Modern Game AI Algorithms
University of Leiden
May 20, 2021
Julia Wąsala, Elise van Wijngaarden, Louis Robinet
Philippe Bors, Job van der Zwaag, Luuk Nolden

1 Introduction

In this project, we combined Natural Language Processing (NLP) and Procedural Content Generation (PCG) to create the Riddler Game. Its idea is simple: given a topic in the form of a Wikipedia article, the riddler picks a related article and generates questions and answers. The player must now guess the name of the article the questions are about. To aid the player, it is allowed to buy answers to the questions generated. If bought wisely, these answers might help solving the riddle. However, as is the current state of question generation, both questions and answers may make no sense, adding to the challenge: players must try to think like a computer and predict whether buying an answer would be beneficial. The goal is then to guess the topic using as little questions and answers as possible, as revealing more subtracts points from the player's starting score.

In this paper, we will relate about the process of Topic and Subtopic selection, the procedure of generating questions and validating answers, and about the game itself and its features. But first, related work.

2 Related Work

Many different techniques are applied at different levels in procedural content generation for games (PCG-G) research. As described in the survey study by [Hendrikx et al., 2013], the popularity of PCG in games has grown in recent years due to demands from the gaming community for novel content. This process of generating more novel content has been impeded by the design budget and the time-to-market issues [Kelly and McCabe, 2006] [Lefebvre and Neyret, 2003] [Smelik et al., 2009] [Iosup, 2009], which could be solved through the means of automatically generating new content instead of manually crafting it, as has been the custom in game design. [Hendrikx et al., 2013] discuss different levels at which content can be procedurally generated, with examples like textures and vegetation

at the lowest level and ecosystems and world design at the higher level.

The idea of data-games –games at least partially based on open data– is a newer concept, stemming from fields like computational creativity [Barros et al., 2018], where it is shown that human-annotated data can be used as an inspiring set to create creative systems [Ritchie, 2007]. Such systems can be applied in different ways to generate content for games. Examples include level generation for Super Mario Bros inspired by gameplay videos from YouTube [Guzdial and Riedl, 2016], using socioeconomic data to generate playable Monopoly boards [Friberger and Togelius, 2012] and generating adventure games like playable murder mysteries using articles and images from Wikipedia and OpenStreetMap [Barros et al., 2019].

There are many interesting sub fields, one of which focuses generating content catered to individual players. An example of research on this topic is learning-based PCG, where the PCG process is informed by a data-driven machine learning process aiming to learn to create ideal content for a given player [Roberts and Chen, 2015]. Another example is generating educational game content tailored to the player's abilities, allowing the player (for instance children learning to read English) to reach higher performance gains [Hooshyar et al., 2018].

Our research is similar to [Barros et al., 2019] regarding the usage of Wikipedia articles. It differs with respect to complexity: the Riddler game is simpler, and procedural content generation is only applied at one level, as it is generated regardless of the behaviour or skill-level of the player. However, as [Hooshyar et al., 2018], the Riddler game could be used in an educational context, for example as a tool for teachers for classes such as history.

3 Natural Language Processing

In this section we describe everything related to the processing of language: from selecting the topic and generating the questions to validating a provided answer.

3.1 Topic selection

To play the game, the player needs to insert an overall topic, which needs to be an existing page on Wikipedia. For example, having provided the topic 'Nintendo', the game uses the Wikipedia API to search for ten Wikipedia articles relevant to the input string. To ensure relevancy, the topic selection code will remove any lists and disambiguation pages from this list, since these are not usable as topics. The

results are then displayed as options to choose from. The player can now simply pick 'Nintendo' as an overall topic, but can also adjust focus to the related article about the 'Nintendo DS'.

3.2 Subtopic generation

After choosing a topic, the game will try to find subtopics. To do this, the selected Wikipedia page is visited and all its first paragraph and sidebar hyperlinks to other Wikipedia pages are collected. After unusable links like lists, disambiguation pages, the main page, files, categories, portals and others are removed, the remaining ones are ordered by the number of page views by humans in the last month. Next the pages with the most page views are visited to check whether said page is actually relevant to the main topic. This is done by checking if the first paragraph or the sidebar of the subtopic also mentions the topic. Once there are 10 relevant subtopics found, the pages are returned to the game. Next, the game randomly chooses one of the subtopics for the player to guess. To illustrate the procedure, having selected the topic 'Nintendo', the subtopic generation finds the article 'Satoru Iwata', which is referenced and references the former article.

3.3 Question generation

To generate questions, the open source NLP library QuestGen is used [Golla, 2019]. Having received the URL of the selected topic, Beautiful Soup is used to extract the title and the text of the Wikipedia article. From the text, the first one hundred characters and every character after that until a full stop is removed. This is done to create better questions, as the first few sentences of an article are rather global and bland, resulting in extremely simple questions like 'What is a Japanese company: Nintendo'. Next, the question generator is called for the next fifty characters (until a full stop) and the resulting questions are written to a pandas dataframe. As long as there are less than ten questions generated, the generator is called on the following fifty characters, until we have ran out of characters or have reached ten questions. If not enough questions could be generated, the player is notified and asked to find a different topic. This happens mostly on smaller articles with not enough text.

To ensure that questions do not reveal the topic, a penalty is applied for every word in the question that is also part of the topic string (stop words excluded). For example, if the question is 'What is Iwata's place of birth?' and the topic is 'Satoru Iwata', one penalty point is given to this question, as it contains the word

'Iwata'. The same goes for the answer. The dataframe is then sorted by penalty (lowest penalty on top) and written to the disk to be read by the game.

As the generation of questions can take a while, we added the option to not generate questions and play the game using the questions already saved to disk. This allows for separate question generation and playing of the game: a load time of three minutes for every playout would be quite irritating.

3.4 Answer validation

To ensure fair answer validation, we incorporated certain techniques for handling the user's input. Firstly, the system that we produced is not case sensitive allowing both capitalised and lowercase characters to represent the correct answer (e.g. "Universiteit Leiden" vs. "universiteit leiden"). This is especially useful for topics that include names to some extent.

Another important aspect is that names of people are commonly expressed using initials. The answer validation system allows all possible combinations of initials and surname to be entered by the user, and these are all validated as correct (e.g. "Pieter Pietersen Heijn", "P. Pietersen Heijn", "P.P. Heijn", etc.).

Thirdly, we allow the user to express numerical characters by writing them out and vice versa ("Windows 7 vs. Windows Seven").

Lastly, in our observation of the structure of Wikipedia titles, we noticed that parentheses are frequently used to attain title disambiguity. A title can refer to multiple different entities (e.g. "The Batman" vs. "The Batman (film)") and parentheses with additional information are therefore used to remove the ambiguity in the names. Naturally, users of the Riddler game are not obliged to enter this additional information.

Using the methods described above, the answer validation system creates a list consisting of correct possible answers, all derived from the original answer. These answers are then compared to the answer that the user provided using Pythons SequenceMatcher. This matcher computes the ratio of agreement between all possible answers and the provided one. If at least one possible answer is similar enough (using a certain threshold) to the user input, we congratulate the player with their victory. Therefore, this system allows the user to make small spelling mistakes and small language errors without brutally being punished for it. Furthermore,

it allows for words being left out, like "DC Universe" being correct when "DC Extended Universe" is the desired answer.

4 PyGame

In this section, the interface and its creation is discussed.

4.1 Structure of the Game

The first page the player will see when opening the Riddler game is shown in Figure 1 on the left. The title of the game is shown at the top of the page, next to the 'exit' button (which logically will result in closing the game window after being pressed). Under a short explanation of the game, a text field is located which allows the player to search for a topic.

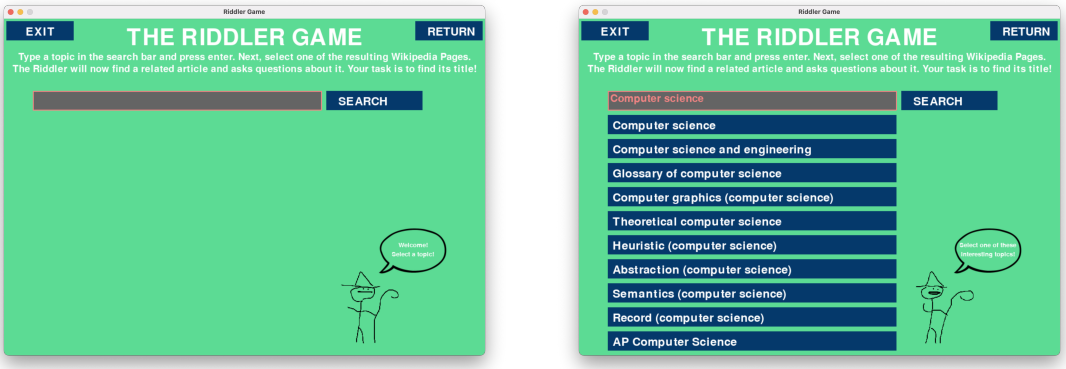

Figure 1: The starting page of the game [left] and the resulting search page [right].

As can be seen in Figure 1 on the right, the player typed in the string 'Computer Science' and clicked the 'search' button. Wikipedia articles similar to the input strings are then displayed underneath. The player can now select one of these topics to play the Riddler game with. Having pressed one, the game will find related Wikipedia pages and generate questions about one that it deems interesting (see Section 3.3). When the questions are generated, the 'play game' button appears, allowing the player to start the game. It is possible however that no topics, subtopics or questions can be generated (this mainly happens for obscure topics with shorter Wikipedia articles). If this is the case, the player will be notified by the little riddler animation on the right, which will try to help the player navigate

this menu.

After the player presses the 'play game' button, the screen changes into the game mode. This screen is shown in Figure 2 on the left. In this screen, the 'exit' button, 'menu' button and the score are situated at the top of the window. Pressing the 'menu' button will result in going back to the screen related in the previous paragraph. Underneath are the questions shown which are clues towards the topic that the player should guess. The game starts with one question, after which the player can choose to show another question by clicking the button that says 'show another question'. This however results in the score going down, making it worthwhile for the player to try and guess the topic with as little clues as possible. The player can also choose to buy an answer to one of the questions if he thinks this will help him unravel the riddle. This can be done by clicking the button which says 'buy an answer'.

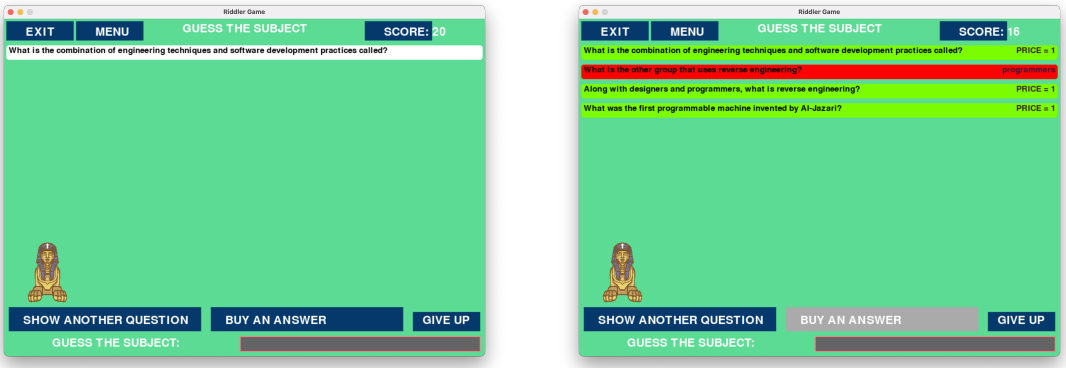

Figure 2: The game when it is played [left] and the page to buy answers [right].

Having pressed the 'buy an answer' button, the screen will change to the state that is shown in Figure 2 on the right. Here, the questions are shown in either green or red colour. If the question is green, it means that the answer to this question is not yet bought and that the score of the player is high enough to buy it. If the question is coloured red, it means that the player cannot buy the answer to this question because it is too expensive, or that the player already bought the answer. If not yet bought, the price is stated on the right side of the question. Otherwise, the answer is stated in the same place, as can be seen in Figure 3 on the right.

As there is a limited amount of questions generated per topic (eight in the current implementation), the screen in Figure 3 on the left is shown if all of the available questions have been displayed. The player is now still allowed to make guesses and

buy answers, but will know that no more clues in the form of questions are in the pipeline.

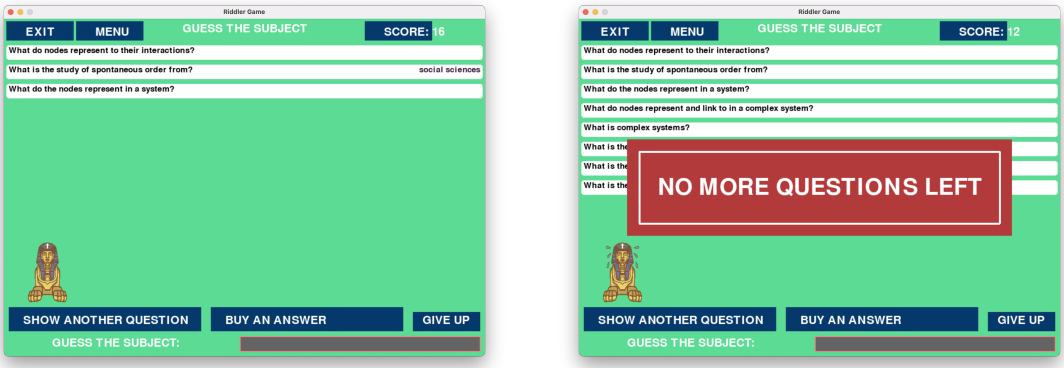

Figure 3: The game when answers have been bought [left] and when there are no more questions [right].

Next, the player can click the 'give up' button, which shows the answer, but also results in losing the game (shown in Figure 4 on the left). Last but not least, the player can type the answer into the text field at the bottom of the game. If the answer provided is correct, the player wins the game (shown in Figure 4 on the right). If not, points are subtracted and the player can try again (if enough points remain).

For every state described above, the Sphinx animation will react accordingly, giving the player feedback and hopefully cheering him on. After winning or losing the game, the player can exit altogether or return to the menu to play again.

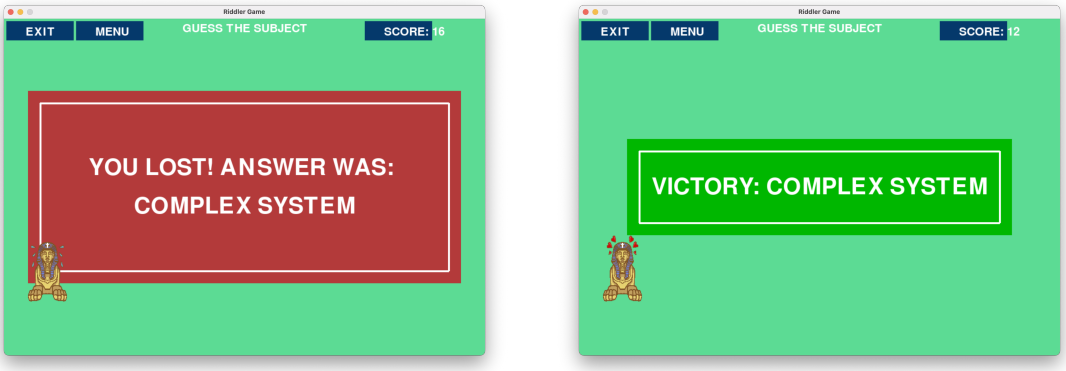

Figure 4: The game when the player gives up [left] and when the game is won [right].

5 Discussion

When researching the possibility of generating questions from open data, one rather significant problem arose with every generator we found online: creating questions from a text is a hard problem for computers at the moment of writing this paper. Before preprocessing the text provided to the generators, the libraries had a hard time to come up with any questions at all and if they did, they were extremely nonsensical. For example: "What is the Microsoft Corporation? Microsoft Corporation." Because of this problem, we came up with the Riddler game, trying to turn this faulty behaviour into a feature. To give the game more usefulness in the future however, for example to test school children on their knowledge of a subject, a better question generator could be switched in.

Otherwise, it would be great to further tweak the QuestGen library that we used. Although it did provide us with the best answers (compared to other question generators), there is no documentation for this library available and no way to change parameters via function calls, making it hard to tweak its performance. Further research should therefore explore the code made available on Github and try to tweak parameters (for example those of the language model BERT) to try and get better results.

Another area of focus would be time consumption. To generate three questions, the library needs at least a solid minute. To create eight questions that meet our requirements, the player has to wait for up to five minutes before the game can be started. Although we alleviated this problem a bit by writing the generated questions to a CSV file, therefore allowing to start playing immediately from a

saved state (via the return button), it remains tedious to wait for the generation. This is made worse if the code has not been able to generate enough questions and asks the player to try again.

Lastly, an interesting scoring system has not been implemented, as one of our team members working on this part had to attend urgent private matters. Therefore, a rather simple scoring system has been used. This starts the player with eight points and subtracts one for every question revealed. At the moment, every answer simply costs one point of score. The player can thus choose to buy all eight available questions, or fewer and some answers. Sadly, no point subtraction has been implemented for guessing a topic, allowing the player to try infinitely. Future work on this project should therefore continue on a more interesting scoring system, as it would add an interesting dimension to the game. For example, if guessing the topic costs three points, one might want to reveal more questions to prevent a game over and maximise the score.

6 Conclusion

We have created a game that derives its content from open data in which the player has full control over the topic, providing them with a level of customisation, adding to the replayability of the game. Having provided the game with the topic of choice, it procedurally generates a subtopic and questions about it, supplying the player with unlimited amount of varying content. Using PyGame, the questions generated have been put into the Riddler game, which is, according to us, a rather fun game to play, especially with multiple people trying to make sense of the mumbo jumbo the computer came up with. Let us therefore enjoy it until we reach singularity and the AI will make fun about our mumbo jumbo (or worse).

References

[Barros et al., 2018] Barros, G. A. B., Green, M. C., Liapis, A., and Togelius, J. (2018). Data-driven design: A case for maximalist game design.

[Barros et al., 2019] Barros, G. A. B., Green, M. C., Liapis, A., and Togelius, J. (2019). Who killed Albert Einstein? From open data to murder mystery games. *IEEE Transactions on Games*, 11(1):79–89.

[Friberger and Togelius, 2012] Friberger, M. and Togelius, J. (2012). Generating interesting Monopoly boards from open data. pages 288–295.

[Golla, 2019] Golla, R. G. (2019). Questgen. https://questgen.ai/.

[Guzdial and Riedl, 2016] Guzdial, M. and Riedl, M. (2016). Toward game level generation from gameplay videos.

[Hendrikx et al., 2013] Hendrikx, M., Meijer, S., Velden, J., and Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9.

[Hooshyar et al., 2018] Hooshyar, D., Yousefi, M., Wang, M., and Lim, H. (2018). A data-driven procedural-content-generation approach for educational games. *Journal of Computer Assisted Learning*, 34.

[Iosup, 2009] Iosup, A. (2009). POGGI: puzzle-based online games on grid infrastructures. In Sips, H. J., Epema, D. H. J., and Lin, H., editors, *Euro-Par 2009 Parallel Processing, 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings*, volume 5704 of *Lecture Notes in Computer Science*, pages 390–403. Springer.

[Kelly and McCabe, 2006] Kelly, G. and McCabe, H. (2006). A survey of procedural techniques for city generation. *Institute of Technology Blanchardstown Journal*, 14.

[Lefebvre and Neyret, 2003] Lefebvre, S. and Neyret, F. (2003). Pattern based procedural textures. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, I3D '03, page 203–212, New York, NY, USA. Association for Computing Machinery.

[Ritchie, 2007] Ritchie, G. (2007). Some empirical criteria for attributing creativity to a computer program. *Minds Mach.*, 17(1):67–99.

[Roberts and Chen, 2015] Roberts, J. and Chen, K. (2015). Learning-based procedural content generation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 7:88–101.

[Smelik et al., 2009] Smelik, R., de Kraker, K. J., Groenewegen, S., Tutenel, T., and Bidarra, R. (2009). A survey of procedural methods for terrain modelling.