aMUSE: Translating Text to Point and Click Games

Martin Černý [1] and **Marie-Francine Moens** [2]

Abstract. In this demo we will show aMUSE — a system for automatically translating text, in particular children stories, to simple 2D point and click games. aMUSE consists of a pipeline of state-of-the-art natural language processing tools to analyse syntax, extract actions and their arguments and resolve pronouns and indirect mentions of entities in the story. Analysed text serves as data the game mechanics operate on, while the story is represented graphically by images the system downloads from the Internet. The system can also merge multiple stories from a similar domain into a branching narrative. Users will be able to both play games created by aMUSE and create games from their own texts using the aMUSE editor.

1 INTRODUCTION

Video games are a powerful media for telling stories and for transferring experiences and feelings in a more general sense. Games are different to most other art forms in that they require active collaboration on the receiver's part. Thus adapting a story to the video game genre requires more than visualisation of the story events on screen: The game mechanics must also be designed to support the story or actively convey parts of the experience.

Recent research has shown that both game design and adaptation of text to game can be, to some extent, performed automatically. Most of the work so far either a) focuses on the game mechanics and does not consider the story of the game, or b) uses a large amount of domain-specific knowledge.

In this demo we will show aMUSE — a system that can automatically translate stories given in natural language to simple games without using any domain-specific knowledge. As our focus is on the story, we have chosen to generate games in the 2D point and click adventure genre. Games in this genre are inherently story-driven and consist of the player clicking on various objects to trigger interactions. If the correct interaction is found, the story progresses further. We have chosen this genre as it allows for a very direct mapping between the story and the game mechanics.

2 RELATED WORK

A system called Angelina can fully automatically design simple 2D and even 3D games [3, 2]. Game-o-matic [9] uses common-sense knowledge databases to generate 2D arcade games involving given topics. Our work is orthogonal to these efforts as it translates a story written in a natural language to a predefined game mechanic instead of generating the mechanics.

In the context of adapting a text to an interactive experience, De Mulder et al. [4] discuss transforming patient guidelines into educational 3D experiences. The authors use a large domain-specific knowledge base to provide common-sense grounding to the fragmentary information present in the text.

Some progress has been made on generating 3D scenes from text to be later used in a whole interactive experience [5]. However, the system is not fully automatic, as it relies on crowdsourced domain-specific knowledge to correctly position the entities in the scene and does not produce playable experiences yet.

3 THE SYSTEM

The aMUSE system consists of four parts: editor, translator, server and frontend. The editor is a graphical application that lets the user enter stories, group stories to form projects and control the execution of the translator. The translator is responsible for finding an interactive representation of the story which is passed to the frontend. For fast startup of the translator and due to some technical aspects of the technologies used, some of the tasks performed by the translator are carried on a dedicated server. The frontend is a simple game engine written in Flash that visualises the game provided by the translator.

To translate a story, the translator first passes it to the server. The most important part of server-side processing is *semantic role labelling* (SRL) using the Lund pipeline[3]. SRL builds upon syntactic features of the sentence to discover *semantic frames*. A frame represents a concept in the sentence (the *root*) and annotated arguments of the concept (the *roles*). We use frame definitions given in PropBank[4].

For example the sentences "The city was taken by the Romans" and "The Romans took the city" have different syntax, but both contain the frame $take.01(taker : Romans, thingTaken : city)$. The numbered suffix to the frame root distinguish between various meanings of the same word: e. g., "I cannot take it anymore" would resolve to $take.02(tolerator : I, thingTolerated : it)$. The Lund SRL was trained on news texts, so we used transfer learning [8] to adapt it to handle stories better.

The last crucial part of server-side processing is coreference resolution using Stanford CoreNLP [6]. Coreference resolution links all mentions of the same entity (pronouns, in particular) throughout the whole story. The annotated text is then returned to the translator.

The translator uses the semantic frames to find possible interactions for each sentence of the story. In our case, interaction is an agent-action-target triplet, where either agent or target may be omitted (but not both). All frames with roots that are verbs are candidates for interactions. Simple hard-coded heuristics are used to choose the agent and the target among the frame's roles.

Now, every story is represented as a linear multigraph with sentences as nodes and possible interactions as edges from the previous sentence to the sentence that defined the interaction. Optionally, the translator can merge multiple stories to form a non-linear story

[1] Charles University in Prague, Czech Republic, email: cerny.m@gmail.com
[2] KU Leuven, Belgium, email: sien.moens@cs.kuleuven.be

[3] `http://nlp.cs.lth.se/`
[4] `http://verbs.colorado.edu/propbank/`

graph. To achieve this we check all pairs of sentences A, B. If they are from different stories, but have similar frames then for each interaction (X, A) we add (X, B) to the graph and vice versa, i.e., at these nodes the game can switch to a different story, depending on the interaction chosen by the player. This approach was inspired by the story generation process described in [7].

The translator then lists all the entities present in the story and schedules at which point in the story they should appear. As coreference resolution is not flawless, we make the simplifying assumption that two entities with the same name are the same and merge the respective entity mentions. The translator then requests images for the entities from the server which uses Spritely [1] for this task.

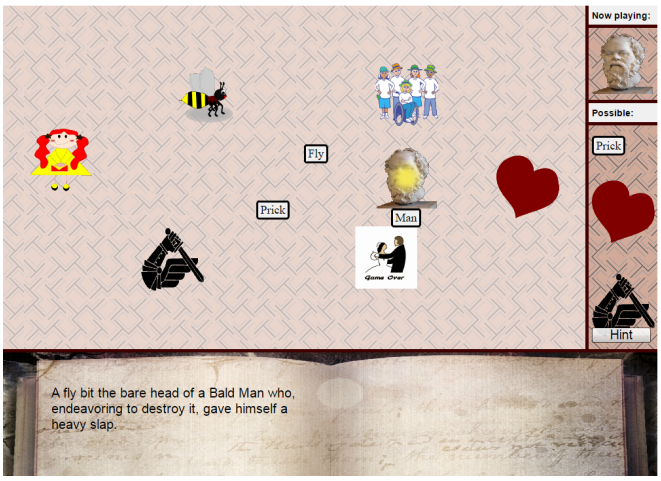

Figure 1. Screenshot of the aMUSE frontend.

The frontend then uses the story graph as the basic structure to guide gameplay. It keeps the current node in the graph and when the user performs an interaction corresponding to any of the outgoing edges, the story progresses to the edge's target, i.e., every action of the user corresponds to progressing the story one sentence further.

Originally, we intended that the user will represent the protagonist of the story and perform only the interactions where he is the agent. In this case, the other interactions would be performed by the system automatically as a kind of a cutscene. This however led to a large number of non-interactive nodes, so we decided to alter the game design a little: the user is no longer a character in the story; he represents a disembodied entity, whose single goal is to make the story happen. To do this, the user can take control of any active entity and act (click on objects) on its behalf. The resulting interactions are very abstract and it is almost impossible to decipher the story from the interactions themselves. To allow the player to follow the story, the original text of the sentence is shown in a stylized book. The screenshot of the frontend is given in Figure 1.

So far, we have not been able to finish our work on extracting spatial relationships between the entities from text, so the entities only float around the screen without any structure.

4 CONCLUSION

Our system is capable to automatically translate stories written in natural language into a specific type of playable experiences. While many of the interactions that the system produces make sense, it also produces absurd options, mostly due to imperfections in natural language processing (NLP). To some extent, this can be enjoyable from the user perspective, but there is definitely room for improvement.

The system works reasonably well on short stories targeted at very small children, as the vocabulary and syntactical structure is simple. However, the main reason that short stories work better than longer ones is that the gameplay is very limited and it is not fun to click through a longer story. Although longer stories also degrade accuracy of coreference resolution. Semantic and syntactic complexity of the text is currently the most limiting factor for our tool. We tested the system on Aesop's fables, where the resulting gameplay was still more often relevant to the story than not. However, when run on fairy tales collected by Andrew Lang, which have long and complex sentences and archaic language style, only a minority of the resulting interactions were reasonable. Further issues arise from incorrect association of words with images.

Our system can serve as a demonstration of the power (and remaining deficiencies) of the contemporary NLP technology. We believe that NLP is at the level where it can improve games and gaming experience. While we are aware of game-related research using syntactic analysis of texts, we are not aware of usage of SRL in this context, although there are high possible benefits.

Examples of games created by the system can be played online[5] and the system itself is fully open-source.

ACKNOWLEDGEMENTS

This research is partially supported by the EU FP7-296703 project MUSE, student grant GA UK No. 559813/2013/A-INF/MFF and by SVV project number 260 224.

REFERENCES

[1] M. Cook. Spritely — autogenerating sprites from the web. `http://tinyurl.com/spritelypost`, (2013). Last checked: 2015-01-16.

[2] M. Cook and S. Colton, 'Ludus ex machina: Building a 3d game designer that competes alongside humans', in *Proceedings of the Fifth International Conference on Computational Creativity*, pp. 54–62, (2014).

[3] M. Cook, S. Colton, A. Raad, and J. Gow, 'Mechanic miner: Reflection-driven game mechanic discovery and level design', in *16th European Conference on Applications of Evolutionary Computation*, volume LNCS 7835, pp. 284–293. Springer, (2013).

[4] W. De Mulder, Q. Ngoc Thi Do, P. Van den Broek, and M.-F. Moens, 'Machine understanding for interactive storytelling', in *Proceedings of KICSS 2013: 8th international conference on knowledge, information, and creativity support systems*, pp. 73–80, (2013).

[5] R. Hodhod, M. Huet, and M. Riedl, 'Toward generating 3d games with the help of commonsense knowledge and the crowd', in *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 21–27, (2014).

[6] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, 'The Stanford CoreNLP natural language processing toolkit', in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, (2014).

[7] N. McIntyre and M. Lapata, 'Plot induction and evolutionary search for story generation', in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1562–1572. Association for Computational Linguistics, (2010).

[8] Q. Ngoc Thi Do, S. Bethard, and M.-F. Moens, 'Text mining for open domain semi-supervised semantic role labeling', in *Proceedings of the First International Workshop on Interactions between Data Mining and Natural Language Processing*, pp. 33–48, (2014).

[9] M. Treanor, B. Blackford, M. Mateas, and I. Bogost, 'Game-o-matic: Generating videogames that represent ideas', in *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, p. 11, (2012).

[5] `http://tinyurl.com/amuseExamples`