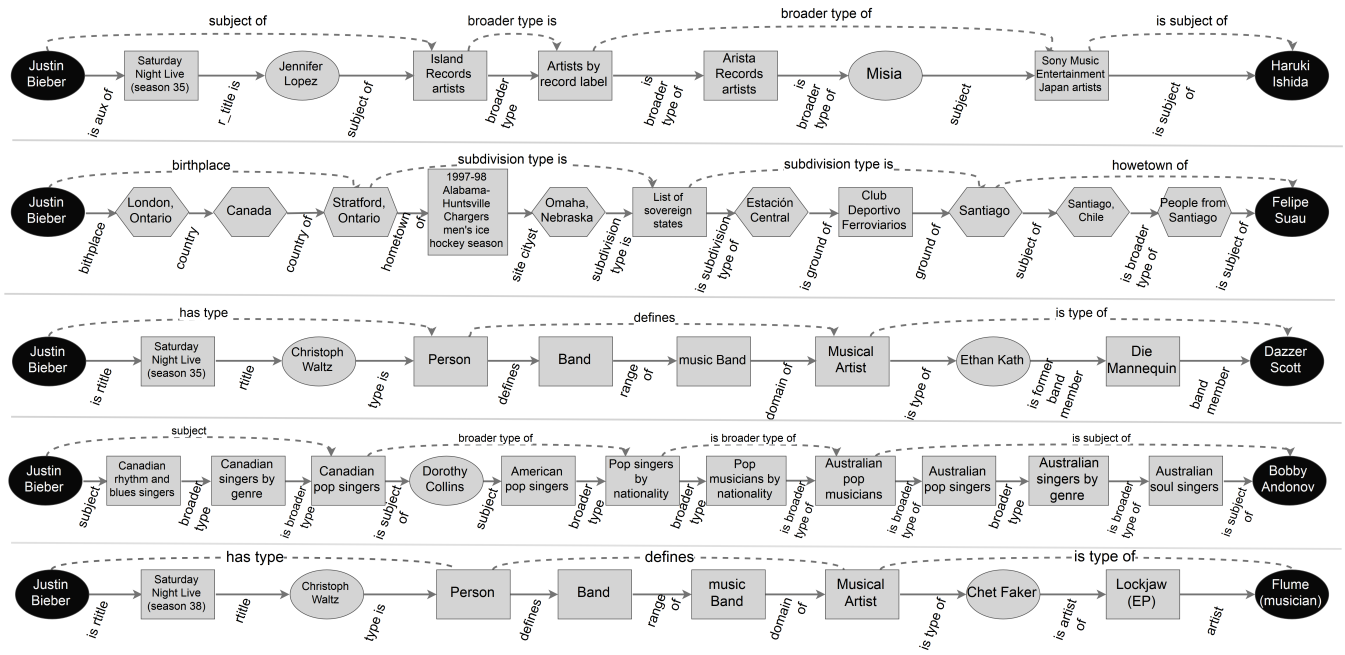


Figure 2: All major and minor paths between the victim and each suspect. Major paths are denoted with dotted arrows, minor paths have black arrows. Locations are represented as hexagons, NPCs as circles and items (books) as squares.

the culprit, then the characteristic is added to the list. The algorithm also inserts an empty symbol in each list, allowing a suspect to remain without any clue. At this point, the algorithm recursively pairs suspects and characteristic types (or empty values), so that each type cannot be used more than once (i.e. if a suspect is paired to a type, this type cannot be assigned to any other suspect). The algorithm backtracks if it does not find the optimal solution, similarly to a DFS, and stops only if it finds the maximum possible fitness or exhausts the search space. If the algorithm did not find the maximum fitness, it will select the next suspect as the potential culprit and restart the process, until it iterates through all X suspects or finds the optimal solution. The fitness is the number of successfully paired types, and the maximum fitness is $X - 1$ (when each type is paired to one suspect).

The GA uses cascading elitism (Togelius, De Nardi, and Lucas 2007) to ensure both fitnesses are optimized: first, the population is sorted using the solvability fitness, and half of the population is eliminated. The remaining population is sorted by diversity, and half of those individuals are removed. The remaining ¼ of the original population is cloned and mutated until the population reaches its original size.

Finding relations between Wikipedia articles

An adventure game can be viewed as a series of linked, concurrently or sequentially, challenges and events. As such, they may be represented as a directed graph, starting at the game's initial state/event, and leading to possible endings. If we identify the victim as the initial node in our graph, and each suspect as a potential ending, we can generate a path between the victim and each suspect, which would essentially amount to the game itself. To generate this graph, our system calculates paths from the victim to each suspect using DBpedia. In this context, a path represents the relations between a sequence of Wikipedia articles, and consists of nodes (articles) and edges (the links between them).

The system queries DBpedia multiple times for possible paths between two individuals. At first, we select a "major path", a longer path between any given two articles that models the general relations between these two. Secondly, for each pair of articles in the major path, we select another "minor path", a shorter, more refined path that expands on the general idea. It provides a longer gameplay and a different, more indirect relation between the victim and the suspect. Figure 2 shows all five paths obtained for a set of suspects: major paths are represented with dotted arrows, and minor paths with black arrows. Paths are evaluated with a weighted sum based on their uniqueness and length. To evaluate uniqueness, we calculate the entropy of the nodes and of the edges of a path in relation to all nodes/paths found in that query: the more uncommon the nodes/edges, the better the path is evaluated. More details on the generation of major and minor paths are provided by Barros, Liapis, and Togelius (2016).

Data transformation

The previous step has secured a set of suspects and a set of characteristic types. This section describes how these DBpedia entries are transformed into playable, interactable game objects, and describes how these objects are placed in the gameworld based on their relationship with the victim. Game objects are divided into three groups: locations, items

```
Initialize empty stack elements;
for i ← 1 to depth-1 do
    if elements not empty then
        Choose an empty node n at depth i;
        Pop top of elements stack and assign to n;
    else if rolled the chance for adding a puzzle then
        Create a random SOLUTION/BLOCK pair;
        Choose an empty node n at depth i;
        Assign SOLUTION to n;
        Push BLOCK to elements stack;
while elements not empty do
    Choose an empty node n at depth depth;
    Pop top of elements stack and assign to n;
```

Algorithm 1: Pseudocode for puzzle placement, for a set of paths of maximum depth *depth*. Empty nodes are nodes with no blocks or solutions assigned to them.

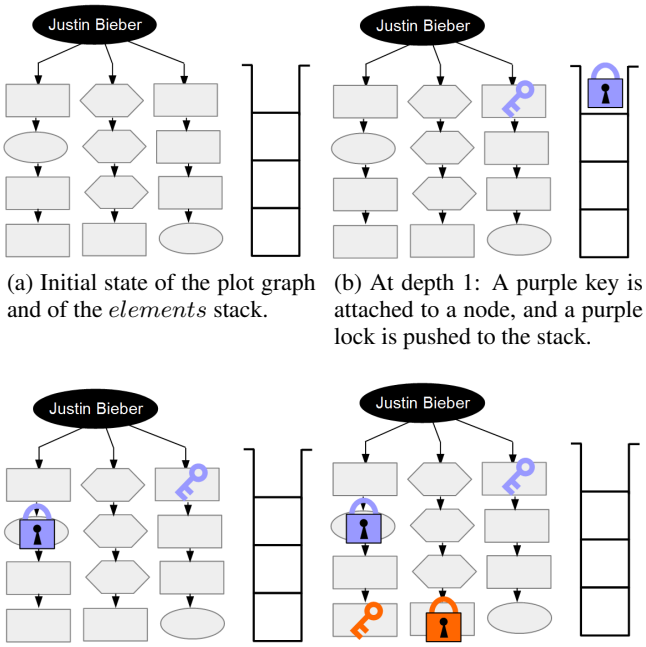
and NPCs. To create them, we use the following guidelines:

- Any article tagged with type "Person" (real or fictional) is instantiated as an NPC. An image of the person is obtained from Wikimedia Commons[2]. If no image is found, a random image of a person of the same sex is used.
- Any article that contains a geographic coordinate and is tagged with type "Place" in DBpedia is transformed into a city or a state. Information about this location is added to the game object created. Locations can be accessed in-game through the world-map. A map of this place is obtained through JMapViewer[3] and OpenStreetMaps (Haklay and Weber 2008).
- If an article has a geographic position, but is not tagged as a "Place" in the DBpedia ontology, it becomes a building: a location within a city/state where the player can interact with NPCs or items. It can be accessed in-game by clicking on the building icon while the player is viewing the map of a city/state.
- If the article does not follow any of the rules above, a game item is created with information on this article. At this point, the game items include only books.

Once all game objects within a path are created, we have something resembling a tree, with the victim as the root, and the suspects at the leaves. However, it is still necessary to add a set of clues and conditions between them, so that each object appears in the path in the correct order. Clues for items include text (e.g. if the next node in the path is a location, the book item will have some text indicating that this location is interesting), and for NPCs they will include dialogue sub-trees. For locations and buildings, however, it is necessary to create either a random NPC or a random item, and set the clue as above. Once all the clues are ready, the algorithm creates conditions between each object, so the player cannot interact with a game object or visit a location before they have seen the previous clue in that path.

[2]Wikimedia Commons: http://commons.wikimedia.org/

[3]JMapViewer: http://wiki.openstreetmap.org/wiki/JMapViewer



(a) Initial state of the plot graph and of the *elements* stack.

(b) At depth 1: A purple key is attached to a node, and a purple lock is pushed to the stack.

(c) At depth 2: The purple lock is popped from the stack and attached to a node.

(d) At depth 4: A new set of orange key/lock is created and attached to different nodes.

Figure 3: Example of position of pairs of solutions (keys) and blocks (locks). In Fig. 3d both the key and the lock are placed at the same depth (as it is the maximum depth).

Finally, additional puzzle elements (keys for locks, flashlights for dark rooms, tickets for private events) are included in the path. The path can be represented as a tree, therefore it is possible to group nodes by their depth, and add such puzzle elements while traversing the tree from the root to all leaf nodes. By always generating pairs of elements in order (SOLUTION: e.g. keys, tickets; BLOCK: locks, private event), we guarantee that the node will be reachable by placing the solution (on any path) at a lower depth than the block. For this, our algorithm works as shown in Algorithm 1. Figure 3a shows the initial state of a tree with three branches and maximum depth 4. At depth 1, the algorithm adds a solution (key) to a random plot node and a block object (lock) to the stack, as seen in Figure 3b. At the next depth, the lock is popped from the stack and added to a random node. At the maximum depth the chance for adding a new puzzle is rolled, so the key is placed in a random node at depth 4 and consecutively, since we are at the maximum depth, the lock is placed at the same depth in a random empty node.

Results

While our system can, theoretically, create mysteries for any given victim, our tests focused on a single subject. The reasoning behind this mainly involved query limitations imposed by DBpedia. To test our suspect and clue selection algorithm, it was necessary to run an enormous amount of queries, which could quickly become a problem for a non-

	Average	(sd)
Number of cities	19.33	(5.07)
Number of buildings	50.60	(3.04)
Number of NPCs	31.00	(8.57)
Number of items	29.60	(5.21)
Ratio of real NPCs (over all NPCs)	82%	(6%)
Average path length per suspect	11.93	(1.03)

Table 1: Average and standard deviation (in parentheses) of game objects generated and average length of paths selected by the crawler for each suspect. Results are collected from 50 independent runs of the generator with Justin Bieber as a hypothetical victim.

dedicated server. To avoid this, we limited our search to Justin Bieber (according to Wikipedia, "a Canadian singer and songwriter") as the hypothetical victim of the murder mystery; as a contemporary celebrity figure, it was expected that he would have many connections in Wikipedia. The system queried DBpedia for every possible article about a person that had some non-trivial characteristic type and value in common with Justin Bieber. As described above, characteristics that were too broad were excluded (e.g. "category: Living people", which includes every person currently alive, and hardly an indicative motive for a crime).

In order to assess, quantitatively, the types of games generated by WikiMystery, the full generative process was executed 50 times, with Justin Bieber as the hypothetical victim and five suspects required ($X = 5$). Table 1 shows the average and standard deviation of the number of game objects created, as well as the average path length per suspect. All of these metrics are indicative of the game's playtime: there are apparently many cities and buildings for the player to visit; however, most locations contain items (books) which provide information and far fewer contain NPCs which the player can talk to. This indicates that discovered paths rely more on categories (e.g. Saturday Night Live, see Fig. 2) rather than people. It should be noted, however, that most NPCs represent real-world people (since less than a fifth of the NPCs are random, and therefore less interesting).

Example Game

An indicative run is included in Fig. 1 which shows the real people chosen as suspects, Fig. 2 which shows the paths between victim and suspects, and Table 2 which describes the characteristic types and values of the suspects. Table 2 shows, in bold, the clue that will be used to prove the innocence of each of the suspects. For instance, Haruka Ishida is eliminated as a possible culprit when the player discovers a clue that the culprit was **not** born in 1993: since Haruka Ishida is born in 1993, he can not be the culprit. Note that some values are shared between more than one suspects (e.g. both the culprit and Flume were born in 1991) and thus can not be used to rule out that particular innocent suspect.

When the game described in the above Figures and Tables is played, the player starts at the city "London, Ontario". In the city map, she can find the house of a dead fictional Justin Bieber and come upon objects that indicate the existence of

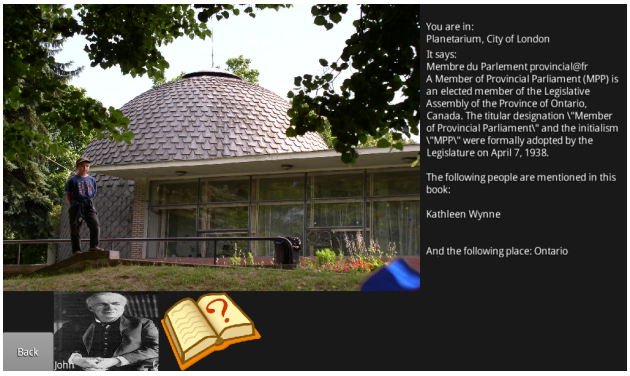


Figure 4: Screenshot of a 'planetarium' building location with a random NPC and a book containing clues.

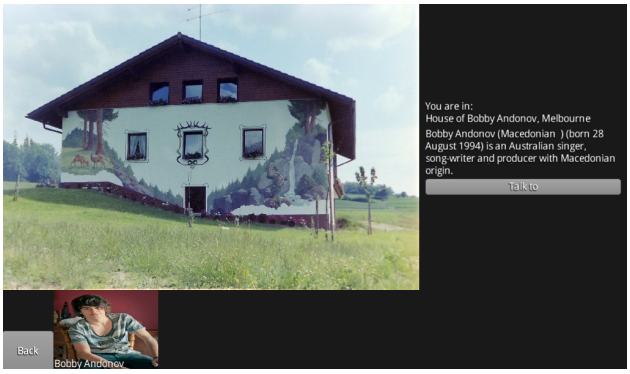


Figure 5: Screenshot of the 'house of Bobby Andonov' location with the NPC Bobby Andonov.

the suspects and the next clue in each path: e.g. for Haruki Ishida, a book on "Saturday Night Live (season 35)". The player can then travel to places (including Omaha and Santiago), talk to various NPCs (some based on real people, such as Jennifer Lopez, others randomly generated) and search for clues (pieces of text in books, letters, etc). At one point, the player stumbles upon a locked house in Toronto, where a fictional Ethan Kath resides. To enter this house, she needs to go to a theatre in Concepción, Chile, where she will find a key. Eventually, the player can find the location of each suspect and, based on the clues gathered along her journey, identify the culprit: (fictional) Felipe Suau. Figure 4 shows a screenshot of a moment in the game, where the player is in a planetarium in the city of London, and can talk to a random NPC named John, or inspect a book she finds nearby. The book is titled "Membre du Parlement provincial" (author's note: Member of Provincial Parliament) and mentions the NPC of Kathleen Wynne and the place of Ontario (which can now be interacted with and visited respectively). Figure 5 shows a building quoted as being the 'House of Bobby Andonov' (the background image is one of a random house), where the suspect NPC instantiated from Bobby Andonov can be found (his image from the Wikipedia article is used, shown at the bottom of the screen).

	Characteristics			
Innocent Suspects	**Genre**	**Occupation**	**Birth year**	**Background**
Flume (musician)	Electronica	**Producer**	1991	Non vocal instrumentalist
Haruka Ishida	J-pop	Singing	**1993**	Solo singer
Bobby Andonov	**Pop music**	Singing	1994	Solo singer
Dazzer Scott	Electronica	Singing	0025	**Solo singer**
Culprit				
Felipe Suau	Electronic music	PUNCHI PUNCHI Director	1991	Non vocal instrumentalist

Table 2: Solution found by the solver for five individuals. Each suspect is paired to one attribute (marked in bold) that will differentiate him from the killer. Note that Dazzer Scott's birth year is not a typo; 0025 is the value returned by DBPedia.

Discussion

The main purpose of the overarching Data Adventures project is to create a generator capable of producing complete adventure games from open data, such as Wikipedia and OpenStreetMaps. The version of the system presented in this paper is able to generate complete adventures with multiple paths. It differs from other approaches which use data to create gameplay because, in this case, the data influences the gameplay directly: if we select a different starting point (i.e. the victim), we obtain a different story, characters, dialogues and general playable paths. Examples of different playthroughs when the starting point is different have been showcased by Barros, Liapis, and Togelius (2016), for a simpler adventure game generator using the same techniques.

The system is capable of evolving an interesting set of suspects, one of which is the killer. It can also successfully map suspects to predicates in such a way that the culprit can be pinpointed among innocents, thus guaranteeing that the game can be won. An interesting note is that the algorithm seems able to cluster suspects within certain domains (e.g. choosing only artists, or only sport-related people), which emerge from the fitness function's attempts at maximizing characteristic types shared among all suspects (musicians will have the same characteristic types such as "associatedBand" or "producer").

At the moment, our system still has limitations. The dialogues, items and puzzles are created from a limited set of templates, which we intend to expand in the future. The latest results also show flaws in the calculation of uniqueness while searching for paths. Our evaluation is based on comparing nodes and edges, which can be represented with words (e.g. "residence", "New_York" or "20th_Century_Mathematicians"). Our current method compares two words with a hard comparison, i.e. if they have one character different, they are different nodes/edges. This implies that nodes like "Pop singers by nationality" and "Pop musicians by nationality" are considered as different as "Canada" and "Misia". Further work should improve this by using a word comparison algorithm, such as Jaro-Winkler (Winkler 1999). We also evaluate the paths separately, not accounting for similarities between paths (Fig. 2 includes paths with many similar or identical nodes among them such as "Band" or "Saturday Night Live"). Measuring how different the paths are to one another is an important step to avoid repetitiveness. Another issue with the paths is the occurrence of very general categories, such as "person" and "musical artist" (see Fig. 2). We are working on heuristics for excluding such categories.

Finally, an important limitation of the current system is that data collection for generating games when seeded with an individual (e.g. Justin Bieber) takes considerable time. This is due to the very large number of database requests needed and the nature of network communications. A priority for further development is therefore to create a version working from a local database.

Conclusion

This paper described a system for generating murder mysteries from open data, which can be used as the basis for generating adventure games. The methods presented here primarily revolve around the collection of appropriate data from Wikipedia articles, and the selection of the best suspects and their characteristics based on criteria of diversity (via data uniqueness) and value (via solvability) for a playable adventure game. Moreover, the paper outlined the necessary first steps for a fully generated adventure game which features the exploration of open data and the transformation of real-world frames and associations into playable experiences.

Acknowledgments

The NPCs discussed in the generated adventures are instantiated from real people, but it should be obvious that the similarities end there. The actions of NPCs in the adventure (as victims or culprits) in no way reflect the real-world people they are based on. The output of the generator in no way accuses or misrepresents these real-world individuals. WikiMystery creates fictional counterparts of public figures who have a presence in Wikipedia: any similarity between the (fictional) NPCs in the game and real-world people is therefore due to the data available in these open, freely accessible, online repositories.

We thank Ahmed Khalifa and Scott Lee for all fruitful and helpful discussions. Gabriella Barros acknowledges financial support from CAPES and Science Without Borders program, BEX 1372713-3.

References

Aarseth, E. 1997. *Cybertext: Perspectives on Ergodic Literature*. Johns Hopkins University Press.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Barros, G. A. B.; Liapis, A.; and Togelius, J. 2015. Data adventures. In *Proceedings of the FDG workshop on Procedural Content Generation in Games*.

Barros, G. A. B.; Liapis, A.; and Togelius, J. 2016. Playing with data: Procedural generation of adventures from open data. In *Proceedings of the International Joint Conference of DiGRA and FDG*.

Bringsjord, S., and Ferrucci, D. A. 2000. Inside the mind of brutus, a storytelling machine. In *Artificial Intelligence and Literary Creativity*. Lawrence Erlbaum Associates.

Cardona, A. B.; Hansen, A. W.; Togelius, J.; and Friberger, M. G. 2014. Open trumps, a data game. In *Proceedings of the International Conference on the Foundations of Digital Games*.

Colton, S.; Jacob, G.; and Veale, T. 2012. Full-face poetry generation. In *Proceedings of the International Conference on Computational Creativity*.

Colton, S. 2002. Automated theorem discovery: A future direction for automated reasoning. In *Proceedings of the IJCAR Workshop on Future Directions for Automated Reasoning*.

Cook, M., and Colton, S. 2014. A rogue dream: Automatically generating meaningful content for games. In *Proceedings of the AIIDE workshop on Experimental AI & Games*.

Cook, M.; Colton, S.; and Pease, A. 2012. Aesthetic considerations for automated platformer design. In *Proceedings of the Artificial Intelligence for Interactive Digital Entertainment Conference*.

Correia, J.; Machado, P.; Romero, J.; and Carballal, A. 2013. Evolving figurative images using expression-based evolutionary art. In *Proceedings of the International Conference on Computational Creativity*.

Eigenfeldt, A., and Pasquier, P. 2012. Considering vertical and horizontal context in corpus-based generative electronic dance music. In *Proceedings of the International Conference on Computational Creativity*.

Fernández-Vara, C., and Osterweil, S. 2010. The key to adventure games design: Insight and sense-making. In *Proceedings of the Meaningful Play Conference*.

Friberger, M. G.; Togelius, J.; Cardona, A. B.; Ermacora, M.; Mousten, A.; Jensen, M. M.; Tanase, V.; and Brøndsted, U. 2013. Data games. In *Proceedings of the FDG Workshop on Procedural Content Generation*.

Haklay, M., and Weber, P. 2008. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE* 7(4):12–18.

Hoover, A. K.; Szerlip, P. A.; Norton, M. E.; Brindle, T. A.; Merritt, Z.; and Stanley, K. O. 2012. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In *Proceedings of the International Conference on Computational Creativity*.

Johnson, D., and Ventura, D. 2014. Musical motif discovery in non-musical media. In *Proceedings of the International Conference on Computational Creativity*.

Krzeczkowska, A.; El-Hage, J.; Colton, S.; and Clark, S. 2010. Automated collage generation – with intent. In *Proceedings of the International Conference on Computational Creativity*.

Laclaustra, I. M.; Ledesma, J. L.; Mendez, G.; and Gervas, P. 2014. Kill the dragon and rescue the princess: Designing a plan-based multi-agent story generator. In *Proceedings of the International Conference on Computational Creativity*.

Lopes, P.; Liapis, A.; and Yannakakis, G. N. 2015. Targeting horror via level and soundscape generation. In *Proceedings of the Artificial Intelligence for Interactive Digital Entertainment Conference*.

Martins, T.; Correia, J.; Costa, E.; and Machado, P. 2015. Evotype: Evolutionary type design. In *Proceedings of the International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design*.

Montfort, N., and Pérez y Pérez, R. 2008. Integrating a plot generator and an automatic narrator to create and tell stories. In *Proceedings of the International Joint Workshop on Computational Creativity*.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):76–99.

Robertson, J., and Young, M. 2015. Automated gameplay generation from declarative world representations. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*.

Thorogood, M., and Pasquier, P. 2012. Computationally created soundscapes with audio metaphor. In *Proceedings of the International Conference on Computational Creativity*.

Togelius, J., and Friberger, M. G. 2013. Bar chart ball, a data game. In *Proceedings of Foundations of Digital Games*.

Togelius, J.; De Nardi, R.; and Lucas, S. M. 2007. Towards automatic personalised content creation for racing games. In *Proceedings of the Symposium on Computational Intelligence and Games*, 252–259. IEEE.

Treanor, M.; Blackford, B.; Mateas, M.; and Bogost, I. 2012. Game-o-matic: Generating videogames that represent ideas. In *Proceedings of the FDG Workshop on Procedural Content Generation*.

Turner, S. R. 1993. *MINSTREL: A computer model of creativity and storytelling*. Ph.D. Dissertation, University of California Los Angeles.

Veale, T. 2014. Coming good and breaking bad: Generating transformative character arcs for use in compelling stories. In *Proceedings of the International Conference on Computational Creativity*.

Winkler, W. E. 1999. The state of record linkage and current research problems. In *Statistical Research Report Series RR99/04*. U.S. Bureau of the Census.