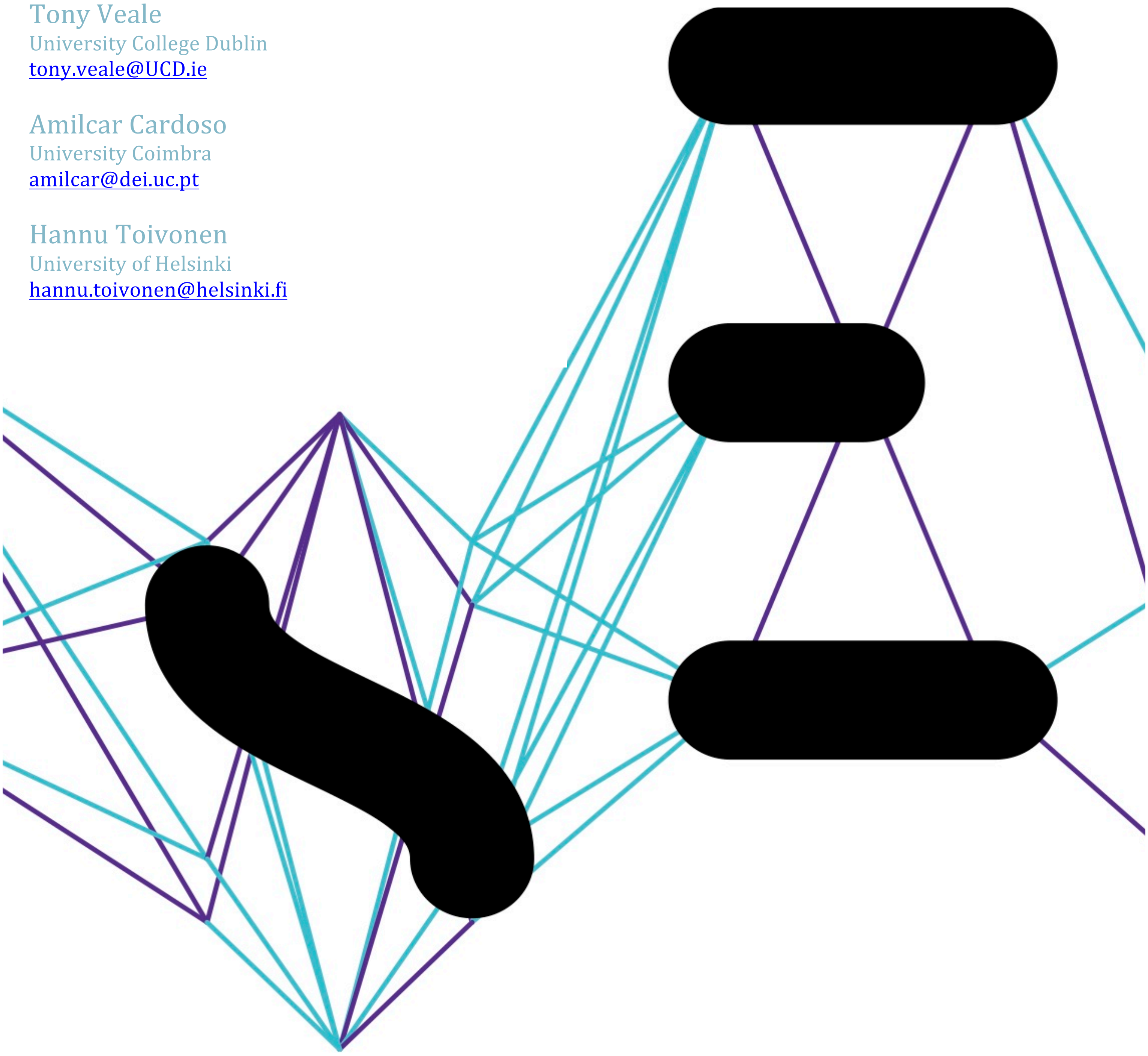
Final Report on CC Schools and Code Camps

Deliverable: D3.6

Tony Veale
University College Dublin
tony.veale@UCD.ie

Amilcar Cardoso
University Coimbra
amilcar@dei.uc.pt

Hannu Toivonen
University of Helsinki
hannu.toivonen@helsinki.fi



PROSECCO PROMOTING THE SCIENTIFIC EXPLORATION OF COMPUTATIONAL CREATIVITY

PROSECCO Education Events: Schools & Camps

1. Overview

A key pillar of the PROSECCO coordination action is its educational programme, which seeks to inform and shape the next generation of Computational Creativity (CC) researchers. To this end, the project has organized a major tent-pole educational event in each of its three years. Beginning with an Autumn School in 2013, the project organized a code-camp in both 2015 and 2016, with a variety of smaller events (such as targeted tutorials) spread between these tent-poles.

This report summarizes the results of these three educational events and seeks to draw some conclusions about the future of educational outreach in CC beyond the PROSECCO project.

1.1. Structure of this Deliverable

In the following section (section two) we present an account of the 2013 Autumn school in CC, which was held in Porvoo, Finland. In section three we present an account of the 2015 code-camp which was held in Coimbra, Portugal. In section four we present an account of the 2016 code-camp which was held in Antwerp, Belgium. As we shall report, these events differ in interesting ways, so in section five we shall aim for a synthesis of the lessons learnt from these three experiences.

2. The First International PROSECCO Autumn School

The 1st PROSECCO Autumn School was organized by PROSECCO partner **Hannu Toivonen** of Helsinki University. The small historical town of Porvoo, outside Helsinki, was chosen as the venue for several reasons: firstly, the town, though quaint and historically interesting, is not a source of distractions to students on the same scale as a metropolitan centre such as Helsinki; the town is easily accessible from Helsinki airport; and the town was the location of the successful 2011 Autumn school in Computational Creativity, also organized by **Hannu Toivonen**.

A large audience of students (approx. 60 students from across Europe, or 70 attendees in all) was in attendance for the classes of the Autumn school. All sessions were sequential with no parallel sessions to divide audiences. All lectures were well-attended and robust questions were encouraged and answered after each lecture. For instance, the psychologist Rachel Giora provided a splendid set of lectures that educated existing CC researchers in the ways of cognitively-plausible research and evaluation, and she also successfully challenged some deep-seated preconceptions and prejudices regarding the relation of computational creativity to human creativity.

The following lecturers delivered courses at the 2013 Autumn school:

- **Tony Veale,** *University College Dublin, Ireland:* **(PROSECCO partner)**
- **Simon Colton,** *Goldsmiths College, University of London, UK:* **(PROSECCO partner)**
- **Geraint A. Wiggins,** *Queen Mary University of London, UK:* **(PROSECCO partner)**

- **Rachel Giora,** *Tel Aviv University, Israel:* **(Invited speaker)**
- **Amilcar Cardoso,** *Universidade de Coimbra, Portugal:* **(PROSECCO partner)**
- **Pablo Gervás,** *Universidad Complutense de Madrid, Spain:* **(PROSECCO partner)**
- **Nada Lavrač,** *Jožef Stefan Institute, Slovenia:* **(PROSECCO partner)**
- **Rob Saunders,** *University of Sydney, Australia:* **(Invited speaker)**
- **Graeme Ritchie,** *University of Aberdeen:* **(Advisory board member)**
- **Bipin Indurkhya,** *IIIT, Hyderabad, India:* **(Advisory board member)**

This Autumn school was a flag-ship event for the PROSECCO project, and was thus the subject of considerable advance planning to ensure its eventual success. The event thus served as a model for the planning of subsequent PROSECCO educational events.

A registration fee of 200 euros (early) and 280 euros (late) was charged for participation in the event, but this fee was waived for students, most of whom also received a PROSECCO grant to cover their accommodation and meals at the event. Attracting new students and new blood into the field is a key objective of PROSECCO, so the utilization of funds to defray student costs was an early priority. The average student grant, inclusive of fees, meals and accommodation, was approx.. 500 euros per person for this event (or 489 euros, to be precise), and in all, 45 students received a grant for the Autumn school. In addition, the school received 6 regular (no-grant) attendees and 8 late (no-grant) attendees, in addition to the 10 invited speakers and the host.

The school's host, PROSECCO partner Hannu Toivonen of Helsinki University, supplemented PROSECCO funds with sponsorship from three other sources: HECSE (2000 euros), the University of Helsinki (2000 euros), and FICS (2500 euros). This additional funding allowed the school to host a number of successful social events that encouraged students and other attendees to better integrate within the CC community. Thus, a *Computational Creativity Club* (CCC) event was held mid-week, in which attendees were invited to demonstrate (and perform) the outputs of their CC software in a convivial party setting. Additionally, an evening creativity workshop was held early in the week, to serve as both a social mixer and as a means of playfully engaging attendees on various approaches to creative problem-solving. This evening event was hosted by international advisory board member Bipin Indurkhya.

The two largest costs for the school were accommodation (9,500 euros) and the waiving of student registration fees (9,000 euros). In addition. 3,500 euros was spent to obtain professional video recordings of all of the lectures (via videolectures.net), 4,400 was spent on catering, 1000 euros was spent on the event venue, and 500 euros was spent on bussing participants to and from airports. The event was financed within the projected budget due to the supplementary sources of funding and to the use of PROSECCO partners as speakers (as the latter could use their own PROSECCO funding to finance their travel to the school).

Though a success in every way we had hoped, and in others we could not have expected, the school was to be the first and the last of its kind in the PROSECCO project. Having established an educational foundation for the rest of the project, we decided to focus in the following years on practical hands-on events in which participants would learn to build CC systems of their own. In part this decision was predicated on the success of the initial Autumn school, and on the availability of online lectures from the school that removed the need for further large-scale lectures on the same subjects, but it also followed from the availability of other online educational resources such as the PROSECCO entry-level text book (at *RobotComix.com*). With a solid foundation of educational resources to build upon, we were thus freed to pursue a more hands-on agenda in the subsequent years, as we shall discuss next.

3. The First International PROSECCO Code-Camp

Given the rationale above, the principal educational outreach event in year two was the *First International PROSECCO Code-Camp*, which was held in Coimbra, Portugal in January 2015 (12th to 15th). The local chair for this event was PROSECCO partner **Amilcar Cardoso**, who also served as program co-chair with PROSECCO coordinator **Tony Veale**. Having learned some important organizational lessons from the Y1 Autumn School – and reusing some of the same tools, such as the UH online registration service – we were able to give this event the large amount of organizational and conceptual effort that was required to execute it successfully.

As with the tutorial offered as part of ICCC 2014, the focus of the code camp (which carried the strapline "*Postcards from the Edge of Tomorrow*") was creative Twitterbot construction. The 32 students who were funded to attend the camp were allocated to eight teams of 4 students each, in such a way that core competences (e.g. Java/Python development skills) were spread as evenly as possible across groups (the same approach, having received positive feedback here, was to be reused in the second code-camp a year later). As such, great care was taken to ensure that each team was a viable combination of technical and conceptual skills. Each team was also assigned a mentor, an established CC researcher, to ensure that they answered the challenge of the camp in a way that was neither too ambitious nor too uncreative. This involved a delicate balancing act from which we learned a great deal for future events (the constructive feedback of mentor Graeme Ritchie, also a member of PROSECCO's advisory board, was especially useful in this regard).

As is it a major goal of PROSECCO outreach events that young researchers be drawn into active roles in the CC community, we considered it vital that young researchers play a key role in the delivery of the code camp's educational content. To this end, two active young researchers – games researcher Mike Cook (from London) and design researcher Tom DeSmedt (from Antwerp) – were invited (and sponsored via PROSECCO travel missions) to deliver the opening lectures at the camp. Mike Cook delivered an excellent survey talk on Twitterbots, focusing on both their techniques and their artistic/conceptual rationale, while Tom DeSmedt gave an invaluable technical tutorial on the use of the *Pattern* library for Python (of which he is a co-developer). Student teams were then tasked with using Python/Pattern to build their own Twitterbots, taking inspiration from the survey talk by Mike Cook to inject a unique theme and personality into their creations. Students found the Twitterbot construction task to be fun *and* technically challenging, with a topical internet resonance that nonetheless echoed the excitement of the '60s experimental poets and their innovative use of the cut-up method.

It was of utmost importance that students not fall into the trap of *Mere Generation* – the alluring belief that machines can be programmed to generate creative outputs without being able to appreciate those outputs for themselves – but to instead build generative systems that would be accepted as creative by the CC community. Our machines cannot appreciate their own outputs if they lack knowledge about the semantic components of their outputs, hence the need to provide students with a comprehensive knowledge-base of interconnected and semantically-grounded beliefs, and the need to enforce the use of this knowledge-base in every Twitterbot that was constructed. This last requirement – essentially forcing students to use a common knowledge resource – was perhaps the most onerous for organizers and students alike, but also the most important, insofar as it gave students a very real sense of the difference between merely generative and creatively generative systems.

The resource in question was manually constructed by PROSECCO coordinator Tony Veale to ensure its quality. Called the *NOC List* (for *Non-Official Character list*), the resource provided approx. 30 thousand semantic triples on the topic of famous figures from popular culture, whether contemporary or historical, real or fictional. The resource provides a sufficiently rich model of 800+ figures (ranging from Julius Caesar to Darth Vader) to support knowledge-based systems for metaphor/simile generation, conceptual blending, and simple story generation. Student groups used the NOC List to build knowledge-based 'bots that, e.g. generated knock-knock jokes (containing in-jokes about famous people), riddles (the *RiddlerBot*, which was the topic of an ICCC-2015 paper by the team in question), game-like interactions and short story-like dialogues. The NOC List continues to be a substantial resource that can be used in future CC events, or as the basis of future CC coursework wherever CC courses are taught. The resource is freely available to the CC community via PROSECCO's Github:

https://github.com/prosecconetwork/

We were pleased with the execution and outcomes of the code-camp, feeling the students involved showed an obvious and strengthened enthusiasm for CC. Our positivity was to be reflected in the decision to host another code-camp in 2016 rather than the initially scheduled Summer School of hybrid of School and Camp as outlined in the DoW. We felt, however, there were lessons to be learnt from the first code-camp that could be put to good use in organizing the second, and sought out the advice of elder lemon **Graeme Ritchie** – a team mentor and PROSECCO advisory board member – on what changes might next be made:

> Here are some comments on the recent code camp in Coimbra. (I'll also put shorter notes on this in my contribution to the PROSECCO Advisory Board report.)
>
> The code camp seemed to be broadly very successful, particularly with regard to the organizational aspects. There were two aspects of the academic activities that I think would merit some reconsideration if another such event is planned: the emphasis on the goals set to the participants, and the supervisory role of mentors.
>
> It is all too easy for students to perceive CC as being primarily about clever hacking, without regard for more theoretical, abstract, or principled matters. Also, some of the students attending a code camp might not have a background in CC as we understand it, particularly those aspects which are less concerned with cute programming. Although Tony [Veale]'s introductory talk mentioned many of the creativity issues, it was tied very closely to the Twitterbot topic, and the tone had already been set by starting with a talk about software tools, followed by Mike [Cook]'s talk specifically on the twitterbot artifacts. Instead, the talks could have been in the reverse of the order used, and the first talk could have placed even more emphasis on what the aims of CC projects might be. Also, having talks on the mornings of the

first two days, and group work sessions on the first two afternoons, might have helped the students to get into the ideas first, before the actual coding took over. The first afternoon group session, for example, could have been explicitly about ideas and not about coding details.

Some of the steering of the students could have been done by the mentors, which brings me to my second point. The remit given to the mentors was to leave the students alone as much as possible. This raises the question of whether it was worthwhile employing such an experienced and knowledgeable team of mentors. There was an opportunity for the students to learn working practices and methods from the mentors, and to pick up a better idea of what CC is about, but the "*hands off*" remit worked against this. One participant (not from my group) said afterwards that he would have preferred if mentors could have taken on a "project manager" role. On a personal note, I was uncomfortable with this lack of involvement. Having spent a lot of time over the years supervising students at various levels, I think I could have contributed much more to helping the students to organize their work and to focus on the more important aspects. However, I felt inhibited by the general advice to let the students make their own way.

As I said at the start, the event as a whole seemed to be successful. However, I think it could have been even more effective, and the above comments are intended as constructive suggestions for future code camps.

Thanks again to the organizers for making this such an enjoyable and smooth-running event, and thank you very much for inviting me to it.

As both a member of PROSECCO's international advisory board and a team mentor at the code-camp, Graeme's feedback and advice were deemed especially valuable. We thus describe in the next section the extent to which this advice shaped our organization of the second PROSECCO code-camp in 2016.

4. The Second International PROSECCO Code-Camp

After the success of the first code-camp held in Coimbra, Portugal in 2015, PROSECCO organized a second international code-camp on Computational Creativity in the city of Antwerp, Belgium in April 2016. The local organizers of the camp were Lucas Nijs of the Sint Lucas School of Art and Walter Daelemans of the *CLiPs* Natural Language Processing group of the University of Antwerp.

As in the previous event, scholarships were made available for students to participate in the camp. All students were expected to have reasonable coding abilities in a language such as Java or Python, though some students participated in a non-technical capacity, by bringing their knowledge of aesthetics or literature to bear on the development work of their team mates. Substantial data/knowledge resources were provided to students so as to undertake the main coding challenge of the camp. This challenge concerned the development of automated story generation systems to produce original stories at the level of plot (the *fabula*) and surface rendering (idiomatic English). These resources built-upon and expanded the knowledge sources created for the first PROSECCO code-camp in Coimbra in 2015. They included:

The NOC list: The Non-Official Characterization List.

This is a knowledge-base providing detailed information on many facets of over 800 popular figures from pop-culture, fiction and history. Originally constructed for the 2015 code-camp in Coimbra, the NOC provides a rich source of biographical and *mise-en-scene* information for creating interesting and credible characters in a story context.

Scéalextric: A Path-building Story Generation System

This is a comprehensive knowledge-base of action triples and logical connectives for building plots as interconnected action sequences. The name derives from the Irish word for "Story" ("**Scéal**") and the brand name of a popular slot-car racing system, "**Scalextric**", since plots are assembled from pre-fabricated segments of different shapes, sizes and twistiness in much the same way Scalextric tracks are constructed.

As in the previous code-camp, participants formed groups to build their own autonomous software systems for generating original and interesting stories. Much of the camp thus involved intense coding in groups, but frequent brainstorming sessions and tutorials were also scheduled, in addition to a lively social programme that allowed participants to mingle and establish new connections.

Student participants were formed into teams of four people each, using the tiered approach of the first camp that allowed us to distribute technical abilities evenly amongst the groups. Thus, strong programmers were distributed amongst the teams so that every team had at least one experienced programmer to anchor the development work. Teams in the first code-camp were named after characters from Hergé's Tintin, perhaps making Belgium the inevitable choice for this second camp. In this instance, each team was now named after a different Bond Villain, in the hope that participants would be inspired to seek out their inner wicked genius. (When given the choice, participants voted to reject team-names based on the Seven Dwarves and Santa's Reindeer). When constituted using the tiered approach, the eight teams were:

- **Blofeld**
- **Drax**
- **LeChifre**
- **Jaws**
- **OddJob**
- **Stromberg**
- **Dr. No**
- **Goldfinger**

Each team was strongly encouraged to live-blog during their design and development work, and their postings can be sampled on our post-PROSECCO website/blog http://BestOfBotWorlds.com. Teams were additionally encouraged to write a closing summary of their approach to story telling, and their posts – which we include in this deliverable for completeness – show the diversity of the topics that were tackled during the camp. For instance, teams explored the role of Jungian archetypes in storytelling, while others explored the role of sentiment and dramatic tension, and others still playfully tapped the humorous potential of national stereotypes.

4.1. Scale of the Camp

Approximately 40 participants attended the camp, and received a capped grant of 750 euros each to defray their travel and accommodation costs. No registration fee was levied. A small number of participants had also attended the previous code-camp in Coimbra, and these experienced hands were welcomed as a source of institutional memory for the other participants to call upon. We hope that this trend will continue; rather as in the way that one makes wine or yogurt, it is good to preserve culture by preserving some of the substance of previous iterations.

4.2. Structure of the Knowledge-Resources Provided

All of the knowledge given to camp participants – both in the NOC list and in the Scéalextric distributions are stored as a collection of semantic triples. Now, one can use XML, RDF, RDFS or OWL to store a collection of semantic triples, but the core of the triple stays the same regardless of the formalism we use to encode it. So why bother with a complex syntactic sugar for the sake of formalistic appearances? When it comes to maintaining and editing and sharing a large body of semantic triples the most flexible format is a spreadsheet. As mundane as it may sound, a spreadsheet is perfect for this kind of representation work.

Every cell, representing as it does a value-containing intersection of a named column and a named row, represents a single triple. We can share spreadsheets easily (and post them on the Web as Google docs for communal editing) and cut-and-paste relevant parts with abandon. Every piece of information in the NOC or in Scéalextric is to be found in a spreadsheet. Each sheet can be saved as a *tab-separated-values* (TSV) text file for easy processing by Java and Python programs, and developers are encouraged to add new columns and rows to each one as the need or opportunity arises, or to create new spreadsheets with complementary dimensions of knowledge. For example, consider a semantic relationship *P(X, Y)* where *P* is a predicate that holds between X and Y. We can represent a collection of triples of the form P(X, Y) in a spreadsheet with a column labeled P, a row whose first (and key) value is X, and a cell at the intersection of this row and the column labeled P that contains the value Y. This cell may contain multiple values Y_1, Y_2 ... Y_n, each separated by commas, so this cell would represent a group of predications $P(X_i, Y_1)$, $P(X_i, Y_2)$ ... $P(X_i, Y_n)$

The NOC distribution contains a variety of spreadsheets, each one a triple store that stores its semantic triples in this fashion. The following are the main spreadsheets in the distribution (the TSV Lists directory contains text file equivalents in which the spreadsheets have been saved in tab-separated-values [TSV] format). The resources are free for use in a CC research context, and subsequent modification and enlargement is encouraged.

The NOC List.xlsx

This spreadsheet encodes the central triple-store in which characters are defined. Other triple-stores/spreadsheets in the NOC distribution provide further detail on the concepts used in this central triple-store. The first and key column provides the names of the characters, one row per character. Other columns then contain attributes of each character and related ideas, such as Gender, Political leaning, Address, Spouse, Opponent, Weapon, Vehicle, Typical Activities, Typical Clothing, Group Affiliation, Domain, Fictive status (real or fictional?), Creator (if fictional), Taxonomic Category, Positive Talking Points and Negative Talking Points

Vehicle fleet.xlsx

This triple-store provides useful detail on the vehicles associated with characters in the central NOC triple-store above.

Weapon arsenal.xlsx

This triple-store provides useful detail on the weapons associated with characters in the central NOC triple-store above.

Typical Activities.xlsx

This triple-store provides useful detail on the activities associated with characters in the central NOC triple-store above.

Clothing line.xlsx

This triple-store provides useful detail on the clothing items associated with characters in the central NOC triple-store above.

Creations.xlsx

This triple-store provides useful detail on the creations associated with characters in the central NOC triple-store above. For instance, Seth McFarlane is the creator of Family Guy and this triple-store indicates that Family Guy is a TV show

Fictional Worlds.xlsx

Family Guy is also a fictional world in which a variety of characters live and interact. This triple-store characterizes the fictional worlds associated with the characters of the main NOC triple-store.

Domains.xlsx

This triple-store provides useful detail on the genres and domains associated with characters in the central NOC triple-store above.

Category hierarchy.xlsx

This triple-store provides useful detail on the taxonomic categories associated with characters in the central NOC triple-store above. In effect, it clusters these categories into broader categories to form a semantic hierarchy or IS-A hierarchy in the old-fashioned AI sense.

The Scéalextric distribution likewise contains a variety of spreadsheets, each one a triple store that stores its semantic triples in this fashion. The following are the main files in the distribution:

Script midpoints.xlsx

This triple store contains the action triples which are described above. Each row contains at least one triple, where the first column holds the first action in the triple, the second column holds the second action, and the third column holds the third action. Since each cell may contain multiple comma-separated values, then a single row may in fact contain multiple action triples. If column 1 holds N_1 actions, column 2 holds N_2 actions, and column 3 holds N_3 actions, then the row encodes N_1 by N_2 by N_3 different action triples

Idiomatic actions.xlsx

This triple maps actions (verbs) onto one or more idiomatic renderings. The action is simply a verb, and it is assumed that the protagonist (subject) is A and the antagonist (object) is B. The idiomatic rendering contains A and B as placeholders, to be replaced in the generation process with the actual protagonist and antagonist.

Initial bookend actions.xlsx

This triple store provides a scene-establishing preamble for every action verb, so that a story beginning with that action/verb might be prefaced with this preamble.

Closing bookend actions.xlsx

This triple store provides a moralistic epilogue for every action verb, so that a story ending with that action/verb might be concluded with this "message" for the reader.

Action pairs.xlsx

This triple store provides a logical link between actions that may follow each other in a story (as derived from the structure of the action triples in ***script midpoints.xlsx***)

4.3. Lessons Learnt from the First Code-Camp

As noted in the feedback offered by advisory-board member Graeme Ritchie after the first PROSECCO code-camp, some streamlining and improvements are possible and desirable. We now outline the changes made to the organization of the second code-camp in line with his recommendations:

1. ***An unhealthy focus on Hacking and Software tricks***

   Graeme notes that the first code-camp was strongly task-focused, to avoid wasting precious time in the tight schedule. By focusing heavily on the specific topic (Twitterbots) from the outset, some participants may have learned the wrong lesson, namely that CC is a matter of clever engineering tricks rather than computing philosophy in action. In the second code-camp we remained equally task-focused, again due to the limited time-frame, but initial talks were aimed at a higher-level, far above the level of programming methodology and clever "hacks." Introductory talks focused on the issues of what constitute a compelling narrative, and were followed by a discussion of representations rather than processes.

2. ***Ideas first, implementation later***

   Implementation is what puts the "Code" in a "code"-camp and so, at a certain level, implementation is what an event link this is all about. Yet implementation is much more than a matter of writing code, and good program structure is fundamentally tied to good domain insights. In an OOP language such as Java or Python one is, after all, striving to build a model of some aspect of human experience. By pushing ideas to the forefront, and considerations of programming language to one side, participants debated the substance of stories and characterization before they debated the substance of their code. The result of this change can be seen in the topics pursued by the various groups, who for the most part strove to develop interesting ideas in narratology rather than clever programming tricks.

3. ***Ill-defined Role for Mentors***

   The first code-camp was conceived as a follow-on and a replacement for the traditional Summer or Autumn school. CC researchers that would have otherwise been called upon in a school setting to deliver lectures were now called upon to mentor small groups of students in the development of their own CC systems. But mentors are *not* supervisors, and should not direct the work of a team as a PhD supervisor directs the work of a group of PhD students. This relatively hands-off role proved to be problematic for some mentors. In fact, this role of *mentor* was surplus to our requirements from the beginning, and at the second code-camp we had little need of so many chefs to stir the broth. Instead, wisdom was in supply in the form of experienced participants who had taken part in the first code-camp and could share their knowledge and best practices with the other students.

4. ***Improved Knowledge-Sharing***

   Participants often value the opinions of peers over those of their esteemed supervisors and mentors. To promote peer-review, each team was required to live-blog their work on the official website http://BestOfBotWorlds.com. This forces frequent periods of reflection on team members, promotes the sharing of ideas, perspectives and solutions amongst peers, and allows us to preserve a record of their thought processes that will also inform our future code camps.

In these respects the second code-camp provides an excellent model for future code-camp endeavors in CC, with the caveats we outline next.

4.4. Lessons Learnt from the Second Code-Camp

Though benefiting greatly from the lessons of the first code-camp, the second camp has its own lessons for the organization of future CC events. In both camps we placed a strong emphasis on the role of knowledge-representation. While it is tempting to view the Web as the font of all knowledge, and to subsist on noisy but large-scale Web-derived information when high-quality curated symbolic knowledge is in short supply, the latter is almost always superior to the former (though a mutually beneficial hybrid is even more preferable if the task allows). The second code-camp provided participants with an abundance of high-quality symbolic resources. Indeed, it may be fair to say that PROSECCO, a coordination action and not a research project, has provided more substantial resources for future CC development than any other CC-themed project that has been funded to do precisely this kind of research. Yet when presented in the tight time-frame of a code-camp, an abundance of resources can easily become an *over*-abundance, and promote paralysis of thought when shrewd analysis leading to concrete action is the ideal outcome. For the most part we were successful in fighting this tendency toward analysis-paralysis in the second code-camp, but the lesson for future camps is clear: well-defined tasks, coupled with appropriately-sized resources, make for the best group challenges. Though it is better for a resource to be too big rather than too small, we shall have to cut our cloth more carefully in future events if we are to avoid overwhelming participants – initially at least – with so much useful knowledge that it ceases to be useful.

5. A Message in a Bottle: One-Day CC Tutorials

PROSECCO has organized two one-day tutorials at the ICCC conference over its lifetime, as a means of offering a small-scale introduction to the major themes and philosophies of CC research. These exploit the organizational structure of the host conference, and appeal to an audience that is mostly captive (already planning to attend the conference) or outside the field but looking for a convenient way in.

A one-day PROSECCO-funded tutorial on Computational Creativity was offered at ICCC-2014 in Ljubljana, Slovenia, which was free to all attendees, registered or otherwise. An audience of 60+ attendees attended the tutorial, which was jointly delivered by PROSECCO partners Geraint Wiggins and Tony Veale. The former provided an introduction to the formal study of CC, while the latter introducing the attendees to the CC potential of Twitterbots.

A further one-day tutorial on Computational Creativity was later organized at the 7th ICCC, that is ICCC-2016 in Paris, France. The event provided an introductory overview to all those wanting to begin exploring the field, particularly students starting a research program in a CC-related area. The program consisted of three introductory lectures, two by PROSECCO partners and one by a member of our advisory board:

Characterizing Computational Creativity, by **Geraint A. Wiggins**,

Data Mining and Machine Learning in Computational Creativity, by **Hannu Toivonen**

Assessing the Performance of a Creative System, by **Graeme Ritchie**.

The slides of the lectures have been made available at the PROSECCO project website. Twenty three students registered online for the Tutorial, with nineteen actually attending on the day. Eleven attendees were young researchers that received PROSECCO scholarship grants (given after a selection from 17 received applications). In the decision process, priority was given to European applicants and to those who had not previously received any funding from the project. All the attendees to the tutorial were also registered in the main sessions of the conference and thus had the opportunity to immerse themselves in Computational Creativity research for the week-long duration of the conference.

6. A Map for a Post-funding PROSECCO Future

All three of PROSECCO's tent-pole educational events were successful in their own ways, and in each case one can fairly conclude that the major goals of the project were met. Nonetheless, these big-ticket events come with an equally big price-tag, so it is important to derive economical lessons from our PROSECCO experiences wherever possible.

The main lesson is that we cannot realistically continue to *pay-for-play* when it comes to participation in our educational events. Miniature summer-schools can be organized as one-day tutorials at conferences, and our code-camps must also stand on their own financially when PROSECCO is no longer there to fund them. This financial independence can be achieved in a number of ways. First, we can run our CC courses, tutorials and camps under the auspices of other events with their own built-in audiences. For instance, Tony Veale delivered a week-long course on Computational Creativity as part of the Interdisciplinary College (IK-2016) annual Spring school in the Ruhr Valley, Germany, and has been invited to give a follow-up course at IK-2017. Second, just as with our CC tutorials (of which PROSECCO has funded two: the first in 2014, the second in 2016, in each instance at the ICCC conference), we can co-locate our future code-camps with prominent conferences or workshops, to also appeal to an already present audience. Though attendees at high-profile conferences are already in the habit of paying to attend tutorials, it does not seem realistic to charge for attendance at a code-camp, yet it should not be necessary to pay for attendance either. The third option complements the first two: we can arrange tutorials and code-camps at specific educational institutions for the benefit of students and staff at those institutions. To the extent that the institution's cohorts are already internationalized, we can still attract an international audience even when restricting attendance to local participants. To the extent that the cohorts are *not* very international in make-up, we can reach an international audience by holding multiple camps in different countries as the opportunities arise. By tying tutorials and camps to specific educational institutions and cohorts, we may also find it easier to attach ECTS credits to participation.

This is the route we are currently charting in a post-PROSECCO funding world. On September 12-17, PROSECCO coordinator Tony Veale organized a 1-week hybrid lecture-series / code-camp at the university of Saarbrucken, for students of the university and neighboring institutions (e.g. DFKI). Over the course of 5 days, with lectures delivered in the mornings and hands-on practical sessions arranged in the afternoons, students explored both the promise and the reality of CC research. Future PROSECCO-*lite* events may be arranged and delivered in a similar fashion. PROSECCO has given us the means to create the resources and the materials to make this lightweight model a possibility. It is now up to us to make it a success.

Appendix I

Plan of the Saarbrucken Code-Camp-*lite*

The following announcement from the local organizer, Josef Van Genabith of Saarbrucken University and DFKI, outlines the form and the content of the event.

Dear all,

UdS FR4.6 and DFKI are delighted to invite you to a one week course with lectures and labs on

Computational Creativity

given by **Dr. Tony Veale**, University College Dublin, Ireland

12.09-16.09.2016, Konferenzraum 1.20, FR4.6, Building A2.2, Saarland University

Morning lectures 10:00-11:30, Afternoon labs 15:00-16:30

Places are limited. To register for the course please contact Lucia and Corinna **mlt-sek@dfki.de** by **Fri. 09.09.2016 EoB**.

Course overview and requirements:

1. Tweet Dreams Are Made Of This

Twitter is an ideal paradigm for the development of public-facing AI. The design and construction of autonomous AI Twitterbots will be our focus in this lecture.

2. Computational Combinatorial Creativity

Creativity was long conceived as Generation Ex Nihilo, that is, the making of something out of nothing. Yet human creativity is not divine, and every novel output is a product of pre-existing inputs. This lecture will explore the mechanics of Combinatorial Creativity, and show how these mechanics provide a solid foundation for non-human Computational Creativity.

3. Divergent Thinking on the Web

This lecture conducts a computational exploration of the differences between Divergent and Convergent Thinking, and of how Divergence can be supported by Creative Web Services

4. Automated Story Telling

Humans are the story-telling species. We are defined by our narratives. This lecture explores the representations and processes needed to enable original story-generation by machine.

Requirements:

Participants are urged to check-out the blog **BestOfBotWorlds.com** in advance of the course, and to **register for an account with the site**. Relevant text book materials are available at **RobotComix.com**

Large scale symbolic knowledge resources will be provided for the afternoon lab sessions. Some Java code will be provided as a primer. Participants may use Python or Ruby but the lecturer will use Java!

Best regards,

Josef van Genabith

Appendix II

Group Outcomes in the Second Code-Camp

As noted earlier, each group at the second code-camp was required to live-blog their work so as to facilitate peer-review and cross-group sharing of knowledge. For completeness we include here the final summaries posted by each group to the camp blog. Note that the blog is intended to remain active after the life-span of PROSECCO, and will be used in subsequent code-camps where future participants can learn from the experiences of those who have gone before them. What follows is the actual and unedited text of each group's posting on http://BestOfBotWorlds.com

Team “Blofeld”

How can we combine multimedia data sources such as Google Images to creatively paint pictures for a story that has been generated by a computer? In the Blofeld group at the computational creativity code camp 2015, we got inspired by this question. Members of Blofeld group are:

1- Philipp Wicke (pwicke@uni-osnabrueck.de)

2- Robert Pfeiffer (bob@blubberquark.de)

3- Kasia Bigaj (k.bigaj@student.uj.edu.pl)

4- Hamid Reza Ghaeini (Ghaeini@acm.org)

Our idea is to have multimedia content by using the Google API. The actual process consists of these four steps:

- Telling a story.
- Extracting most salient words
- Putting pictures (representing the salient words/concepts) on a canvas.
- The story determines the scaling/format/position of the pictures.

In this report our focus is on the first part. First of all, we used the non-official characterization list (NOC) to find two actors of our story by some randomized algorithm which tries to find two related actors at random. This is how our story begins! Then from the Scealextric knowledge base, we chose some idiomatic actions that are somehow related to those two actors. Here is where our question arises: Which methods do we have to apply to retrieve the most salient feature of a story in order to find an apt visual representation i.e. an image via Google? To find a correlation between our idiomatic actions of our actors we proposed to use some deterministic approaches. Then we will finish our story by using the story ending knowledge base. How could we be able to extract the most salient features of a story? They were already provided during the generation of the story. So we could easily use the memory of the story teller algorithm to crawl the internet for a corresponding image. Let's say our story was about James Bond, we would ask the Google API to search for James Bond images. It all worked out pretty nice until the point where we had to ask Google. Unfortunately we could not use the API and went for Imgur which turned out to be a bad idea. The images are not correctly annotated and the outcome for our search queries was pretty bad. All code is available at:

https://github.com/ghaeini/codecamp2016

An example story is:

"Eliot Ness paid well and expected absolute loyalty in return. Rick Deckard finally caught up to Eliot Ness. Rick Deckard finally tracked Eliot Ness down. Thereafter Rick Deckard kept Eliot Ness on a very tight leash indeed; Eliot Ness was never truly free again."

Team “Drax”

Team: Peter Hammar, Samarpan Rai, Alessandro Valitutti, Ben Gruber, Amit Kothiyal, and Paul Bodily

Our approach to the problem of generative creative narrative was to focus on the specific problem of generating narrative which follows a specific plot flow. For example, we may want to generate a narrative that follows the flow of a hero cycle in which the protagonist starts in a fairy neutral position, must pass through a low point, and then ultimately arrives in a high point. This emotional progression extends to the reader, creating a range of feelings that are what create an enjoyable, engaging narrative.

New Annotation of Data

To achieve this goal of creating narratives with plot flow, we needed to first have some idea of the polarity of potential actions which might occur with a protagonist. For example, "Character A loses a loved one" would be a negative action for Character A and "Character A overcomes Character B" would be a positive action for Character A and a negative for Character B. In the dataset given us all actions had one character as the subject and a second character as the object. Thus we labeled each action with the polar impact it would have on Character A and Character B. We used values ranging from -2 to 0 to 2 to denote actions ranging from negative to neutral to positive respectively. These annotations are housed in a new knowledge base which we have labeled "Draxberg's Action Polarities".

We also intended to have some notion of the polarity of the protagonists in our dataset. For example, "Mother Theresa" is generally viewed as a good person or a hero whereas "Saddam Hussein" is generally viewed as a bad person or a villain. We used values ranging from -1 to 0 to 1 to denote characters ranging from negative to neutral (which we tried to avoid where possible) to positive. To this end we added annotations to "Veale's NOC List" which contains a list of characters from which our narratives select its characters.

These two additions to the existing knowledge-base (available upon request) constitute one significant contribution from our research efforts. It should be added that these tasks were undertaken in collaboration with Team Stromberg which served to expedite the annotation process.

Narrative Generation

Given these added resources, our generative system proceeds as follows:

1. Select a plot flow (e.g., for the hero cycle the plot flow might be [0, 1, -2, 2])
2. Select an initial action whose polarity matches the first polarity in the plot flow Example: "preaches"
3. Select a protagonist whose Category makes them a candidate for performing the initial action. Example: "Mother Theresa" (whose category, "religious leader", makes her a potential subject for "preaches")
4. Randomly select an opponent for the protagonist from the set of characters who EITHER A. Shares common either "Positive Talking Points" or "Negative Talking Points" elements with one of the characters listed as the protagonist's

opponents (as per "Veale's The NOC List") AND shares the same Category as one of the protagonist's opponents OR B. Is portrayed by the same actor as portrays one of the characters listed as the protagonist's opponents Example: "Saddam Hussein" (who shares common traits and Category with "Lucifer", the latter being listed as an opponent of "Mother Theresa")

5. For each of the polarity values *p* in positions 2 to the end of the plot flow, select a pair of action m, a such that a taken from the "After Midpoint" category from "Veale's script midpoints" matches the following criteria: A. The polarity of action a matches polarity p from the plot flow; B. The action a is a consequence of the previously selected action (as per the value in the "Before Midpoint" category
6. Having selected actions for each polarity in the profile, render each action using the values in "Veale's idiomatic actions" associated with the key matching the selected action. For the first and last actions, use the values in "Veale's initial bookend actions" and "Veale's closing bookend actions" respectively. Also add (with some randomness) appropriate linking prepositions between rendered actions, as found in "Veale's action pairs".
7. For each idiomatic rendering of an action, substitute the name of the protagonist (from step 3) for any appearance of the variable "A" and the name of the opponent (step 4) for "B" with the following variation: A. The first time the name is mentioned use the full "Character" name, every time there after use only the first name of the character (excluding titles such as "Dr.", "Count", etc.); B. Once the "Character" name has been used at least once in an action rendering, for all subsequent actions, use the first name of the character's "Canonical Name"; C. For the final action, use the full "Canonical Name" of both characters in the rendering; D. With some randomness and in accordance with the polarity of the action for each character, add adjectives describing the character before the character name taken from the "Positive Talking Points" and "Negative Talking Points" in "Veale's The NOC List".
8. For each action a, add polarized filler sentences which express detail about the characters with sentences reflecting the same polarity as the action a. Note that these are not actions, merely descriptive sentences. For the opening action, generate descriptive sentences describing each character in detail as taken from the "Positive Talking Points" and "Negative Talking Points" in "Veale's The NOC List" and according to the polarity of the characters.

Descriptive Sentences

The descriptive sentences described in step 8 serve to elongate the narrative, thereby extending the emotion of each action so as to allow the reader to more fully appreciate the emotions before suddenly moving on to the next. These sentences were generated using template sentences that allow for substitution of character- and polarity-specific values for the action to which they are appended. For example, if "Saddam Hussein" had just "apologized" to "Mother Theresa" (an action which has a positive polarity for "Mother Theresa"), then this template may be rendered as

Mother Theresa started to appreciate Saddam Hussein.

Several such templates were created, most of which incorporated various character-specific attributes from "Veale's The NOC List" such as typical clothing items, typical activities, addresses, and character Category, as well as attributes such as gender-specific pronouns or possessive adjectives.

Examples of Baseline Narrative Generation System

Marriage was always in the cards for Hugh Hefner and Derek Zoolander.
But, Derek Zoolander stole Hugh Hefner's thunder.
But, Hugh Hefner imitated Derek Zoolander's style and adopted it as its own.
But, Derek Zoolander heckled Hugh Hefner with cruel jibes.
But, Hugh Hefner hurled insults at Derek Zoolander.
So, Derek Zoolander shut Hugh Hefner up.
So, Hugh Hefner grew to resent Derek Zoolander.

Marriage was always in the cards for Juliet Capulet and Edmund Blackadder.
However, Edmund Blackadder completely deceived Juliet Capulet.
As a reaction to this, Juliet Capulet filed for divorce against Edmund Blackadder.
Nevertheless, Edmund Blackadder furtively followed Juliet Capulet everywhere.
For this reason, Juliet Capulet stabbed Edmund Blackadder.
In spite of this Edmund Blackadder came back to haunt Juliet Capulet.
Juliet Capulet hoped that Edmund Blackadder would be satisfied with the sacrifice and not demand Juliet Capulet's own blood.

Examples with Intelligent Character Selection Added

Dr. Evil had a knack for getting others to do things.
Still Shrek trusted Douglas implicitly.
Yet, Douglas crucially underestimated Shrek.
As a consequence, Shrek took over the position once occupied by Douglas.
Still Douglas looked up to Shrek as a role model.
However, Shrek pulled Douglas's strings.
Would Shrek use his newly consolidated power over Douglas Evil wisely? Or squander it foolishly?

Monica Lewinsky was a pushover for a witty chat-up line.
Monica became utterly smitten with Michelle Obama.
Nevertheless, Michelle pulled Monica's strings.
For this reason, Monica's attitude hardened toward Michelle.
As a consequence, Michelle publically denounced Monica.
For this reason, Monica sought forgiveness from Michelle.
Monica Lewinsky's family would now seek revenge in a never-ending cycle of vengeance.

Harry Potter needed a top-class trainer to reach the top.
In spite of this Harry failed to properly reward Darth Maul.
The effect of this was that Darth unceremoniously dumped Harry.
As a reaction to this, Harry begged Darth's forgiveness.
For this reason, "Get back here right now" said Darth to Harry.
For this reason, Harry sought forgiveness from Darth.
Harry Potter was never again seen in these parts.

Mark Zuckerberg looked everywhere for inspiration.
The effect of this was that Mark worshipped the ground beneath Steve Jobs.

For this reason, Steve filled Mark's head with doctrinaire ideas.
However, Mark misrepresented Steve's intentions.
As a consequence, Steve had Mark silenced.
As a reaction to this, Mark obeyed Steve's orders.
That's what happens when you give your trust too easily.

Examples with Plot Flow Profile Added

Profile: [2,1,0,-1]
Marriage was always in the cards for Barack Obama and Michelle Obama.
Michelle loved everything about Barack.
As a consequence, Barack showed true loyalty to Michelle.
Yet, Michelle pushed Barack too far.
As a consequence, Barack lost all faith in hectoring Michelle.
The effect of this was that Michelle publically denounced Barack.
Barack Obama pledged to never seek out Michelle Obama's company again.

Profile: [-1,0,1,2]
Sarah Palin and Osama Bin Laden were from different sides of the political divide.
Sarah delivered a crushing defeat to Osama.
The effect of this was that Osama campaigned vigorously for Sarah.
Nevertheless, "You're fired" shouted Sarah at Osama.
For this reason, Osama furtively spied on Sarah.
Tough-talking Sarah appealed to Osama's sense of wonder.
Will romance flourish for Osama Bin-Laden and Sarah Palin? does a happy marriage beckon for Osama Bin-Laden and Sarah Palin? will Osama Bin-Laden and Sarah Palin be star-crossed lovers or will love prevail?

Profile: [0,1,-1,2]
Bruce Wayne had money and Dr. Evil needed money; it was a match made in heaven.
Yet, Douglas totally took advantage of Bruce.
The effect of this was that Bruce decided to sue Douglas.
Douglas seduced and enthralled Bruce.
Bruce waved the white flag to shrewd Douglas.
Douglas delivered a humiliating lecture to hedonistic Bruce.
Douglas Evil looked down from his throne and laughed; it was a hollow laugh.

Profile: [0,1,-1,2]
Sarah Palin and Osama Bin Laden were from different sides of the political divide.
However, Sarah delivered a humiliating lecture to Osama. Osama spent days insipidly plotting terrorist outrages.
In reaction, Osama hated everything about Sarah. Osama felt his ramming with a hi-jacked 747 was looking quite weak right now.
Yet, Sarah paid Osama to do what it asked. Osama often remarked, "My, what the moot Conservative you've become, Sarah!"
Osama showed true loyalty to Sarah. Sarah started to notice Osama.
For this reason, Irrepressible Sarah rewarded bold Osama amply. Sarah was beginning to look quite strong compared to Osama. Will romance flourish for Osama Bin-Laden and Sarah Palin? does a happy marriage beckon for Osama Bin-Laden and Sarah Palin? will Osama Bin-Laden and Sarah Palin be star-crossed lovers or will love prevail?

Challenges

Several challenges exist to the approach we've taken. The biggest challenge derives from the need for more data. Even with as large a knowledge base as we are given, finding a sequence of satisfactory actions (step 5) that meet the constraints of our plot flow is very difficult. Often the sequence progresses to a place where no satisfactory actions transitions are available. Our system overcomes this by generating and testing for this exception. Our research suggests that even for the most frequently successful plot flows, only about 1 in 20 generations succeeds in finding a full sequence of satisfactory actions.

Ideally all actions would not only adhere to the polarity constraints, but also to constraints limiting actions to those which agree with the characters' Category values. This would be overly prohibitive in the system's current state, for which we did not implement this latter constraint. One way to overcome this would be to leverage information in "Veale's Category Hierarchy" such that if no actions were available due to a character's Category, a higher level of the hierarchy might be used to find a more generic Category for which there are satisfactory actions.

The final challenge we faced was that we found some subjectivity existed in the annotations we made for characters and actions. For example, our expectation was that by choosing an opponent similar to those listed as the protagonist's opponents we would get two characters with opposite character polarities. This was often not the case however, and enforcing it often resulted in no choices for an opponent. As such it occasionally happens that two characters of like polarity are chosen as characters for the narrative (e.g., "Saddam Hussein", who is enemies with "George Bush", and "Osama Bin Laden", who is apparently similar to "George Bush"). This may or may not be undesirable, but was not what we expected. One approach to solving this might be to use social network analysis of the reader to cater the character polarities to the preferences of the reader.

There were also many cases where the idiomatic rendering of actions slightly altered the meaning of the action, making the transitions between actions awkward. This is an example of the problems resulting from using a knowledge base: your sentences can only be as good as the accuracy of the symbol representations in the knowledge base. Further improvement to the knowledge base by refining idioms and splitting actions whose meaning varies by context is needed.

Implementation details

The program was developed using a combination of the Java (in Eclipse) and Python programming languages. We used Git to collaborate our programming efforts; Trello to collaborate our team management; and *GroupSpaces* to for easy communication with team members. Progress of our program was periodically reported on the Code Camp Blog, http://bestofbotworlds.com. We also created a website to demo narratives created by our system, http://greenspray.koding.io/cck2016-drax/demo.

One of the greatest benefits of the camp and the project was learning how to work together to leverage the several strengths of each member. We were happy to find that each member contributed significantly in terms of contributing both intellectually and programmatically to the final version of our system.

Conclusion and Future Work

We were pleased with the results of our narrative system. We found that in several of our stories the plot flow was well-reflected in the sequential actions, leading us overall to conclude that our efforts to create a narrative-generating system that follows a particular plot flow had been largely successful. In the future we'd like to implement more efficient ways of discovering meaningful good plot flows and finding action sequences that match them (without resorting to a generate-and-test methodology).

We feel that the results of our project and many others at PROSECCO 2016 provide a glimpse of A) how collaborations effective facilitate research that is greater than the sum of individual members and B) how computer systems can be implemented to achieve specific goals in narrative story generation. Narration places a pivotal role in communication universally. Our society stands to benefit immensely from machines equipped with the ability to communicate in this medium.

Acknowledgements

The Drax team would like to acknowledge Tony Veale for providing the initial knowledge base, his constructive criticism during brainstorming sessions, and help developing data-accessory methods. We would also like to thank Team Stromberg for their assistance in annotating the data as outlined above.

Team “Le Chifre”

Members of Le Chifre team:

Marco Scirea [msci@itu.dk]
Samuel Doogan [samuel.doogan@ucdconnect.ie]
Audrey Michels [a88354@aup.edu]
Alexander Myronov [alexander.myronov@gmail.com]

The core idea of our project was to create something more interactive than pure text generation. Given our shared interests, we quickly converged on the idea of creating a game-ish "choose your own adventure" text game. This could also be seen as a way of enabling the user to explore the NOC list in an entertaining way.

At first we wanted to be able to create a Hero's Journey type of structure (see our previous post) and use various archetypes to structure interactions between characters. To this end we chose four archetypes (the Hero, the Mentor, the Trickster and the Shadow) and had to find a way to annotate the character of the NOC list with which archetypes they mostly reflect. We identified some of the main characteristics of these archetypes and programmatically annotated the list with them (see previous post).

Sadly during the codecamp we didn't have time to implement all what we set out for. The final product is a system that allows you to explore the first part of the journey: the interaction between the Hero and the Mentor. The user is free to make the characters interact with each other (in logical ways, according to Scealextric) and should make them reach the point where the Hero is inspired by the Mentor to embark on the journey to defeat the Shadow. The characters are chosen according to our categorization and the settings and interactions are defined accordingly to the characters that are interacting. (see previous post for example)

Upon beginning the implementation, we quickly realized that certain action triples presented to the user were unsuitable for the given archetype they were interacting with. For example, when following the "Hero's Journey" template, one of the first and central plot devices involves having the Mentor archetype encourage and inspire the Hero to leave their home and set out on a grand adventure. Naturally this means that interactions involving harming or insulting the mentor would not be conducive to the advancement of the plot. From using the program ourselves, we found that on a regular basis the suggested actions would inevitably lead to negative outcomes which did not adhere to the "Hero's Journey" template at all. One possible solution, and one we tried to implement (but as previously mentioned did not have time to fully complete) was to annotate each action triple with a measure of sentiment polarity using a sentiment analysis toolkit. Each action was analysed by taking their associated idiomatic representations and given a polarity score. From these scores, the program would choose the most relevant actions to present to the user. In the case of the Mentor, these would be ones which have a positive outcome for the Protagonist and a neutral to good outcome for the mentor. In this manner, only positive actions were supposed to be chosen in order to encourage the user to arrive at the next logical step in the "Hero's Journey". However, this approach also posed a number of challenges. Firstly, the seemingly primitive nature of automated polarity scoring led to certain incoherent action triples being chosen. To tackle this, the team decided to use a manually annotated polarity score kindly provided by another team. The other major challenge lied in the data set itself. It seemed that the majority of action triples inevitably led to disastrous and negative outcomes. This makes sense when trying to provide conflict and drama, along with climaxes and crises, all of which are essential in a good story. However, for our purposes, we needed more benevolent action triples which suited the interactions between two “good” characters. The lack of such action triples meant that very few options could be given to the user, and this in turn limits both the scope of certain sections of the story, along with damaging the immersiveness of the “choose your own adventure” mechanic.

To conclude we are pretty satisfied with the system achieved even if it’s still very limited: it creates fun stories and interactions that are interesting to read. Better still it is challenging, as it forces the user to imagine ahead how the characters will react to actions they do, which is very seldom intuitive and banal.

Team “Jaws”

Wheewhh, it has sure been a busy few days here in Antwerpen on a computational creativity code-camp for building systems that automatically tell stories! Building a system focusing on computational creativity in a few days is a demanding task, but we (Jaws) think that groups managed to do pretty solid proof of concepts across the board. One thing that caught our attention especially was that several groups did collaboration on making the knowledge handed to us more suitable for their needs.

We, the group Jaws, started from the metaphorical point of view. We were interested in how we could use metaphors to map characters to other characters via their properties. However, what we ended up doing was completely another story… ahem.. Bad jokes aside, we may have bitten off more than we could chew with our initial idea. Making metaphors is a difficult task in its own right, and we still would have needed to create the actual stories based on the metaphors!

The system that was actually implemented on the code-camp follows mainly the generate and test approach. We start by generating a bunch of plot lines that in essence are sequences of actions, paired with characters that are taking the actions. Also, we select a location where the plot is taking the place. Then, we evaluate each generated plot (or story skeleton, the fabula) based on how well the characters fit together, etc., and select the plot line which we evaluated to be the best. And realize the fabula in natural language by using idiomatic representations of the underlying actions.

To generate a single plot, we start by selecting a valid midpoint from the given knowledge base (consider the midpoint to be the main twist in a story). We then generate all possible paths from the list of story beginnings to the midpoint and from the midpoint to the story endings. For this we use a directed graph built from the action pairs given to us. Each action is a node in the graph, and edges denote that actions can be taken consecutively. To get the paths, we use a shortest path algorithm from source node to target node and simply discard all nodes that don't have a valid path to the node.

After we have a sequence of actions, we need to select the characters that are suitable for this particular plot. For this, we use the exemplars of the selected midpoint. We have mapped each character in the NOC list to a set of exemplars. We have accomplished this by utilizing MetaphorMagnet to retrieve the typical properties of each character and exemplar. A detailed description of the approach can be read from another blog post. With the exemplar mapping, we have the possible character combinations for each midpoint and can select the most suitable one.

Having the sequence of the actions, and the characters we still need to realize the actual sentences for the story. For this we carried out some fairly simple sentence planning, linearization, as well as rudimentary surface realization (e.g. managing tenses, person reference, various forms of agreement, and the like). Essentially we made sure that we made use of established natural language generation techniques as much as possible in the available time.

We also wanted to develop our narrative by describing the characters and locations around which the story revolves. What we ended up doing was a set of templates for character and location descriptions, and selected at random the template to be used. However, we soon encountered a problem: setting description templates relies heavily on the mood (or atmosphere) of the location. So, we annotated the given locations with their basic mood: positive, neutral or negative. This allowed us to make more precise templates based on the mood of the location.

Of course, a lot of our implementation ended up half-assed as we ran out of time. However, we believe that we have an intriguing start for generating stories that have interesting character combinations, viable sequences of actions, and more developed narrative using setting and character descriptions that add flavor to the story.

The resources and the code generated during the code camp can be found from github.

Team “OddJob”

An answer set programming approach towards story generation

Team **OddJob** comprises:

Thomas Smith
Zoe O'Shea
Matthew Thompson
Pantelis P. Analytis
Hanyang Chen
Prosecco Code Camp Antwerp

The goal of our system was to create stories using Answer Set Programming, and by composing them out of tropes.

Answer Set Programming (ASP) is a form of declarative (logic) programming, similar to Prolog, where a program consists of rules that evaluate to true or false. Using this method, the programmer is able to generate stories by describing the story itself, rather than how to generate it step-by-step. The system then searches all possible answers, with the output being constrained by the rules that form the story description.

Tropes are recurring themes or motifs in a story, and each trope has a list of associated roles. For example, the "Hero's Journey" trope contains a Hero, a Villain and a Dispatcher (to send the Hero off on their quest). The idea is that a story contains many tropes at once, which describe the structure, themes and characters in it. Constructing a narrative using tropes as components therefore gives the author some control over its structure.

To generate our stories, we take characters from the NOC list based on the roles they have in a trope, and feed them into an ASP solver to generate a story. This is all done through a web-based UI.

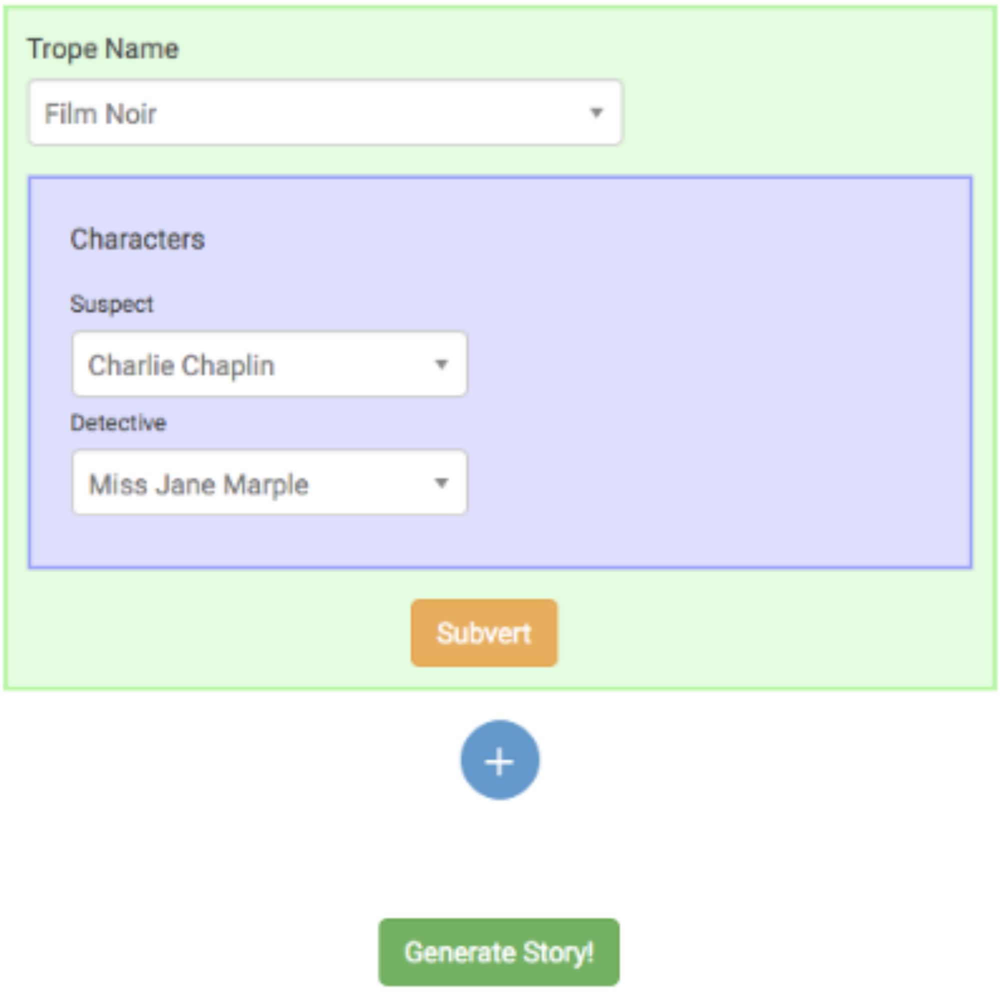
Tropes

Based on the information on the TV Tropes website and the Periodic Table of Storytelling, we added trope information to characters in the NOC list. We did this by first selecting categories of tropes that seemed most within our time range to implement and those most likely to produce successful results. These included: Structure, Setting, Story Modifiers, Plot Devices, Archetypes, Meta-tropes and Fandom Reactions. A new document was created and the selected storytelling tropes were listed along with their popularity in 'kilowicks' (thousands of links to their pages within the wiki). The goal of this was to give a reasonable sense of the precedence of a particular trope - and therefore its relevance to the project. The various classifications of tropes were assigned to members of OddJob; using the 'Identifier' provided by the Periodic Table of Storytelling individuals were to ideally select the top ten most popular tropes and assign them/their Identifiers to the entirety of the NOC list where appropriate. This proved to be a very time-consuming (and subjective) task. Later, we also specifically updated our "Trope List" to include whether or not a given trope featured one or more characters in its implementation.

We programmatically created tropes by searching for certain verbs in the relationship verbs triples. For example, to generate the “Film Noir” trope, we searched the triples for the “is_investigated_by” verb, and used the roles listed in the triple sets to get a list of suitable roles for that trope. The user can then select a trope from a drop-down menu, and its “role” menus will automatically be populated with suitable characters.

User Interface

We created a web-based UI with which a user can combine tropes from the database with which to generate stories. The user selects the tropes that they want in their story, and the characters they want to fit the roles for each trope. The characters and roles from each trope are then sent to the ASP solver, which generates the story text.



Team “Stromberg”

Members of Stromberg team:

Aniruddha Ghosh [arghyaonline@gmail.com]
Derrall Heath [dheath@byu.edu]
Laura Maria Andrei [andrei.laura.maria@gmail.com]
Olli Alm [alm@cs.helsinki.fi]

The project development followed several stages which were obtained by dividing the initial task as it follows:

1. Choosing 2 characters which should be the protagonists of our story and attempting to add some content to the preamble by providing some short descriptions about these characters.

By using the tagged NOC file where a negative, neutral or positive connotation was given to every character in a separate "Sentiment" column, we could select a positive and a negative character among which a confrontation should take place in mid-action

We used the NOC data files containing the positive and negative treats of the characters to enrich the preamble content.

Also emphasized the positive and negative treats of each character by adding in the preamble some general superlatives specific to evil or good extracted from the *superlatives good bad.xslx* file.

2. The story line development was created with the intent of following a sentiment curve which is given through parameters expressing the sentiment intensity which should be expressed through the characters actions.

In the situation where the program cannot return a midpoint result for a specific sentiment by filtering the required intensity called through our functions parameters this plot segment will be randomized in terms of choosing the sentiment intensity among other actions that can be found in *script_midpoints.tsv*.

The code also gives the opportunity to chose the story line length which will be composed by progressively adding actions from *script_midpoints.tsv*

Team “Dr. No”

We started our brainstorming trying to identify the ingredients that are required for a good story, and we agreed the most important one is the "plot twist", i.e. the moment where something unexpected happens, creating dramatic tension and interrupting the flow of an otherwise very obvious (and boring) series of events.

Thus, we decided to generate stories using action pairs from Scéalextric, where the two actions are connected with a "but". A quick test confirmed that a story with the following structure can already be considered reasonable and meaningful:

- initial_bookend_action
- before_action
  but
- after_action
- closing_bookend_action

However, we were still missing something that would make our stories unique and more interesting. We decided that playing with stereotypes could be an interesting choice for that: they are often widely recognizable and they can be fun when inverted or when they are not taken seriously.

To do so, the first thing that is required is a resource linking a category (e.g. "actors", "Canadians", "women") to a negative trait (e.g. being vulgar, narcissistic, posh, absent-minded). We tried to do this annotation partly automatically, by checking which traits are shared by most members of each category in the NOC list (for example, which are the most common properties of actors), but the results were rather disappointing. We thus decided to manually annotate the data we needed, i.e.:

- the grouping criterion (e.g. members of the NOC list where "Address 3" is "USA", or "Category" is "Policeman")
- stereotypical traits for the group thus identified (e.g. "ignorant" or "corrupt")

This data can be found in *Dr.No's_Stereotypes.xlsx*, in the Google Drive folder linked at the end of this post.

One important choice still had to be made: would our story *confirm* or *disrupt* the stereotype? In this second case, the story would be more "acceptable" from a moral point of view, and it could add another level of "unexpectedness" (e.g. imagine the story starting with "Most people think all Italians are cheaters", thus leading you in a direction, but then, suddenly, the Italian protagonist is betrayed by his partner for no particular reason: who could see this coming?). However, it seemed to us that linking stereotypical traits (e.g. "being a cheater", "being lazy", ...) to actions that disconfirm them is actually harder and more time-consuming than doing the opposite (e.g. linking "being a cheater" with "betraying"). So we settled for this approach and created another manually annotated resource, *Dr.No's_Stereotypical_trait-Action.xlsx*.

Here we connected a trait ("corrupt") to one or more actions in Scéalextric that are typically done by someone with this trait ("being corrupted by someone"). We also annotated whether the person with the trait should be the subject or the object of an action (as in "someone corrupts somebody", where the trait should be associated to the second actor).

So, the algorithm now becomes:

1. Pick a random action pair where "but" connects the two actions
2. Pick a random stereotypical trait that is connected with the action in the second part of the action pair. If none exist, pick another action pair.
3. Pick a random member from the NOC list that belongs to a category with the stereotypical trait. If none exist, pick another action pair.
4. Pick another random member from the NOC list as the other actor.
5. Generate the following story:
    - initial_bookend_action
    - before_action

      but
    - after_action
    - closing_bookend_action

We have also decided to introduce a fictional narrator that is telling the story, and showing the moral (i.e. it applies the trait of the specific actor to the whole category). This way we can also introduce a third character in the generated story, adding a bit more variety.

This is supported by two other resources, *Dr.No's_Openings.xlsx* and *Dr.No's_Conclusions.xlsx*, where the narrator is introduced and the moral is drawn respectively. We were going to select a random narrator from a list of well known people that are known for not being the smartest on the planet, and that could possibly believe in stereotypes (e.g. Donald Trump, Kim Kardashian, Lindsay Lohan), but the code was not yet updated to do this, and sets "Dr. No" as the narrator for every story.

Last but not least, we tried to experiment with slightly longer stories using action triples: the algorithm is exactly the same, with the exception that -instead of selecting a random action pair connected by "but"- we select an action triple where "but" connects the last two actions.

The code, the resources and a README describing them can be found at http://bit.do/Dr_No

--The **Dr. No** Team comprises:

(Robert Costa, Lorenzo Gatti, Lieven Menschaert, Charles Pierse and Stefan Riegl)

Team “Goldfinger”

Theory
In his essay “Philosophy of Composition” poet Edgar Allan Poe (1846) purports to describe his approach to literary composition using the example of his poem The Raven. Despite the poetic framing, the account, in fact, prescribes a reasoning process for creating narratives in a step by step manner. Among other suggestions he states that a work of art should start with creating the climax.

Combining that with a classical dramatic structure, where the climax follows a part of rising tension and precedes a part with falling tension provides an interesting approach to fabula generation using the Scealextric data. That, however, requires a measure of tension that is not yet provided by the data. Two datasets offer potential to perform add an annotation for plot-tensions: The script midpoints table, which contains plausible action triples and the action-pairs table that offers valid conjunctions for plausible action pairs. Because one potential action pair can be used for several action triples they offer more variability, and the decision was made to annotate them. An additional tension column was added, and 2705 out of 3699 rows were hand annotated using a tension scale ranging from 0 (no tension) to 5 (high tension). As a guideline, tension was taken to be the grade of unexpectedness in transition from the first action to the second, in conjunction with the development of the violence level.

Thus the pair “are_insulted_by:are_arrested_for_killing” receives a tension rating of 5 due to the implied violence, and the unexpected plot twist, while “hire:are_assisted_by” receives a rating of 0 for the complete lack of violence and a predictable development.

Exploratory work was performed on other ways to measure tension. Since the tension of an action can be affected by the location it has been performed in, a way to compute location tension has been added. This tension is determined in an automatic fashion by computing the average sentiment of all the ambience words provided of the location. The results prove interesting but somewhat questionable due to the idiosyncrasies of the dataset.

Fabula
The fabula generation starts at the climax by choosing a random action-pair with an appropriately high tension level. It then generates the fabula leading to the climax. For that, the Before Action from the action-pair is used to find all suitable mid-point triples.

Such a midpoint triple is considered suitable for our narrative arc only if:
Its After Action is the same as the Before Action of the climax. The tension of the (Midpoint, After-Midpoint) action-pair is below or equal to the climax action

The tension of the (Before-Midpoint, Midpoint) action-pair is below or equal to the previous action pair.

The same procedure is used to generate the fabula that follows the climax (with an appropriate switch in the compared actions).

Such a baseline system is already surprisingly apt in generating coherent fabula consisting of five action-pairs. We assume that it is the backward/forward generation starting at a climax that is responsible for the coherence, as compared to a completely unidirectional generation. The unidirectional case does not have a distinct climax that, supposedly, acts as a stabilizing semantic fixed-point in our approach. This is yet to be empirically evaluated.

Be that as it may, an example for fabula discovered by this approach is: [('eject', '3.0', 'are_threatened_by'), ('are_threatened_by', '3.0', 'attack'), ('attack', '5.0', 'are_arrested_for_hurting'), ('are_arrested_for_hurting', '3.0', 'beg_forgiveness_from'), ('beg_forgiveness_from', '0.0', 'are_forgiven_by')]

Being able to generate a classical dramatic tension arc, we decided to enable the system to create narratives that follow an arbitrary, provided tension arc. As it turned out this was not trivially realizable with the baseline system, so an algorithmic restructuring had to take place, that unfortunately could not be completed in the restricted time available.

The envisioned structure involves dynamically transforming the rated action-pair list into two trees: one for the fabula leading to the climax, and one leading *from* the climax. For each tree, the root node is expanded into all action-pairs whose rating is equal to the climax rating. Each of these nodes, in turn, is expanded into all the possible tension ratings. Subsequently, all suitable midpoint-triples are identified and split into two action-pairs. The first action pairs are collected under the node with the appropriate rating. These again are expanded into all possible ratings, and the second action-pairs are collected accordingly.

Following this procedure an arbitrarily deep tree can be constructed. This tree forms the conceptual space of the fabula, and due to its structure it can be traversed in a tension-rating aware manner. Finding the fabula for both sides of the climax becomes a task of finding a suitable path in the respective tree. This can be done using random search or classical tree-search algorithms like MCTS and is a storybook case of *exploratory creativity* (cf. canonical computational creativity literature like Boden, 1990).

The refactoring procedure conducted by us, in itself is a fine exemplar of a process relevant to creativity and called representational re-description (cf. Gervàs and León, 2015). By manually building the action-tension tree the available data was re-represented by us from a inconvenient list into a easily explored tree. In order to enhance the perceived creativity of the system Colton (2008) generally suggests to “climb the meta mountain”, which means to transfer responsibility for more and more tasks from the programmer to the system. Following this tenet, an appropriate extension for the system would be to implement an autonomous re-description.

Initially this would merely mean automatically building trees of the structure above from arbitrary, rated lists. A good first instance for that would be the action-valence data generated by one of the other teams. The more tricky step, and one involving further research, would be to implement an appropriate logic on how to decide which representation to take when, during exploration. Such a system would also exhibit *transformational creativity* (Boden, 1990) which is a highly desirable trait.

Discourse
As soon as the Fabula module delivered a suitable framework, the implemented discourse module exchanges each action verb with an action event through a connection within the Scealextric. This event is an idiomatic embellishment of the action pair. Because the tension arc has modeled the inherent flow of the story, the coupling of the action events does not create a chaotic turn of events. Rather, it tends to produce a logical, but also intuitive, whole.

An interesting addition to the discourse module are functions that are able to randomly embellish sentences. The core makes use of pattern's drivel, a function that generates a description of a noun. An integer constant (not coincidentally named "NUTSNESS") determines the probability of such embellishments. At times, the result happens to be ridiculous and looks like a sentence from James Joyce's notorious novel "Finnegan's Wake". However, almost all of the results have a surprising and comic effect. An enhanced version of the embellish function could perhaps be a step from "simple" pattern connection to computational creative storytelling, or at least the illusion of it.

We did not focus on the appearance of special characters in our stories. Nevertheless, for the sake of completeness, characters are generated from the NOC list and included in the text. A suitable introduction and ending has also been added.

Example Story:

Fabula: [('woo', '3.0', 'are_kissed_by'), ('are_kissed_by', '3.0', 'flirt_with'), ('flirt_with', '5.0', 'are_abducted_by'), ('are_detained_by', '5.0', 'escape_from'), ('escape_from', '3.0', 'are_pursued_by')]

Story: Keith Moon's natural defenses against seduction would have to be pierced. JD Salinger set about to get Keith Moon into bed in a parliament chamber. This is a vibrant place. Keith Moon kissed JD Salinger tenderly. Later, JD Salinger took Keith Moon for a candlelit dinner in a yoga studio. This is a sweaty place. Horrible. Keith Moon detained JD Salinger against its will. JD Salinger broke free of Keith Moon's clutches, yet Keith Moon pursued JD Salinger. Keith Moon would never give up the hunt for JD Salinger.

Narrative Arc: [3, 3, 5, 5, 3]

Team Goldfinger Participants:

Tom De Keyser, Pieter Fivez, Robert Homewood, Leonid Berov