Synthetic Literature.
Writing Science Fiction in a Co-Creative Process

Enrique Manjavacas [1, 3]
enrique.manjavacas@uantwerpen.be

Folgert Karsdorp [2]
folgert.karsdorp@meertens.knaw.nl

Ben Burtenshaw [1, 3]
benjamin.burtenshaw@uantwerpen.be

Mike Kestemont [1, 3]
mike.kestemont@uantwerpen.be

Computational Linguistics & Psycholinguistics Research Center [1]
The University of Antwerp, Lange Winkelstraat 40-42, Antwerp, Belgium

Meertens Instituut [2]
Oudezijds Achterburgwal 185, 1012 DK Amsterdam, The Netherlands

Antwerp Centre for Digital Humanities and Literary Criticism [3]
The University of Antwerp, Prinsstraat 13, Antwerp, Belgium

Abstract

This paper describes a co-creative text generation system applied within a science fiction setting to be used by an established novelist. The project was initiated as part of The Dutch Book Week, and the generated text will be published within a volume of science fiction stories. We explore the ramifications of applying Natural Language Generation within a co-creative process, and examine where the co-creative setting challenges both writer and machine. We employ a character-level language model to generate text based on a large corpus of Dutch novels that exposes a number of tunable parameters to the user. The system is used through a custom graphical user interface, that helps the writer to elicit, modify and incorporate suggestions by the text generation system. Besides a literary work, the output of the present project also includes user-generated meta-data that is expected to contribute to the quantitative evaluation of the text-generation system and the co-creative process involved.

1 Introduction

In this paper we present ongoing work towards developing a text editing application, through which an established author of Dutch-language literary fiction will use an AI-based text generation system with the goal of producing a valuable piece of literature. Our aim is to create a stimulating environment that fosters co-creation: ideally, the machine should output valuable suggestions, to which the author retains a significant stake within the creative process.

The present project is part of a large-scale initiative by the CPNB ('Stichting Collectieve Propaganda van het Nederlandse Boek', 'Collective Promotion for the Dutch Book'). In Fall 2017, CPNB will launch their annual media campaign, which this year focuses on robotics. To this end, CPNB will distribute a re-edition of the Dutch translation of Isaac Asimov's *I, Robot* that is planned to include an additional piece written as part of a human-machine collaboration.

This project report is structured as follows. We first introduce the CPNB in greater detail, with special emphasis on their annual media campaign. We go on to introduce this year's 'Robotics' theme, and the way it centers around Asimov's *I, Robot*. Then, we concisely survey the state of the art in text generation from the point of view of co-creation between human and machine. Next, we describe our current

text generation system, starting with the large body of Dutch-language fiction (4000+ novels) that is at the basis of our experiments, as well as its preprocessing. We describe our choice of architecture for Natural Language Generation (NLG)—a character-level Language Model (LM) based on Recurrent Neural Networks (RNN) with attractive properties for the present task — and discuss how author and genre-specific voices can be implemented through fine-tuning of pre-trained LMs. We present ample examples to illustrate the model's output for various settings. We also discuss possible ways to evaluate our system empirically—a common bottleneck of text generation systems—, through the monitoring of user's behavior and selectional preferences. Finally, we discuss the design of the interface of our application, emphasizing various ways in which the author will be able to interact with the software.

1.1 Trust for the Collective Promotion of the Dutch-language Book

The CPNB[1] is a trust and PR agency based in The Netherlands that aims to promote the visibility of books and the publishing sector in Dutch society at large. The agency is responsible for a number of high-visibility annual initiatives, such as the 'Boekenbal' ('Book ball') and Boekenweek ('Book week').

These initiatives often center around specific themes. For the 2017 campaign *Nederland leest* ('The Netherlands reads'), the CPNB chose 'robotics' as the overarching theme for their campaign. Thereby further exacerbating the debate as the societal opportunities and challenges that come with the increase of artificial intelligence in everyday life, as well as literature. The campaign, for instance, includes the distribution of promotional material for children (see Figure 1), as well as copies of *I, Robot* (1950)—the well-known science fiction novel by Isaac Asimov—in its Dutch translation *Ik, Robot* (1966) by Leo Zelders, which serves as the focal point of the 2017 campaign. The novel is composed of interrelated short stories, prepublished in the journal *Astounding Science Fiction* between 1940 and 1950. They revolve around the fictional character of robot-psychologist Dr. Susan Calvin. The novel is especially famous because of the 'Three Laws of Robotics' which feature as an intriguing ethical backdrop.

The CPNB wanted to encourage debate about the role of AI and robotics in literature through the addition of a 10th short story co-created by an established fiction writer and a machine. An award-winning Dutch author, Ronald Giphart, agreed to take part in this experiment.

Figure 1: Make-it-yourself cardboard robot. Promotional material distributed as part of the 2017 'Nederland leest!' campaign by the CPNB on robotics and books.

1.2 Co-creativity

Co-creativity is a collaborative process between multiple agents, where in this context, one agent is a computational system. Davis sees co-creativity as the 'blending' of improvisational forces (Davis, 2013). This goes against the pragmatic distributional of labor that we might see in creative support

[1]Stichting Collectieve Propaganda van het Nederlandse Boek: `https://www.cpnb.nl`.

systems, or how computers are treated in everyday life, and invites them into an indistinct and overlapping process of creativitiy. Where crucially, the result of the output is greater than 'the sum of its parts' (Davis, 2013).

Interestingly, as pointed out by existing literature (Lubart, 2005; Davis, 2013; Jordanous, 2017), the public are suspicious of systems that purport to be autonomous whilst in fact involve human participation. Whilst for Lubart the opposite is the case. Lubart reorientates the scientific perception of these systems into one aligned with Human Computer Interaction, where they are examples of successful facilitators of improvisation (Lubart, 2005). Lubart clarifies co-creativity into four distinct roles for a computational system; 'Computer as nanny', 'Computer as penpal', 'Computer as coach', 'Computer as colleague' (Lubart, 2005, p. 366). In this project we are most interested in achieving the last, though in practice, much of what our system does could be considered under the second. For a more thorough overview of co-creativity and its role within computational creativity research, see the proceeding of The International Conference on Computational Creativity 2012 (Maher, 2012), and for a broader of view of the term in relation to computing, look to the work of Edmonds and Candy (Edmonds et al., 2005; Candy and Edmonds, 2002).

Developing NLG systems within a co-creative environment allows researchers to utilize the human agent within the system's workflow, allowing for approaches that are potentially too experimental for a solely computational approach. Furthermore, co-creation adds a collaborative and challenging dimension to the process of writing, which in turn encourages the human writer. That said, though collaboration is commonplace in writing, it is not always welcome. The creative process of writing is associated with a fluidity that can easily be hindered or broken; Umberto Eco's renowned 'How to write a thesis' asserts that writers should nurture their process (Eco, 2015). In developing this system alongside novelist Ronald Giphart, we sought to apply our work within his established methodology in a way that enriches both parties.

From a technical point of view, there is a possibility to limit the collaborative NLG system to an assistive role, solely aiding the writer. However, a valid collaboration should provoke and challenge the writer. It should test them, push them, and ask them to reconsider their approach. To achieve this balance we chose to treat the writer as a competent handler of text, completely capable of dealing with generated language, and unlikely to be overwhelmed. This approach certainly would not work for all applications, but seems appropriate to a professional science fiction writer.

As Natural Language Generation develops into a useful instrument in the creation of fictional prose, inherent questions arise around how computational systems relate to human writers. Nowhere else are these questions more at home than in science fiction literature, where readers and writers are eager to explore the speculative limits of technology. This willingness allowed us to consider the practical implications and qualities of co-creative writing, and how they manifest within the interface itself (see Section 4).

2 Related Work

Natural Language Generation within a collaborative writing environment is an active area of research. The co-creative setting gives scope to apply experimental approaches within the dynamic context of a working process. Here we will outline two established approaches: the structural diagramatic approach, and the auto-completion approach. Ahn, Morbini and Gordon use causal graphs to map the narrative steps of a story which the writer can manipulate into the eventual story structure, the system will then use probabilistic modeling to generate language around that skeleton. This approach gives the system access to the abstract narrative core of a story's structure; arguably, in doing so the system imposes upon the writer a far more structured approach than they are likely familiar with. A collaborative system should be able to fit within a writers existing working process (Ahn et al., 2016). Roemelle and Gordon offer a more hands on approach to assistive writing. Their system acts as a 'Narrative Auto-Completion', where the writer is prompted with possible sentences (Roemmele and Gordon, 2015). Though straightforward, this approach is highly intuitive and unobtrusive; however, the system risks fulfilling the role of tool rather than

collaborator. As such, Creative Help is a retrieval-based system, as appose to the generative approach presented below.

Narrative generation has been a central topic of computational creativity for decades. One of the first examples is Tale-Spin, a system that generates Aesop's Fables guided by a user's keyword suggestions (Meehan, 1977). More recently and nearer to this project, McIntyre and Lapata developed a probabilistic narrative generator that uses user-input to retrieve related phrases (McIntyre and Lapata, 2009). The system here differentiates itself from those by working on the character level. Generated text reproduces the style and voice of its training material, but does not directly sample quotes verbatim from the training material.

3 Method

3.1 Collection and Preprocessing

The first step in constructing our NLG system was to compile a sufficiently large corpus of literary works. In the present study, we employ a large collection of Dutch novels in epub format (Williams, 2011), which contains a diverse set of novels in terms of genre, and is heterogeneous in style. In total, the collection consists of 4,392 novels, written by approximately 1,600 different authors. The average number of novels written by each author is 2.5. The large standard deviation of 6.5 is caused by the skewed distribution in which a few authors contribute relatively large oeuvres, such as detective writer Appie Baantjer. The novels were tokenized for words, sentences and paragraphs using the Tokenizer Ucto, which was configured for the Dutch language (Van Gompel et al., 2012). The total number of sentences, words and characters in the tokenized collection (including punctuation) amounts to approximately 24.6M, 425.5M, and 2.1G, respectively. On average, each novel consists of 3k sentences, 59k words, and 309,531k characters.

3.2 Character-level Language Models for NLG

The aim of this project is to contribute to literary writing in a co-creative environment, as opposed to solely narrative generation. Therefore, we approach NLG using character-based Language Models (LM) which typically reason at a local level, in the order of some few hundreds of characters. Because of this, the LM is only implicitly aware of the global narrative structure, but still powerful enough to capture sentence semantics in a unsupervised fashion.

An LM is a probabilistic model of linguistic sequences that estimate a probability distribution over a given vocabulary conditioned on the previous text (left-to-right model). More formally, at a given step t, an LM defines a conditional probability, expressing the likelihood that a certain vocabulary item (typically a word or character) will appear next:

$$LM(w_t) = P(w_t|w_1, w_2, ..., w_{t-2}, w_{t-1}) \quad (1)$$

Different LM implementations exist, which diverge in the manner in which they model the previous text. Given their probabilistic nature, LMs are straightforward to deploy for NLG. The generative process is defined by sampling a character from the output distribution at step t, which is then recursively fed back into the model, potentially preceded by the previous output of the model, to condition the next generation at step $t + 1$. A few decoding approaches can be implemented based on different sampling strategies. For instance, a rather naive approach towards sampling is to select each character so as to maximize a generated sequence's overall probability. Nevertheless, for a large vocabulary size (e.g. in the case of a word-level model), the search soon becomes infeasible; therefore, approximate decoding methods, such as beam search, are used to find an ideal solution. When used for generation, the naive *argmax* decoding strategy has a tendency towards relatively repetitive sentences, that are too uninspiring to be of much use in a creative setting. For the present work, we therefore decode new characters via sampling from the multinomial distribution at each step.

It is interesting to note that the different decoding approaches stand in a trade-off relationship between diversity and correctness. For example, whereas *argmax* decoding will tend to generate sentences that are very similar, general and monotonous yet formally correct (e.g. more similar to the sentences observed in the training corpus), multinomial sampling will make the output diverge more from the original training data, and therefore produce a more varied output, with a tendency towards formally incorrect sentences. Focusing on our chosen

approach—multinomial sampling—, the described trade-off can be operationalized and used to our advantage by letting the author explore model parameters. This is implemented by exposing a parameter τ, commonly referred to as "temperature", that controls the skewness of the model's output distribution. Given the output distribution at a given step $p = (p_1, p_2, ..., p_V)$, a vocabulary size of V, and the temperature value τ, we can compute a transformation of p^τ of the original p through Equation 2

$$p_i^\tau = \frac{p_i^{\frac{1}{\tau}}}{\sum_j^V p_j^{\frac{1}{\tau}}} \tag{2}$$

p^τ will flatten the original distribution for higher values of τ— thereby ensuring more variability in the output. Conversely, for lower values of τ it will skew the distribution—thereby facilitating the outcome of the originally more probable symbol. For τ values approaching zero, we recover the simple argmax decoding procedure of picking the highest probability symbol at each step, whereas for high enough τ the LM degenerates into a random process in which at any given step all symbols are equally probable regardless of the history.

In terms of implementations there are two major approaches to statistical language modeling—ngram-based LMs and RNN-based LMs.

3.2.1 Ngram Language Models

Ngram-based LMs (NGLMs) go back to at least the early 1980s in the context of Statistical Machine Translation and Speech Recognition (Rosenfeld, 2000). An NGLM is a direct application of the Markov assumption to the task of estimating the next character probability distribution—e.g. it uses a fixed-length ngram prefix to estimate the next character probability distribution. An NGLM is basically a conditional probability table for Equation 1 , that is estimated on the basis of the count data for ngrams of a given length n. Typically, NGLMs suffer from a data sparsity problem, because with larger values of n possible conditioning prefixes will not be observed in the training data and the corresponding probability distribution cannot be estimated. To alleviate the sparsity problem, techniques such as smoothing and back-off models (Chen and Goodman, 1999) can be used to either reserve some probability mass to redistribute it across unobserved ngrams (smoothing), or resort back to a lower-order model to provide an approximation to the conditional distribution of an unobserved ngram (back-off models).

3.2.2 RNN-based Language Models

More recently, a new class of LMs based on Recurrent Neural Networks (Elman, 1990) have been introduced (Bengio et al., 2003; Mikolov, 2012) and have quickly increased in popularity due to their better theoretical properties (no Markov assumption), expressive capabilities (information flow through very long sequences) and performance gains. An RNNLM processes an input sequence one step t at a time, feeding the input symbol x_t through three affine transformations with their corresponding non-linearities. First, the one-hot encoded input vector is projected into an embedding space of dimensionality M through $w_t = W^m x_t$, where $W^m \in \mathbb{R}^{MxV}$ is a embedding matrix. Secondly, the resulting character embedding w_t is fed into an RNN layer that computes a hidden activation h_t as a combination of w_t with the hidden activation of the previous step h_{t-1}. This is shown formally in Equation 3

$$h_t = \sigma(W^{ih} w_i + W^{hh} h_{t-1} + b_h) \tag{3}$$

where $W^{ih} \in \mathbb{R}^{MxH}$ and $W^{hh} \in \mathbb{R}^{HxH}$ are respectively the input-to-hidden and hidden-to-hidden projection matrices, b_h is a bias vector and σ is the sigmoid non-linear function. Finally, the hidden activation h_t is projected into the vocabulary space of size V, followed by a *softmax* function that turns the output vector into a valid probability distribution. Formally, the probability of character j at step t is defined by

$$P_{t,j} = \frac{e^{o_{t,j}}}{\sum_k^V e^{o_{t,k}}} \tag{4}$$

where $o_{t,j}$ is the jth entry in the output vector $o_t = W^{ho} h_t$ and $W^{ho} \in \mathbb{R}^{VxH}$ is the hidden-to-output projection.

In practice, training an RNN is difficult due to the vanishing gradient problem (Hochreiter, 1998) that makes it hard to apply the back-propagation learning algorithm (Rumelhart et al., 1986) for parameter learning over very long sequences. Therefore,

it is common to implement the recurrent layer using an enhanced RNN version to compute h_t—such as Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or Gated-Recurrent Unit (GRU) (Cho et al., 2014)—, which add an explicit gated mechanism to the traditional RNN in order to control the preservation of information in the hidden state over very long sequences.

3.2.3 Model

For the present study, we implement several variations of the RNNLM, varying the type of the recurrent cell (LSTM, GRU) as well as the values of different parameters, such as the dimensionality of the character embedding matrix W^m (24, 46, ...) and, more importantly, the size of the hidden layer H (1024, 2048, ...). We train our models through back-propagation using Stochastic Gradient Descent (SGD), clipping the gradients before each batch update to a maximum norm value of 5 to avoid the exploding gradient problem (Pascanu et al., 2013) and truncating the gradient during back-propagation to a maximum of 200 recurrent steps to ensure sufficiently long dependencies in the sequence processing. Finally, dropout (Srivastava et al., 2014) is applied after each recurrent layer following (Zaremba et al., 2015) to prevent overfitting during full model training.

3.2.4 Overfitting

After training the full model, we experiment with further fine-tuning on different author and genre specific subsets to steer the NLG towards a particular style. To enforce this effect, we drive the training towards overfitting introducing an intended bias in the models predictions towards sequences that are more likely in that particular book subset. We achieve overfitting by zeroing the dropout rate and running numerous passes through the subset training data, with a sufficiently small learning rate. We have already observed interesting stylistic and genre properties in the fine-tuned model's output—see Section 4.2 for an illustration. That said, how the particular effect of this technique—differing degrees of overfitting—affects the quality of the generated output still has to be evaluated. The degree of overfitting can easily be quantified and monitored by plotting batched-average perplexity values achieved by the model for both the training data and the validation split as shown in Figure 2.

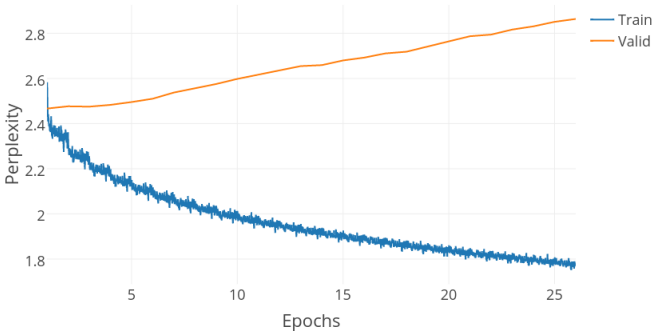

Figure 2: Example of overfitting learning curves during fine-tuning of a full model on a subset of novels by Isaac Asimov.

4 User Interface

Uncharacteristically for an NLP project, the visual interface of the system is paramount to its success. Though ultimately the system will be assessed on the language it produces, such language can only be generated if the writer is able to use the system. Therefore, we have focused on functionalities that give the user a clear representation of how text is generated, and allow them to understand how their own writing is affected by the process. This allows them to play a defined role within the process of writing, whilst also encouraging them to use generated text. The user is able to select which model to generate text from, so that they can use multiple voices and approaches within the same text (see Section 3.2.4). The user can define 'temperature' for any model using a slider bar (see Section 3.2). The generated text itself is shown as a list of suggestions, along with the model's own probability scores, below the main text area. This allows the writer to choose between a set of options, and get a broader idea of the models voice. With the help of user edit meta-data—computed by a string *diffing* algorithms—we can track user changes and present them with a visualization of each fragment's source and the degree of the modification.

Figure 3 is a visual representation of how text annotation functions. Text annotation reveals to the writer how generated text is affecting the final text. As the writer works into text, they could easily lose

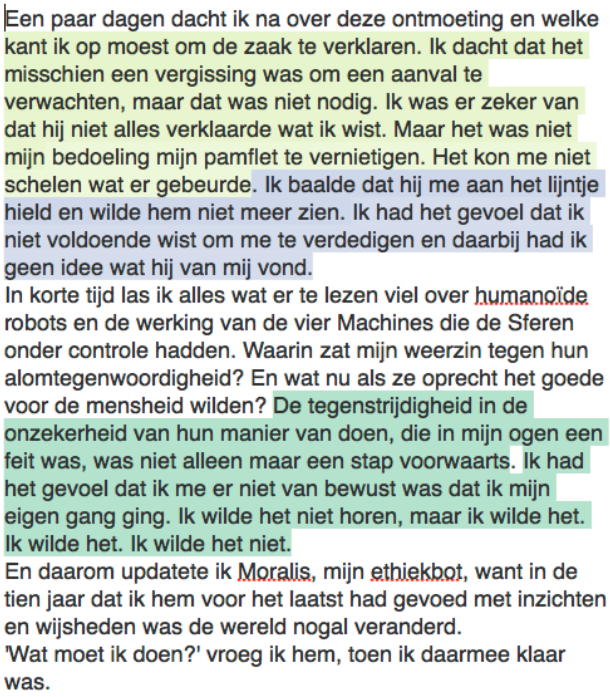

Figure 3: Visual feedback on the co-creation process used by Ronald Giphart. Highlighted fragments are synthetic in origin with the brightness indicating the amount of modification introduced by the user.

track of its source; therefore, the interface is enhanced with visual feedback which highlights based on edit distance between original generated text and its current status. Generated text is initially highlighted in green, as the writer edits that text its color fades into white. At the same time, whole words swapped by the writer are underlined in purple to differentiate lexical changes (see Figure 3).

4.1 Monitoring the Author

Our project intends to explore the co-creative process of science-fiction literature on a quantitative and objective basis. In line with (Roemmele and Gordon, 2015), we acknowledge that such a co-creative interface opens the up possibility for automatic evaluation of generative systems based on user edits of generated strings. Our interface is therefore designed to store all user edits along with the source of the string (human or machine generated). This will enable us to study individual user behavior in relation to the particular properties of the generative system, as well as the aptness of different model variants and their parameter settings (e.g. degree of overfit, temperature, voice) for co-creation, taking user edit behavior as a proxy for output quality.

4.2 Examples

As explained in the previous section, the evaluation of a NLG system in a co-creative setting involving both human and machine amounts to the generated material incorporated (either explicitly or implicitly) by the author in the final work. Suggestions about how to formally and informally evaluate this co-production process were given in the previous section. Here, we provide an exploratory demonstration of the model's generation system, where the goal is to highlight some typical behavior of the system under different parameters settings, author-based fine-tuning, and text seeds.

We begin with a survey of how different temperature values τ impact the generated text. In Table 1 we list a number of generated sentences for different temperatures given the famous opening sentence "Mijn vrouw is dood en al begraven" ('My wife is dead and already buried') from Marcellus Emants' *Een nagelaten bekentenis*. These examples have been generated on the basis of a model trained on the complete collection as described in Section 3.1. Several observations can be made from these examples. First, the examples showcase the model's ability to generate grammatical sentences. Interestingly, the sentences become syntactically more complex as the temperature increases (as explained in Section 3.2), and also exhibit less repetitive grammatical structures (cf. the repetition of possessive *van* constructions with $\tau = 0.1$). Second, a similar diversifying effect can be observed on the lexical level. Higher temperatures are more likely to generate less frequent and semantically more specific lexical material (cf. the decrease in reusing the same word twice, as well as the increase of names in the final two examples). Third, depending on the reader's willingness to search for semantic coherence, most generated sentences can be read as plausible continuations of the input seed. Finally, with increased temperature values the model composes sentences with higher semantic and cognitive complexity, juxtaposing and combining less common concepts and thoughts. Within certain limits, we expect such increased 'creativity' to encourage and provoke the human writer in the process of writing.

τ \ seed	Mijn vrouw is dood en al begraven.	My wife is dead and already buried.
0.1	*De volgende dag werd er een bericht van de politie gevonden. De verklaring van de politie was een van de belangrijkste verklaringen van de man die het verhaal van de politie had gehoord.*	*The next day, a police message was found. The police's statement was one of the most important statements of the man who heard the story by the police.*
0.3	*De dokter keek op zijn horloge. Ze was er zeker van dat hij die dag niet meer zou komen. Hij was niet gek. Hij was niet van plan om zijn dochter te vertellen dat hij haar een keer had gezien.*	*The doctor looked at his watch. She was sure he wouldn't come anymore that day. He wasn't crazy. He did not intend to tell his daughter he'd seen her once.*
0.5	*De tranen weerklonken over haar gezicht en voelde ze in haar keel. Ze was er niet. Ze was er zeker van dat ze alleen maar een affaire had gehad met de meisjes.*	*The tears resounded over her face and she felt them in her throat. She wasn't there. She was sure she had only had an affair with the girls.*
0.7	*Ze zegt dat het weer in San Sebastian naar Californië haar deel van de wereld is. Ze moet van de wereld naar de hemel hebben gelopen.*	*She says that the weather in San Sebastian to California is her part of the world. She must have walked from the world to heaven.*
0.9	*Lucas kon niet met zichzelf trouwen en toen hij achteloos met zijn gebaren zat te praten maakte hij deel uit van de lessen van de waarheid.*	*Lucas couldn't marry himself, and when he spoke painlessly with his gestures, he was part of the lessons of truth.*

Table 1: Example of our current NLG system with translation seeded by "Mijn vrouw is dood en al begraven" (My wife is dead and already buried) for different temperature τ values.

Having explored the impact of temperature on the full model's output, we now proceed with a brief illustration and informal evaluation of the generated output of two fine-tuned models. As explained in Section 3.2.4, we experiment with constructing fine-tuned models for specific styles, genres or authors by post-training on a subset of the collection and driving the training towards overfitting. In this section, we demonstrate the effect of overfitting two models post-trained on novels by Isaac Asimov and Ronald Giphart, who form the heart of the CPNB's robotics campaign. Using the same seed from Table 1, we observe a clear style shift when generating sentences using either the Asimov or Giphart model. For example, with a temperature setting of $\tau = 0.4$ the Asimov model produces utterances such as: "Mijn vrouw is dood en al begraven. *'Het is de grootste misdaad die ik ooit heb gezien.' 'Weet u dat zeker?' 'Ja.' 'En als dat zo is, wat is dan wel de waarheid?'* (My wife is dead and already buried. 'It is the biggest crime I've ever seen.' 'Are you sure?' 'Yes.' 'And if so, what is the truth?'). By contrast, a model overfitted on novels by Giphart generates output such as: "Mijn vrouw is dood en al begraven. *Ik heb het over een door mij gefotografeerde vrouw, een hoofd dat met haar borsten over mijn schouder ligt. Ik heb de film geschreven die ik mijn leven lang heb geleefd.*" ('My wife is dead and already buried. I'm talking about a woman I once photographed, a head with her breasts over my shoulder. I wrote the movie I've lived my life for a long time.') Both continuations are semantically plausible, yet written in completely different styles, and put focus on different concepts (e.g. 'crime' versus 'erotics'), both typical of their respective training material.

5 Conclusion

In this paper we have outlined an applied text generation system and graphical user interface, that together facilitate co-creative environment in which to write science fiction literature. We have highlighted an existing challenge within state of the art systems, to balance a challenging intervention into the writing process, with the risk of becoming a solely a writing tool. The character-level recurrent neural network for NLG that we have used is experimental within a solely computational approach, and therefore we have leveraged the specific advantages of working with a professional writer to maximize this system's ability to be applied. We have how to facilitate a writer to use a language model. We have outlined

evaluation procedures for the current NLG system, utilizing user-generated meta-data and quantifying the extent of retained synthetic text.

Acknowledgments

We would like to thank Ronald Giphart for his time and energy and Stichting Collectieve Propaganda van het Nederlandse Boek for initiating the collaboration.

References

Emily Ahn, Fabrizio Morbini, and Andrew S. Gordon. 2016. Improving Fluency in Narrative Text Generation With Grammatical Transformations and Probabilistic Parsing. In *The 9th International Natural Language Generation conference*, pages 70–74, Edinburgh, UK. ACL.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155.

Linda Candy and Ernest Edmonds. 2002. Modeling co-creativity in art and technology. In *Proceedings of the 4th conference on Creativity & cognition*, pages 134–141, Loughborough, UK. ACM.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394.

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation : Encoder – Decoder Approaches. *Ssst-2014*, pages 103–111.

Nicholas Davis. 2013. Human-computer co-creativity: Blending human and computational creativity. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI Press.

Umberto Eco. 2015. *How to write a thesis*. MIT Press.

Ernest A. Edmonds, Alastair Weakley, Linda Candy, Mark Fell, Roger Knott, and Sandra Pauletto. 2005. The studio as laboratory: Combining creative practice and digital technology research. *International Journal of Human-Computer Studies*, 63(4-5):452–481, October.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sepp Hochreiter. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116.

Anna Jordanous. 2017. Co-creativity and perceptions of computational agents in co-creativity. In *Proceedings of the Eighth International Conference on Computational Creativity*, Atlanta, US. ACC.

Todd Lubart. 2005. How can computers be partners in the creative process: Classification and commentary on the Special Issue. *International Journal of Human-Computer Studies*, 63(4-5):365–369, October.

Mary Lou Maher. 2012. Computational and Collective Creativity: Who's Being Creative? In *Proceedings of the 3rd International Conference on Computer Creativity*, pages 67–71, Dublin, Ireland. ACC.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225, Singapore. Association for Computational Linguistics.

James R. Meehan. 1977. TALE-SPIN: An interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Ar tificial Intelligence*, pages 91–98.

Tomas Mikolov. 2012. Statistical Language Models Based on Neural Networks.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulties of Training Recurrent Neural Networks. *Icml*, (2):1–9.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative help: a story writing assistant. In *International Conference on Interactive Digital Storytelling*, pages 81–92. Springer.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Maarten Van Gompel, Ko Van Der Sloot, and Antal Van den Bosch. 2012. Ucto: Unicode Tokeniser Reference Guide. Technical report.

Greg Williams. 2011. EPUB: Primer, Preview, and Prognostications. *Collection Management*, 36(3):182–191.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent Neural Network Regularization. *ICLR*, pages 1–8.