NEURAL NETWORKS FOR NARRATIVE CONTINUATION

by

Melissa Roemmele

A Ph.D. Dissertation Presented to the

FACULTY OF THE GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(COMPUTER SCIENCE)

May 2018

Abstract

The field of artificial intelligence has long envisioned using computers to automatically write stories. With the advent of machine learning, in particular neural networks, researchers have found new opportunities to automatically acquire narrative knowledge directly from text corpora. There is now interest in developing systems that interface with human-authored stories in order to dynamically predict 'what happens next' in a story. In this thesis, I apply a set of neural network approaches to this narrative continuation task. I examine the task within two frameworks. In the first (closed-choice prediction), the system is presented with a story and must choose the best continuing sentence from a set of provided candidates. In the second (free-text generation), there are no candidates given for the next sentence, and the system must generate a new continuation. I demonstrate some evaluation approaches and applications associated with each framework. I discuss the observed successes and challenges of these neural network techniques in order to motivate future work in this up-and-coming research domain.

Acknowledgements

Thank you to my advisor Andrew for the many opportunities that eventually led to this thesis, starting with a summer internship nearly eight years ago. I couldn't have found a more supportive professor to guide me. His immense creativity not only planted the seeds for the research in this thesis, but also helped me find my own creative spark as an emerging researcher. I appreciate everything I have learned from working with him.

Thank you to the many other people who have influenced my academic journey, including my collaborators at USC and everyone I learned from at Indiana University and Miami University. In particular, thank you to my MA advisor Markus Dickinson.

Thank you to my mom for so strongly demonstrating the independence, self-reliance, and work ethic that I've needed to complete this process. I wouldn't have gotten here if it weren't for her commitment to these values while I was growing up. And to my dad, who introduced me to my first computer and encouraged me to embrace science - I think about the good conversations we'd be having about artificial intelligence right now, one small reason among many that I wish he was still here. To my brothers Brian and Eric - research involves fighting for your ideas, and there is no better practice for that than growing up with siblings. Sorry for

all the faulty arguments I got away with as the older sibling, and thank you for being the best people to tell me when I'm wrong.

Finally, thank you to my husband Ross who commiserated with me through all of the setbacks and reassured me through all of the self-doubt that arose over the last few years. His intellect and talent continues to inspire me every day. Thank you for giving me so much joy.

Contents

Abstract **2**

Acknowledgements **3**

List of Tables **8**

List of Figures **9**

1 Introduction **10**
1.1 What is a Story? . . . 11
1.2 Relation to other AI Tasks . . . 12
1.3 Applications . . . 13
1.4 Overview . . . 15

2 Background **16**
2.1 Early Foundations . . . 16
2.2 Narrative Event Prediction . . . 22
2.3 Interactive Storytelling . . . 25
2.4 Creative Language Generation . . . 30
2.5 Automated Support for Writers . . . 31

3 Technical Overview **34**
3.1 Neural Networks . . . 34
3.2 Recurrent Neural Networks . . . 37
3.2.1 Gated Recurrent Units . . . 42
3.3 Text Processing with Neural Networks . . . 44

4 Choice of Plausible Alternatives **47**
4.1 Task Design . . . 49
4.2 Existing Approaches . . . 50
4.2.1 Pointwise Mutual Information (PMI) . . . 50
4.2.2 Causal Net . . . 51
4.2.3 Other Approaches . . . 53

4.3 Neural Network Approach . . . 54
4.3.1 Sequence Segmentation . . . 55
4.3.2 Sequence Pairs . . . 55
4.3.3 Encoder-decoder Models . . . 56
4.4 Initial Experiments . . . 58
4.4.1 ROCStories Corpus . . . 58
4.4.2 Procedure . . . 59
4.4.3 Results . . . 60
4.4.4 Other Findings . . . 61
4.5 Experiments on Other Datasets . . . 65
4.5.1 Procedure and Results . . . 66
4.6 Conclusion . . . 68

5 The Story Cloze Test 70
5.1 Task Design . . . 71
5.2 Initial Approaches . . . 72
5.2.1 Results . . . 73
5.3 Binary Classification Approach . . . 74
5.3.1 Story Representations . . . 76
5.3.2 Incorrect Ending Generation . . . 78
5.3.3 Experiments . . . 80
5.3.4 Results . . . 81
5.4 Comparison with Other Approaches . . . 84
5.5 Conclusion . . . 85

6 Data-driven Interactive Narrative Engine 88
6.1 Application Design . . . 89
6.2 Dataset . . . 90
6.3 Prediction Models . . . 93
6.3.1 Supervised Approaches . . . 94
6.3.2 Unsupervised Approaches . . . 97
6.4 Results . . . 99
6.4.1 Analysis of Setup Design . . . 101
6.5 Conclusion . . . 102

7 Free-text Story Continuation 104
7.1 Task Design . . . 106
7.2 Generation Models . . . 107
7.2.1 Training Data . . . 107
7.2.2 Case-based Reasoning (CBR) . . . 109
7.2.3 Recurrent Neural Network (RNN) . . . 110
7.2.4 Baselines . . . 112

7.3 Automated Linguistic Analyses . . . 112
7.3.1 Story-Independent Metrics . . . 113
7.3.2 Story-Dependent Metrics . . . 115
7.4 Results and Discussion . . . 117
7.5 Conclusion . . . 120

8 Creative Help **122**
8.1 Related Applications . . . 123
8.2 Interface . . . 124
8.3 Generation . . . 125
8.3.1 Motivation . . . 125
8.3.2 Model Setup . . . 128
8.4 Experiment . . . 129
8.5 Analysis and Results . . . 130
8.5.1 Descriptive Analysis of Suggestions . . . 130
8.5.2 Qualitative Feedback . . . 133
8.5.3 Quantitative Analysis . . . 133
8.6 Conclusion . . . 139

9 Conclusion **142**
9.1 Summary of Closed-Choice Prediction . . . 143
9.2 Summary of Free-text Generation . . . 146
9.3 Final Outlook . . . 148

Bibliography **151**

List of Tables

4.1 Examples of stories in ROCStories corpus and similar COPA items 59
4.2 Accuracy by segmentation unit and pair distance (N) for the FFN and RNN encoder-decoders trained on the ROCStories corpus . . . 61
4.3 Accuracy of FFN trained on ROCStories with different input representations . . . 61
4.4 Accuracy of FFN trained on ROCStories with explicit phrase representations . . . 63
4.5 Accuracy of PMI and CausalNet trained on ROCStories . . . 64
4.6 Accuracy of the FFN encoder-decoder on different datasets . . . 66

5.1 Examples of Story Cloze Test items . . . 72
5.2 Accuracy of initial approaches on the Story Cloze Test . . . 74
5.3 Examples of generated negative endings . . . 78
5.4 Accuracy of binary classification methods on the Story Cloze Test . 82

6.1 Accuracy and F1 scores of supervised models on the DINE evaluation set according to training set (author inputs only, annotated user inputs only, or both) . . . 98
6.2 Accuracy and F1 scores of unsupervised models on the DINE evaluation set . . . 98
6.3 Accuracy of best supervised model (WordEmb LR) and unsupervised model (AvgMaxSim) for each setup design category . . . 101

7.1 Examples of CBT story contexts and continuations generated by each model . . . 108
7.2 Mean scores on metrics for generated sentences and gold sentences . 118

8.1 Examples of generated suggestions and corresponding modifications with their preceding story context . . . 131
8.2 Most common generated words/phrases in the suggestions . . . 132
8.3 Sample of participant feedback . . . 134
8.4 Linguistic metric scores for suggestions vs. modifications . . . 137
8.5 Correlation ρ between metric scores for suggestions and similarity to modifications . . . 138

List of Figures

2.1 Text-based interaction in Zork 27

3.1 Abstract view of a neural network 35
3.2 RNN Language Model 38
3.3 Unrolled view of RNN LM with an example sequence 41

4.1 FFN encoder-decoder model 56
4.2 RNN encoder-decoder model 57

5.1 RNN-based Binary Classifier for the Story Cloze Test, with an example of a positive input (correct ending) and a negative input (incorrect ending) 75

6.1 A DINE story with example user inputs 91
6.2 Logistic Regression (left) and Multilayer Perceptron (right) classifiers 95
6.3 RNN classifier 96
6.4 Feed-forward encoder-decoder 97

8.1 Creative Help interface with a generated suggestion 125

Chapter 1

Introduction

People need stories; they want stories. They always will.

Judy Blume

People have always engaged in storytelling, whether for the purpose of teaching, entertaining, establishing a sense of identity, or simply relating to other people in a meaningful way. Technology has created new opportunities for people to share stories, from digital text, to film, to video games, to social media, to virtual reality. The general trend is one of expanding interactivity to enable people to feel more immersed in a story and to immerse others in their own stories. Certain modalities of interactivity have advanced at a significant pace, such as graphical interfaces where people can visually explore a story environment, for example. On the other hand, gaining agency in this type of environment via natural language interaction still exists more in science fiction than in real technology. Among existing systems, the dominant strategy has been to constrain language interfaces to the point of recognizing only a few utterances for advancing the story. Natural language input has an infinite number of possible forms, and narrative technologies have been challenged to account for this productivity.

In this thesis, I explore this vision of systems that tell stories in collaboration with natural language input from people. I investigate an emerging paradigm in artificial intelligence (AI) and natural language processing (NLP) that has not been thoroughly explored for this particular objective: neural network models. A

neural network is a general machine learning framework that learns a latent feature representation of an input to enable prediction of an output. In the context of narrative modeling, neural networks can be applied to predict 'what happens next' in a story, a task that I refer to as *narrative continuation* (or equivalently, story continuation). I focus on two versions of narrative continuation in this thesis: *closed-choice prediction* and *free-text generation*. In closed-choice prediction, a set of candidates for 'what happens next' is given, and the objective is to select which one is the best continuation of the story. In free-text generation, these choices are not provided, and instead a new continuing sequence is generated. These tasks differ in their applications and strategies for evaluation, but they are complementary in that progress on one can support progress on the other. Accordingly, I explore similar neural approaches for both, showing some examples of how this general framework can be adapted to meet their differing requirements. The objective of this thesis is to show that machine learning applied to narrative text, particularly neural models that compute a latent representation of the story, can enable storytelling applications where the medium of human interaction is intuitive natural language.

1.1 What is a Story?

People have such an intuitive sense of what a story is that it is elusive to define, even from a research perspective. A great deal has been written about what constitutes a story, with overlapping ideas but no exact consensus (Bal, 2009; Leitch, 1986; Stanzel, 1979, e.g.). One problem is that the proposed definitions are difficult to formalize computationally. Gervás (2009) provides an overview of the characteristics of narrative that are most relevant to automated story generation systems, but

these characteristics are largely application-dependent. For this reason, this thesis does not impose any formal criteria for what defines a text as a 'story'. While there are obviously many mediums in which a story can be expressed, I exclusively focus on text representations of stories here. I reference many of the informal concepts associated with stories that are interpretable to general audiences. First, I refer to stories as a form of creative language, but I do not speculate about what makes a text 'creative'. I assume that a story conveys a sequence of events over time (Labov and Waletzky, 2003). These events are bound together by common characters, locations, objects, and actions, which together establish the coherence of the story. Events are not necessarily presented in chronological order, but they at least have some temporal relation to one another. Additionally, stories present causal relations between events by portraying one event happening as a result of another event (Trabasso, 1982). Story events are told from the subjective point of view of a narrator (even if the narrator is third person omniscient). The events also comprise the plot of a story, a concept that is intuitive to most people (e.g. the beginning, middle, and end of the story) (Theune et al., 2003). I also refer to other features of narrative like theme, imagery, and tension, but these are mostly discussed in the context of proposed future work. The experiments in this thesis make use of different story corpora whose properties vary according to these elements, which I address in my discussion of certain results.

1.2 Relation to other AI Tasks

Narrative continuation shares the same general challenges of human-computer interaction involving natural language. Obviously, these systems are all required to have natural language understanding ability. For a particular language, much

of this knowledge will be generalizable across different domains (e.g. English-language fiction and English-language dialogue have many of the same linguistic properties). These systems also must all have some model of users' expectations of what should occur next in a fluent interaction. For example, continuing a user's story in a coherent way is analogous in many ways to providing an appropriate response to a user's question. It is increasingly recognized that this type of commonsense reasoning is as important for natural language understanding systems as linguistic knowledge itself. Accordingly, the same techniques used in this thesis have been applied to other NLP systems like dialogue generation. This generalizability is advantageous from the perspective that research on story continuation supports progress on other NLP tasks, but it does not mean that one generic solution will ultimately work for all systems. There are unique challenges in computationally modeling the components of narrative outlined in the previous section. These challenges are particularly revealed through the shortcomings of the current techniques in this work. Chapter 9 will summarize these problems and propose some future high-level directions for moving towards solutions.

1.3 Applications

Given how remarkably good humans are at telling stories, narrative continuation is a critical (and fascinating) task from a core research perspective. But it has practical applications as well. Chapter 6 illustrates an application of the closed-choice prediction task (the Data-driven Interactive Narrative Engine; DINE), which enables a user to choose the direction of a story via natural language input. The resulting experience is related to video gaming and thus can serve the purpose of entertainment, but it is promising for other domains as well. It can be used as an educational

tool to author scenarios that support some training objective. For example, Prasad et al. (2017) crafted a DINE scenario that casts the user (the protagonist) as the target of workplace harassment, with the interaction focusing on navigating difficult encounters with employees. Though this was a case study not intended to be immediately used for training, it can potentially serve the objective of preventing and resolving workplace harassment by enabling users to explore the effect of alternative actions towards this end. Effective training via these scenarios depends on users being able to respond naturally as if they were actually in that scenario, and this is unlikely to be fully realized without the capacity to use natural language. Systems like DINE may also be applicable to medical interventions like psychotherapy. The use of storytelling to improve psychological outcomes is well-known (Pennebaker, 2000). In Roemmele et al. (2017c), I explored this possibility by authoring a DINE scenario intended to facilitate cognitive-behavioral imaginal exposure therapy for treating anxiety disorders. This type of tool could be utilized by therapists as an additional resource for providing treatment. Generally, DINE can be applied to any domain where the user can benefit from using natural language to drive an imaginary simulation of a real-life scenario.

Alternatively, this work can position the user not just as the consumer of an existing story, but as its primary author. This is the context in which narrative continuation via free-text generation is examined. Chapter 8 demonstrates a platform for free-text generation, Creative Help, that is intended to assist people with writing stories. As discussed in the next chapter, automated writing support tools are a relatively new application of NLP, and they are poised to have a significant impact. Many people are uncomfortable with the prospect of machines replacing human creativity, but they are more open to using tools that augment their own creativity. Applications that offer authors new ideas for what to write about, or

new ways to express these ideas, must be able to interface with authors' content in order to provide targeted support.

1.4 Overview

This thesis is organized as follows. The next chapter (Chapter 2) provides some context for this work by highlighting some of the notable previous research in narrative generation. Chapter 3 gives a general technical overview of the neural network models applied in this thesis. The subsequent three chapters each focus on a formulation of the closed-choice prediction task and the neural models applied to it. In particular, Chapter 4 examines the Choice of Plausible Alternatives (COPA), which involves identifying sentence pairs that are causally related. Chapter 5 explores the related task of selecting a plausible ending for a story, referred to as the Story Cloze Test. Chapter 6 demonstrates how closed-choice prediction can be applied in an interactive user application, specifically the DINE system introduced above. The second part of this thesis addresses narrative continuation via free-text generation. Chapter 7 describes a neural model for this task and examines some strategies for automatically evaluating generated continuations. Chapter 8 then illustrates this generation model in the writing assistance application mentioned above. Finally, Chapter 9 concludes this thesis with some discussion about the insights gained from this work and poses some questions to be addressed by future story generation research.

Chapter 2

Background

Maybe stories are just data with a soul.

Brene Brown

This chapter reviews some of the influential research related to computational modeling of narrative. I first provide a brief history of the foundational work on automated story generation, which is a long-pursued endeavor in AI (Section 2.1). Next, I describe more recent work on using data-driven methods for modeling narrative event structure (Section 2.2). I then discuss the medium of interactive digital storytelling and motivate the opportunity for innovation in this domain (Section 2.3). I briefly identify some of the recent work on creative language generation (Section 2.4), which uses some of the same techniques currently being explored for narrative generation. Finally, I address how AI has been used to support writing and identify the newly emerging vision of automated help for creative story writing (Section 2.5).

2.1 Early Foundations

The vision of computers automatically writing stories was identified soon after the birth of AI. See Gervás (2009) for a lengthy account of the history of this enterprise, which I will briefly summarize here. Like many classical AI systems, early approaches to story generation were based on hand-authored formal rules. They

focused on generating latent story structure rather than story text. In 1973, the first story generator Novel Writer (Klein et al., 1973) emerged. Novel Writer generated murder mysteries containing a fixed sequence of scenes. A formal description of the plot setting and character traits were taken as input, and probabilistic rules governed the chain of events and character behavior comprising the story. Soon after that came TALE-SPIN (Meehan, 1977), which became recognized as the foundation of AI research on story generation. TALE-SPIN generated stories about woodland creatures like the following:

> *Once upon a time George Ant lived near a patch of ground. There was a nest in an ash tree. Wilma Bird lived in the nest. There was some water in a river. Wilma knew that the water was in the river. George knew that the water was in the river. One day Wilma was very thirsty. Wilma wanted to get near some water. Wilma flew from her nest across a meadow through a valley to the river. Wilma drank the water. Wilma wasn't very thirsty any more.*

This system uses formal planning techniques like backward chaining (see Russell and Norvig, 2009) to generate actions performed by characters based on their goals. For instance, in the example above, Wilma has the goal of quenching her thirst. In order to do this, Wilma needs to drink water. In order to get water, Wilma needs to go to the river. In order to go to the river, Wilma must fly from her nest across a meadow through a valley. By doing this, Wilma fulfills her goal of quenching her thirst. Formal planning is less common in AI research today, but planning approaches to narrative generation are still influential (Riedl and Young, 2010, e.g.).

Many of the original systems were interested in simulating the process of writing a story from the author's perspective. For instance, the system Author (Dehn,

1981) proposed that writing is a means of fulfilling an author's goals. The author invents details to justify events that they have already decided are part of the story. Dehn also notes, however, that writers should be willing to revise their goals if better goals emerge 'serendipitously' through the creative process. The system Universe (Lebowitz, 1985) formally encoded an author's goals (e.g. "maintain romantic tension") while generating television soap opera scripts. Similar to TALE-SPIN, goals are represented as formal planning structures that trigger plot fragments. In Minstrel (Turner, 1993b), which generated stories about King Arthur and his knights of the round table, unsatisfied author goals cause the system to query its *episodic memory* for story details that fulfill a particular goal. If some fragment of an existing story in episodic memory also fulfills the author's goal, that fragment is adapted to the current story. MEXICA (Pérez y Pérez and Sharples, 2001) applied the idea of creative writing as a cognitive process of engagement and reflection. During the *engaged* state, the system assembles sequences of story actions based on schemas learned from previous stories. Alternatively, during the *reflective* state, the system evaluates the coherence, novelty, and interestingness of those sequences so that revisions can be made when the system returns to the engaged state.

Some of this early research also considered the role of user interaction in story generation. TALE-SPIN was intended to be interactive in enabling the user to define characters' goals. In the proposed Virtual Storyteller system (Theune et al., 2003), the user serves as the "director" of a narrative. Before the story begins, the director specifies the characters and assigns personalities and goals to each one. Within the system these characters are encoded as agents, with formal rules specifying the characters' behavior according to the attributes the director has given them. The plot emerges from characters' interactions with one another. The

director can intervene in the story by introducing new characters, ascribing new goals to the characters, or disallowing a character to perform a particular behavior.

Alongside formal planning, case-based reasoning was recognized early on as a useful paradigm for narrative generation. Case-based reasoning is a general AI problem-solving approach where a new problem is solved by consulting a known solution for an existing problem (Aamodt and Plaza, 1994). Instead of encoding knowledge of a domain in terms of general rules, knowledge is composed of specific *cases* previously observed in that domain. In the context of narrative generation, case-based reasoning systems are used to establish an analogy between a new story and an existing story in the *case library*, so that the existing story can inform the generation of the new story. Minstrel (mentioned above) was one of the first examples of this approach. It implemented case-based reasoning as a *transform-recall-adapt* process. For instance, if the goal is to generate a story about a boy escaping a dragon, the system may *transform* the domain of this story into one where it can *recall* a similar story about a girl escaping a wolf by hiding in the woods, and then *adapt* this outcome so the boy similarly escapes the dragon by hiding. Case-based reasoning particularly lends itself to the opportunity for user interaction. For instance, Gervás et al. (2005) presented a scheme where a user can query a database of stories by specifying story attributes like characters, roles, and places, and the system retrieves the story that most closely fits these attributes. Going even further, case-based reasoning can support interactive narrative generation, where users contribute directly to the story creation process. The application Say Anything (Swanson and Gordon, 2012) discussed below in Section 2.3 is an effective example of this.

As explained above, most early narrative generation systems were more concerned with generating the underlying narrative structure of a story rather than

the text at its surface. Moreover, at the time when work on narrative generation first began, there was little accompanying research on natural language generation. Among the systems that did explicitly generate text, most implemented some form of what became known as template-based generation. One historical example of the template-based approach was the dialogue system ELIZA (Weizenbaum, 1966). As one of the first 'chatbots', ELIZA was a program that acted as a psychotherapist conversing with a patient (the user) via a text-based chat interface. When the user typed a statement, the system tried to match the statement to a template by detecting keywords like "you" and "me". Rewrite rules would then replace the words filling the template with new phrases. For instance, the user's sentence "I think you hate me" would be mapped to a template [(*) YOU (*) ME], where the wildcard * slots are respectively filled by "I think" and "hate". The rewrite rule for this template is [WHAT MAKES YOU THINK I (3) YOU], where the 3 indicates that the third element occupying the input template (e.g. "hate") should be inserted in this position, resulting in ELIZA's response: "What makes you think I hate you?". As with ELIZA, templates in the early narrative generation systems were written by hand, typically by the researchers designing the system. Each system employed a different set of templates specific to the stories being generated. For instance, MEXICA (identified above) uses templates as specific as [(X) FOLLOWED THE TRACE THROUGH THE FOREST AND FINALLY FOUND (Y) SWIMMING IN A BEAUTIFUL WATERFALL], where slots X and Y are filled by character names at generation time. The TALE-SPIN story shown in Section 2.1 is also an example of template-based generation.

The creators of MEXICA referred to the output of template-based generation as "computer-story language" to contrast it with the natural language of human-authored stories. Hand-authored templates are now seen as insufficient for large-scale narrative generation because of the intractable number of templates needed to generate stories that scale beyond one particular domain. But for systems where the scope of story generation is very limited, templates are appealing for their simplicity and ease of implementation. For instance, Tracery (Compton et al., 2015) allows developers of the interactive narratives described below in Section 2.3 to readily author their own templates for generating text. By filling these templates with words in a non-deterministic manner, these systems can produce juxtapositions whose accidental meaning is intriguing to users. This same concept of leveraging unpredictability for generation is explored in Chapter 8, but using the machine learning methods in this thesis rather than templates.

Even now, there is limited work on narrative generation as it relates to research perspectives in the field of natural language generation (NLG). NLG researchers have traditionally viewed generation as a process of three steps: content determination (what to say), text planning (how to say it), and production (presentation of language to user) (Stent and Bangalore, 2014). The main challenge of this paradigm has been in finding a way to automatically map structured representations of language to free text. Current story generation research shares this same challenge of aligning latent story knowledge modeled by traditional systems to its surface text realization.

2.2 Narrative Event Prediction

Following the pattern of AI research in general, narrative generation eventually shifted away from formal reasoning approaches to more data-driven methods like the ones in this thesis. In contrast to systems where knowledge about narrative structure is hand-authored, data-driven approaches seek to automatically acquire this structure. Existing work has focused on structure in terms of story events. Chambers and Jurafsky (2008) presented one of the first and most well-known efforts to automatically extract event sequences from stories. Their approach first applies NLP parsing to a corpus of stories so that each story is parsed into a series of verb-argument structures. For example, a sentence like "The cat furiously chased the dog down the street" could be encoded as CHASE(cat, dog). Within each story, verbs that reference the same arguments are extracted together as an event sequence, which they refer to as a *narrative event chain*. The coherence of sequences is scored by examining how often the events in the sequence occur together relative to how often each event occurs independently. The model can predict new events in a story by measuring the coherence of each potential new event with the events that have happened in the story so far. This work proposed a specific evaluation of this task, called the *narrative cloze test*. A narrative cloze is a sequence of narrative events where one event has been removed. The task is to predict the missing event based on the remaining sequence. They provide the following example:

> 1) McCann threw two interceptions early. (THREW SUBJECT)
>
> 2) Toledo pulled McCann aside and told him he'd start. (PULLED OBJECT, TOLD OBJECT, START OBJECT)

3) McCann quickly completed his first two passes. (COMPLETED SUBJECT)

Each sentence can consist of multiple events, and each event is represented by the verb-argument structures shown in parentheses. If the event THREW SUBJECT is omitted, for example, the system ranks its likelihood relative to a set of events taken from other stories. Because there is an extensive number of candidates, this task is not optimized for human performance. This shortcoming is addressed by the evaluation framework in Chapter 5 (the Story Cloze Test), which was inspired by the narrative cloze test.

Manshadi et al. (2008) performed the event prediction task by treating verb-argument structures as n-grams in a statistical language model. This work also applied an evaluation based on event ordering, where the system had to correctly distinguish between an event sequence found in a story and a random shuffling of the same set of events. Gerber et al. (2010) modeled temporality and causality between story events using discourse parsing to label adjacent clauses with CAUSE and RESULT relations, e.g. [$_{\text{CAUSE}}$ Packages often get buried in the load] [$_{\text{RESULT}}$ and are delivered late]. When presented with a new event, the system uses the labels of similar events in the corpus to infer these relations for the new event. McIntyre and Lapata (2009) extended this work by placing it in an interactive generation context. Their system inputs a user-specified topic and finds event sequences related to that topic using Chambers and Jurafsky's event coherence metric. These event sequences are transformed from verb-argument structures into natural language text by mapping each event to various sentence templates, and selecting the most probable sentence text for that event according to a language model. The system generates several stories pertaining to a topic, and then applies

a classifier that ranks them according to perceived interestingness and coherence based on linguistic features. An example of a generated story is:

> *The giant guards the child. The child rescues the son from the power. The child begs the son for a pardon. The giant cries that the son laughs the happiness out of death. The child hears if the happiness tells a story.*

Li et al. (2013) also generated stories given a topic provided by a user. This system uses crowdsourcing to elicit stories relevant to a particular story world (e.g. a bank robbery). Crowdsourced authors write the stories in natural language, but with some constraints: each sentence should convey one event, with a single verb per sentence, avoiding complex linguistic constructions like conditionals, compound sentences, and pronouns. The constrained syntax enables the system to cluster sentences across users writing about the same scenario, resulting in clusters that contain different sentence formulations describing the same event. The events for a story world are assembled into a graph based on their predicted ordering and mutual exclusion relations. Each possible traversal through the graph generates a story, resulting in many stories about bank robberies with different plots, for instance.

Granroth-wilding and Clark (2016) and Pichotta and Mooney (2016) have successfully applied neural methods to the prediction of structured event representations as evaluated by the narrative cloze test. Their main focus was not on generating natural language from these event representations. As can be seen from the examples above, the challenges of semantic parsing mean that verb-argument structures exclude a lot of information from the event representations. Many current techniques, for example, would represent the sentence "After her divorce,

Sandy spent a lot of time alone" as SPEND(SANDY, A LOT), which does not sufficiently capture the conveyed event. Future advancements in semantic parsing will inevitably yield much richer event representations, and thus better strategies for translating them to natural language. But for this thesis, I forgo these limited techniques in order to focus on frameworks where the input and output is unstructured natural language.

2.3 Interactive Storytelling

While many story generation systems are intended to produce stories autonomously, a great deal of research has focused on interactive storytelling. Interactive storytelling refers to digital applications where stories emerge through user input. Crawford (2012) gives an extensive review of this field, of which I will provide some highlights. Interactive storytelling actually began before the digital age, with one of its most recognized early forms being the text adventure novel. An example is the well-remembered Choose Your Own Adventure series (Packard, 1982, e.g.), where every few pages the author poses choices to the reader about how to proceed with the story. The author casts the reader as the protagonist by writing the text in the second person:

> *You are hiking in Snake Canyon when you find yourself lost in the strange, dimly lit Cave of Time. Gradually you can make out two passageways. One curves downward to the right; the other leads upward to the left. It occurs to you that the one leading down may go to the past and the one leading up may go to the future. Which way will you choose?*

Based on this choice the reader is directed to different pages with alternate progressions of the story, each of which continues to a further branching point. Another non-digital analogue to interactive storytelling is Dungeons and Dragons[1], a table-top role-playing game that engages players as characters in a fantasy narrative. The narrative relies on a set of rules that govern how players' choices for their characters' actions influence outcomes in the story. One player, the "Dungeon Master", acts as the operator of these rules by interpreting their application to the specific storyline of the game. The experience can be characterized as an oral version of a text adventure novel where each participant seeks a favorable outcome for their character in the story.

Text adventure novels were eventually translated to digital interfaces, where they became known as interactive fiction. See Montfort (2005) for a detailed account of how this medium evolved. An influential early example was Adventure[2], which was soon followed by Zork (Lebling et al., 1979). As with Dungeons and Dragons, these platforms are treated as games where players choose actions to pursue goals in the story. Here, the story is presented as text, but instead of turning pages as in a Choose Your Own Adventure book, the reader enters text specifying their choice of action. Figure 2.1 shows the interface for Zork. Player input appears on the lines beginning with >. The system processes player commands by mapping words to slots in hand-authored grammatical frames similar to the templates discussed in Section 2.1. Frames generally consist of a single verb and one or two objects. The interaction is very rigid; the example illustrates the disruptions that appear when the user input does not readily map to a frame or leaves slots in the frame open (e.g. "That's not a verb I recognise."). If the

[1] company.wizards.com

[2] rickadams.org/adventure/

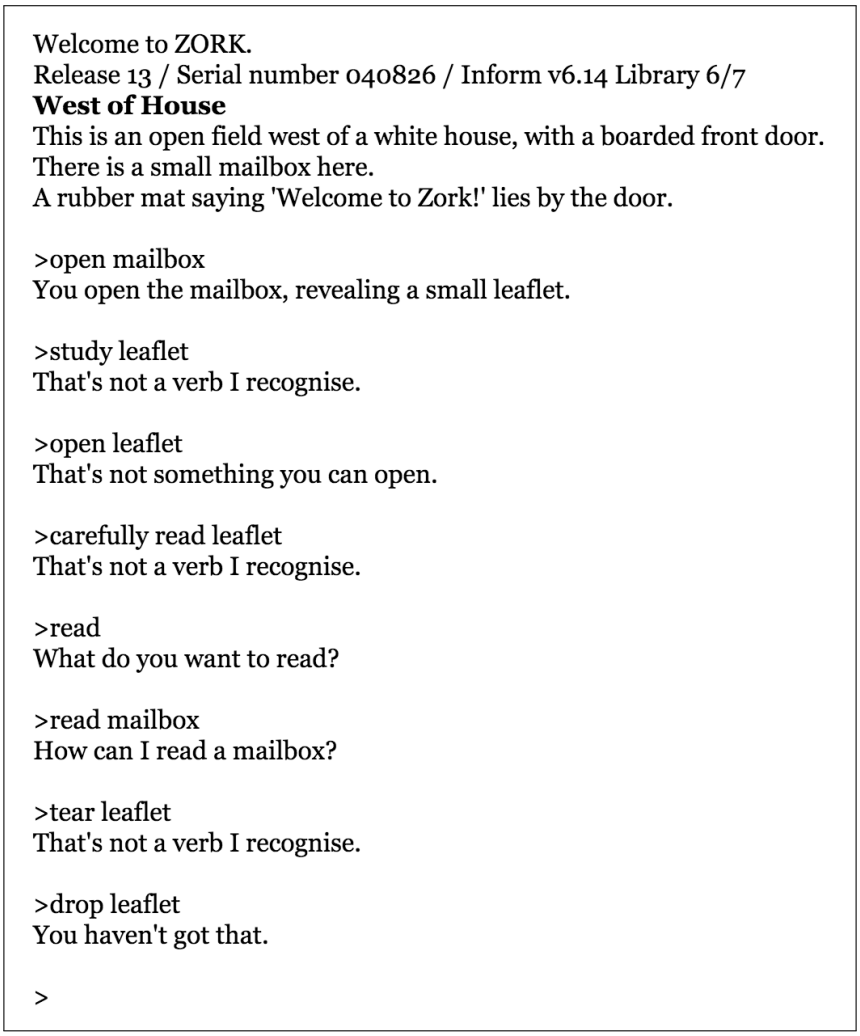

Figure 2.1: Text-based interaction in Zork

input is successfully parsed, the story appears to carry out the specified action by presenting the reader with a new text passage that prompts them towards another action in the story.

Interactive fiction reached the peak of its popularity in the 1980s, during which the software company Infocom produced over thirty works. With the expansion of graphical interfaces for games, interest shifted away from text-based interaction. Interactive fiction has continued to exist within a small community. Much of the popular work continued to adhere to Zork's same rule-based parser model

for interfacing with user text. This parser model was eventually built into the programming platform Inform[3], which has since been used to author several popular interactive fiction works[4]. Recently, a choice-based model analogous to Choose Your Own Adventure has been popularized with tools such as Twine[5] that allow non-programmers to author stories. Twine enables authors to create graphs of linked text passages, where links often act as branches to alternate stories that can be selectively explored by readers. Users choose from given text rather than typing their own input in order to progress the story. The vast majority of interactive fiction uses either the parser-based or choice-based architecture. Our work in Chapter 6 demonstrates an alternative paradigm where input is processed via statistical models instead of these deterministic procedures.

There have been several innovations in interactive storytelling research, focused on problems like character behavior generation (Cutumisu et al., 2006; Young et al., 2004), character believability (Mateas, 1999; Riedl and Young, 2010), modeling player decision-making (Sharma et al., 2010; Thue et al., 2007), drama management (Riedl et al., 2008; Roberts et al., 2006; Sharma et al., 2010), and enhancing users' sense of control (Porteous et al., 2010; Si et al., 2009). However, only a few of them have focused on expanding language-based interaction. Among these systems, NLP techniques have most commonly been applied to generating dialogue between characters (Cavazza and Charles, 2010; Rowe et al., 2008; van Deemter et al., 2008). There has been some effort to expand these techniques to engage with users. For instance, in Cavazza et al. (2002), users could intervene in the story by speaking simple commands to a speech recognition module (e.g. telling one character to

[3]inform7.com

[4]See the Interactive Fiction Database for a comprehensive list: ifdb.tads.org

[5]twinery.org

be nice to another character), causing changes in the representation of characters' goals. Façade (Mateas and Stern, 2003) was recognized as a breakthrough in enabling the user, playing the protagonist, to interact with story characters via natural language text. Developed over a two year period, Façade uses an extensive set of hand-authored templates to classify text input as one of several discourse acts (e.g. AGREE, CRITICIZE, APOLOGIZE), which are enacted by the protagonist to advance the story. However, because there is no systematic technique behind the authoring of these templates, they are only functional within Façade and do not readily generalize to other interactive storytelling systems. Still, Façade is hailed for providing users with a sense of agency over the story (Roth et al., 2011), which is attributed to the expanded capacity for language input.

The application Say Anything (Swanson and Gordon, 2012) was the first to interface with unconstrained language input in an open-domain environment. The term *open-domain* means that the system does not establish a particular 'story world', and instead elaborates on the one introduced by the user. This model of interactive storytelling is analogous to human-computer dialogue. Say Anything proceeds like a dialogue in that the user and system take turns telling a story. The user initiates the story by typing an introductory sentence. The system produces the next sentence to continue the story, and this turn-taking continues. No matter what the user types, the application will always contribute a sentence in return. This experience is very different from the one offered by interactive fiction. Instead of just triggering the progression of the story, here the user directs the story with the system playing a collaborative role. In other words, traditional interactive fiction is a reading-dominant experience, while Say Anything is a writing-dominant one. The technical approach behind Say Anything is explained in Chapter 7, where

the continuations it produces are compared to those generated by a neural network-based model.

2.4 Creative Language Generation

As stories are a form of creative text, work on free-text story continuation can clearly benefit from research on other types of creative language generation. Gatt and Krahmer (2017) note that work in this area has been limited due to little collaboration between researchers in natural language generation and those in computational creativity, but this has shifted more recently. There is an established trail of research on poetry generation (Das and Gambäck, 2014; Manurung et al., 2000; Oliveira, 2012). Recurrent Neural Network (RNN) models similar to the ones explored in this thesis (introduced in the next chapter) have driven recent work on this task. Both Wang et al. (2016) and Zhang and Lapata (2014) used RNN-based models to generate classical Chinese poems with complex structural constraints. Ghazvininejad et al. (2016) integrated an RNN with a finite-state machine to dynamically generate English-language poems from user input specifying the topic and style. Similarly, Potash et al. (2015) and Malmi et al. (2016) used neural models to generate rap lyrics. Other forms of creative language generation include metaphor generation, which Veale and Hao (2007) did by applying a case-based reasoning approach (Section 2.1) to generate metaphors from a user-provided term and a property associated with the term to be highlighted by the metaphor. Harmon (2015) extended this work by using a generate-and-rank approach to maximize desirable linguistic features according to lexical metrics. Other work in this space focuses on jokes and puns (Binsted and Ritchie, 1997; Stock and Strapparava, 2005), riddles (Tan et al., 2016), sarcasm (Joshi et al., 2017), and novel lexical

items (Deri and Knight, 2015; Zhang et al., 2014). Across all this work, there is no obvious approach to how generated content should be evaluated (Gatt and Krahmer, 2017), and the same is true for free-text story generation. Chapters 7 and 8 address this problem.

2.5 Automated Support for Writers

Up until forty years ago, the standard writing interface was ink and paper. The most important technological development for writing has been the digital word processor itself. For any writer accustomed to pen and paper, the opportunity to freely insert and delete text was revolutionary. Though it seems trivial now, research has confirmed that word processing software reduces writers' cognitive load of planning what they will write (Haas, 1989), leads writers to create longer texts and make far more revisions (Hawisher, 1989), and ultimately results in higher-quality writing (Bangert-Drowns, 1993). With the development of word processors was the idea that software could actively intervene in the writing process itself in order to improve the outcome. The earliest target of this support was automated spelling error detection and correction (Damerau, 1964; Kukich, 1992; Brill and Moore, 2000), which is now a standard "auto-correct" task. Limited versions of grammar correction are also built into many word processors. Not all grammatical errors are easy to detect, and this task remains an active area of research (Leacock et al., 2014). Systems have also addressed stylistic issues in writing by flagging language that may be considered overly repetitive or overly vague (Burstein and Wolska, 2003), encouraging the writer towards active rather than passive voice verbs (Macdonald et al., 1982), estimating the 'readability' of a text according to the length of its clauses (Reed, 1989), and suggesting synonyms

for words based on their context (Edmonds and Hirst, 2002). A lot of this work has been applied to educational objectives like automatically evaluating students' essays (Burstein et al., 2003). The success of commercial software that implements this research, such as Grammarly[6], demonstrates that people find these tools beneficial.

Most of these tools focus on the mechanics of writing rather than on developing content. But writers have shown interest in interfaces intended to stimulate their creativity. For example, OmmWriter[7] creates a relaxing, distraction-free writing environment by obscuring other applications on the writer's computer, and displaying pleasant colors and music to focus the writer's attention to the writing task. Scrivener[8] is currently one of the most popular applications among novel writers. It enables authors to curate different forms of inspiration for an emerging work, such as images, web links, and quotes from other published works. It also prompts the writer to set goals in terms of how many words to produce within a session. This is also the purpose behind WriteOrDie[9], which uses principles of operational conditioning to facilitate productivity. If the author writes too slowly or pauses too often during a writing session, the application punishes them by deleting words already present, playing an alarm sound, or displaying a fear-inducing image like a spider. Alternatively, if an author maintains a consistent pace of writing, they are rewarded with a pleasant image or sound.

[6] grammarly.com

[7] ommwriter.com

[8] literatureandlatte.com/scrivener/overview

[9] writeordie.com

The notion of an 'automated creative assistant' has generated a lot of philosophical discussion in AI (Boden, 2004; Candy, 1997; Kantosalo et al., 2014; Shneiderman, 2000). It has been demonstrated in creative domains like music (Huang et al., 2016a; Morris et al., 2008; Nichols et al., 2009, e.g.) and visual design (Davis et al., 2016; Koyama et al., 2017; Lee et al., 2011, e.g). Veale (2012) describes this same vision for using computers to enhance human creativity in writing. At the moment this is a largely unexplored research opportunity, but recent advances in creative language generation (Section 2.4) have made this vision more concrete. Interfaces for eliciting assistance from other human writers (i.e. collaborative writing) have provided a model for these technologies (Bernstein et al., 2010; Kim et al., 2014; Nebeling et al., 2016). For example, Settles (2010) and Watanabe et al. (2017) presented interfaces that help songwriters by automatically generating suggestions for lyrics. Gabriel et al. (2015) demonstrated a platform that makes automated revisions to poems that are customized to authors' detected personality traits and writing style. Systems developed by Puerta Melguizo et al. (2009) and Galitsky and Kuznetsov (2013) support authors by retrieving web content relevant to their writing in order to give them new ideas or help them refine what they have written. Llano et al. (2014) automatically generated abstract "what if" ideas to help authors brainstorm for new stories. In Chapter 8, I present an automated creative assistant that generates a natural language continuation of an author's story. Similar applications have emerged very recently (Clark et al., 2018; Khalifa et al., 2017; Manjavacas et al., 2017), which are further detailed in Chapter 8.

Chapter 3

Technical Overview

I do not fear computers. I fear the lack of them.

Isaac Asimov

Before discussing the specific models for the narrative continuation tasks in this thesis, this chapter provides a general description of how neural networks can be used to represent text. I largely focus on Recurrent Neural Networks (RNNs), and use the specific example of language modeling using RNNs to convey the fundamental concepts. For those already familiar with these concepts, this chapter can be skipped. The neural models applied in the subsequent chapters all make use of the techniques outlined here.

3.1 Neural Networks

A neural network is a machine learning framework that uses a set of mathematical functions to predict an output(s) from a given input(s). Neural networks are named based on the proposal that they model the activity of neurons in the human brain, but this is still only a theory with little empirical support. Still, neural networks have become essential to machine learning because of their ability to automatically fit complex patterns in the input variables to optimize prediction of the output variables. At an abstract level, a neural network can be visualized as a stack of *layers* connected by *weights*, as shown in Figure 3.1. These elements are all

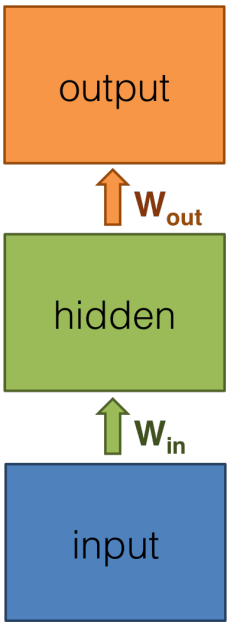

Figure 3.1: Abstract view of a neural network

represented as numerical matrices: each layer is the result of some configuration of matrix multiplications between the previous layer and the incoming weights, where each weight matrix has a predefined number of dimensions (nodes). The weight matrices W_{in} and W_{out} in Figure 3.1 have connections between every pair of nodes in the layers they adjoin. The weights facilitate transformations between the information going from one layer to the next, where each transformation results in a set of values that ultimately support the prediction of the output at the uppermost layer. The distinguishing property of a neural network is that the layers apply non-linear functions to the incoming values, which is what enables them to approximate complex relations between inputs and outputs. The non-linear intermediate layers between the input and output are referred to as *hidden* layers, since they contain a latent representation of the input features used for prediction. The functions

for the hidden and output layers in Figure 3.1 can be as simple as plain matrix multiplications:

$$\begin{aligned} hidden &= \sigma(input W_{in}) \\ output &= hidden W_{out} \end{aligned} \tag{3.1}$$

where σ is a non-linear function like the sigmoid function. The shape of the 2-D matrix W_{in} is the number of dimensions in the input variable by the number of hidden layer dimensions, which can be freely chosen. The idea is that since each dimension is a parameter in the model, more dimensions will result in better performance, but there is no specific guideline for what this number should be for a particular task. Most models in this thesis use 500 nodes, which is a common setting that enables relatively efficient training, since efficiency scales negatively with the number of nodes. The size of the matrix W_{out} is the number of hidden layer dimensions by the number of dimensions in the output variable. The neural network in Figure 3.1 is the type referred to as a feed-forward model, where the computations all move in the forward direction towards the output layer, as opposed to a recurrent model discussed in the next section. The behavior of the model depends on the weights, which are the parameters that are updated during training in order to optimize prediction. Prior to training these weights are initialized to small random values. During training, the model observes the inputs to compute a prediction for the output, and then receives feedback about the correct output in order to make an update to the weights. This feedback is given by an objective (loss) function which computes the error of the predictions made by the model; for example, the error may be the difference in the probabilities assigned by the model to the output variables relative to the true probabilities of the outputs as observed in the training data. Updates to the model during training aim to

minimize this error, which is facilitated by an optimization algorithm like Stochastic Gradient Descent (SGD) (Bottou, 2010). During SGD, the error is calculated periodically while iterating through the training instances, and the gradient of this error with regard to the weights indicates which direction the weights need to change in order to reduce the error. The gradient is computed through the backpropagation algorithm (Chauvin and Rumelhart, 1995; Rumelhart et al., 1988). There are various other optimization algorithms besides SGD that accomplish the same purpose, such as Adam (Kingma and Ba, 2015) and RMSProp (Tieleman and Hinton, 2012), which are utilized in this work. Training usually continues until the network has made numerous passes (epochs) through the training data, or until there is no significant further reduction in the error. To make training more efficient, the training data is divided into 'batches' of N input-output instances. The model observes all instances in a batch in parallel before updating its parameters, rather than iterating through the dataset one instance at a time. The details of both backpropagation and optimization algorithms are too expansive to outline here, so see the cited works for a thorough explanation.

3.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a specific type of neural network specifically used to model sequential data (Elman, 1990). In an RNN, the input is an order-specific sequence of values, and each hidden layer applies a recurrent function that iteratively computes a representation for a sequential unit based on the representation of the previous unit. Outputs are predicted according to this sequential information. Because language is composed of discrete units (such as words) that naturally occur in sequences (such as sentences), RNNs are particularly suitable for

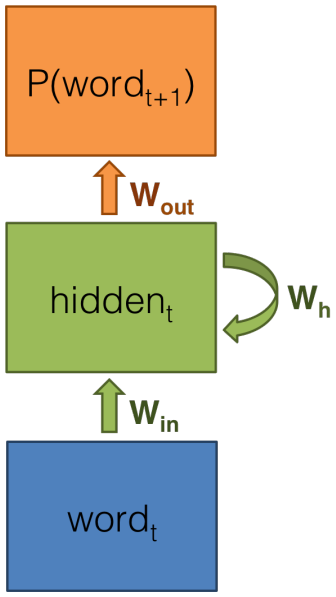

Figure 3.2: RNN Language Model

language processing tasks, and they are currently one of the most highly utilized machine learning frameworks in NLP.

A straightforward way to illustrate an RNN is as a language model. Language models are not dependent on any particular machine learning technique; a language model is any model that assigns a probability to a word sequence (see Jurafsky and Martin, 2014). A language model trained on English-language stories, for instance, should assign higher probability to the commonly observed sequence "Once upon a time" than the much rarer sequence "Once upon a book". In particular, if the language model observes "Once upon a", the conditional probability of the word "time" as the next word should be greater than that of "book". There are different methods for estimating this probability, one of which is to use an RNN, which is currently the most popular approach to language modeling (Chung et al., 2014; Mikolov et al., 2010; Sundermeyer et al., 2012). Figure 3.2 shows the basic architecture of an RNN language model (RNN LM), which looks very similar to Figure 3.1 with the addition of another set of weights associated with

the hidden layer (W_h). The network has an input layer that encodes a word in the sequence; a recurrent hidden layer that "memorizes" the words observed previously in the sequence and integrates this information with the current word; and an output layer that contains the predicted probabilities of all possible words that could appear next in the sequence. The same concepts introduced in 3.1 apply here: these layers are connected by weight matrices that transform values via non-linear functions. The view in 3.2 is the model at a given timepoint t in a sequence, where each timepoint corresponds to a word. In the simplest case, the input $word_t$ is a vector whose number of dimensions equals the number of words in the model's lexicon, which is chosen before the network is initialized. For a given word in a sequence, the vector will contain zeros in every dimension except for the dimension designated to that word, where the value is one (this is referred to as a *one-hot* vector). Alternatively, it is common for words to be represented as distributed vectors of real values called *embeddings*, which are discussed further in Section 3.3. In either case, $word_t$ is passed to the hidden layer via the weight matrix W_{in}, whose size is the number of input dimensions (i.e. the lexicon or word embedding size) by the number of hidden nodes. The novelty of the hidden layer in an RNN is that it includes weights W_h that loop back to itself, which means it can retain information across words in the same sequence. The values for this layer are referred to as the *state* of the model. Thus, when a word is encoded into the hidden layer via W_{in}, the existing hidden state is updated via the recurrent weights W_h, and the results of both computations are combined. This new state $hidden_t$ is the network's representation of the sequence that has been observed so far. These values are projected to the output layer by applying the weight matrix

W_{out}, which has a dimension for each word in the lexicon. Specifically, $hidden_t$ and $output_t$ can be computed as:

$$hidden_t = \tanh(word_t W_{in} + hidden_{t-1} W_h) \tag{3.2}$$

$$output_t = hidden_t W_{out} \tag{3.3}$$

The values in $output_t$ can be "compressed" between 0 and 1 by applying the softmax function, which normalizes them by their total sum. Consequently, each value represents the estimated probability of the corresponding word w in the lexicon W being the next word that appears in the input sequence:

$$P(word_{t+1}) = softmax(output_t) = \frac{\exp(output_t^w)}{\sum_{v=1}^{V} \exp(output_t^v)} \text{ for } w = 1, \ldots, W \tag{3.4}$$

Figure 3.3 shows an 'unrolled' view of the model in Figure 3.2 to exemplify the representation of the sequence "Once upon a time, there...". As indicated, the output for a given $word_t$ in this sequence is the probability of the word that follows it: $P(word_{t+1})$.

An interesting property of neural networks is that they can be stacked together, resulting in a model with several hidden layers (hence the term *deep learning*), where the input to a layer is the output of the previous layer. For example, the output $hidden_t$ could be used as input for a new layer $hidden_t^2$ that applies Equation 3.2 again with new weight matrices: $hidden_t^2 = \tanh(hidden_t W_{in^2} + hidden^2{}_{t-1} W_{h^2})$. Equation 3.3 is then applied to the topmost hidden layer to produce the output.

RNNs are trained according to the same general procedure outlined in Section 3.1. For an RNN LM, the training objective is to maximize the probabilities of

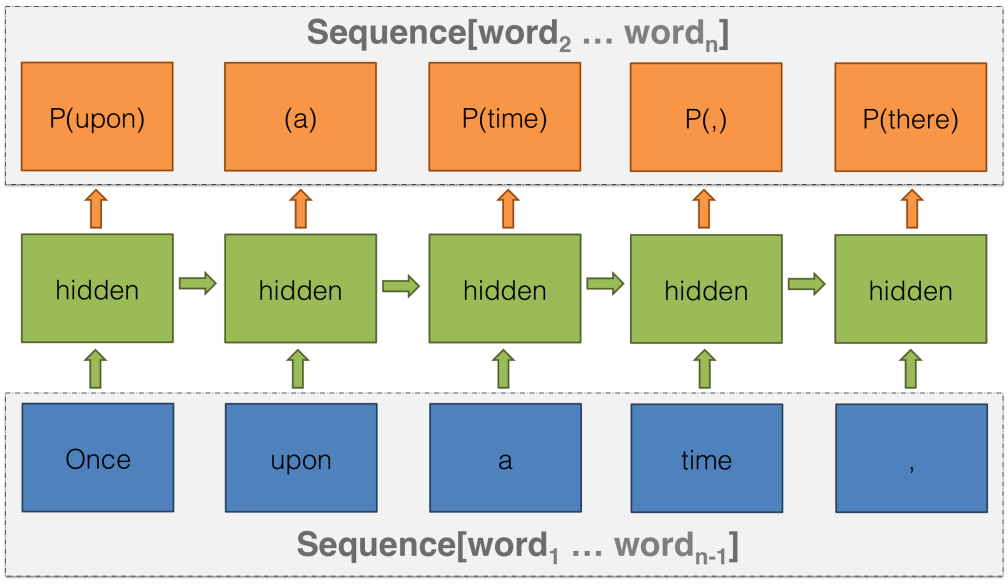

Figure 3.3: Unrolled view of RNN LM with an example sequence

the words that actually appear next in the sequences observed during training (the true words). To do this, cross-entropy can be used as the error function, which calculates the average difference between the true probability distribution and the probability distribution predicted by the model. For example, if $true_t^w$ if the correct probability of a word w at timepoint t in a sequence and $predicted_t^w$ is the predicted probability of that same word, then cross-entropy can be calculated as:

$$error = -\frac{1}{|W|} \sum_{w \in W} true_t^w \log(predicted_t^w) \tag{3.5}$$

where W includes all words in the lexicon. This is equivalent to the mean of the negative log probability predicted by the model for all true words. This function is negative because the objective of training is to minimize this value, such that lower values indicate more accurate predictions. The above section discussed optimization and backpropagation, which are applied the same way to training

RNNs. RNNs specifically make use of the Backpropagation Through Time algorithm (BPTT) (Werbos, 1990) to compute the gradient of the error across all timepoints in order to update the weights.

3.2.1 Gated Recurrent Units

In theory, RNNs have no limitation on the length of sequences they can model. However, the RNN LM has a better memory for words that have recently occurred in the sequence as opposed to words several prior timepoints away. Of course, this is desired behavior for a language model, as words depend heavily on the local context in which they appear. The traditional approach to language modeling with n-grams relies only on the previous few words for predicting the next word. A fair amount of the time, this is enough for the model to simulate knowledge of syntactic rules: for example, a well-trained language model should not assign high probability to a sequence like "Once upon a are" because it would have been very unlikely to observe a determiner ("a") immediately followed by a *be* verb ("are") in standard English text. On the other hand, many linguistic dependencies are separated by several words in a sequence, and this becomes a challenge when going beyond syntax to model the semantics of language. For instance, in the sequence "Hal was walking his dog one morning. A squirrel ran across their path. Hal's dog strained so hard, the leash broke! He ran towards the", most English speakers would expect the word "squirrel" to appear next. Assigning it high probability requires a model to remember its observation of "squirrel" two sentences back. An ideal language model should be able to do this, and this is what research on RNN LMs has aimed for.

RNNs may run into what is known as the *vanishing gradient* problem, where the gradient computed during backpropagation becomes so small that the model

stops learning these longer-range dependencies (Bengio et al., 1994). There are now RNN architectures that are observed to be more robust to this vanishing gradient problem. The two more well-known ones are Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014). It has been demonstrated that both these variants are better at remembering longer sequences, according to intrinsic evaluation metrics like cross-entropy (Jozefowicz et al., 2015; Sundermeyer et al., 2012). Theoretically, this is enabled by their mechanism for selectively memorizing only certain information about a sequence and "forgetting" what is no longer needed. This mechanism occurs in the way the hidden state of the network is calculated. The hidden state relies on the previous state and the current input as specified in Section 3.2, but it uses an additional set of equations to determine which values to maintain when updating the hidden state. Specifically, as an alternative to Equation 3.2, the hidden state of a GRU layer is computed as:

$$\begin{aligned} reset_t &= \sigma(word_t W_{in\_r} + hidden_{t-1} W_{h\_r}) \\ update_t &= \sigma(word_t W_{in\_u} + hidden_{t-1} W_{h\_u}) \\ hidden_t &= update_t \odot hidden_{t-1} + (1 - update_t) \\ &\quad \odot \tanh(word_t W_{in} + (reset_t \odot hidden_{t-1}) W_h) \end{aligned} \tag{3.6}$$

where $\odot$ represents elementwise multiplication. This method introduces two new components to the hidden layer, an update gate and a reset gate. Both gates modulate the amount of information carried over in the hidden state from one timepoint to another. Intuitively, $reset_t$ controls how much information to include from the previous hidden state; $update_t$ controls how much information to include in the next hidden state. Note that when the reset gate is 1 and the update gate is 0, the result is equivalent to the simple recurrent layer defined by Equation 3.2.

All RNN models presented in this thesis specifically use GRU units, but I will typically refer to them with the general term RNN.

3.3 Text Processing with Neural Networks

Here is an example of how text is processed in an RNN LM, which is similar to the process we employ for our models. First, the text in a training corpus is tokenized into individual words[1]. In English, tokenization largely corresponds to white spaces between words, with some exceptions. Punctuation marks, for instance, are tokenized as separate words. Once the text is represented as a sequence of discrete words, each unique word type is entered into a lexicon (vocabulary). As the number of words in this vocabulary grows, so does the size of the input layer and consequently, the number of parameter weights to be learned by the network. In order to ensure that the network can learn efficiently, there must be some limit to the number of words in the vocabulary. The standard solution is to set a frequency threshold such that only words that occur at least as many times as that threshold are added to the lexicon. If the threshold is three, for instance, then words that appear less than three times in the data will not appear in the lexicon. Instead, these words are mapped to a generic <UNKNOWN> word category so that all these infrequent words have the same representation in the network. To convert words from text form to numeric form, each word is assigned a unique numerical index. This index indicates the non-zero dimension of the one-hot vector for that word (e.g. the vector for a word with an index of 2 has a value of 1 in its second position and zeros everywhere else). A very common technique in NLP is to use distributed N-dimensional vectors of real values to represent words, known as

[1]Most of the NLP processing in this thesis, including sentence and token segmentation, part-of-speech tagging, and syntactic chunking, was done using the spaCy library: spacy.io/

embeddings. These embeddings can be learned directly within a model by adding an additional layer to the model above the word input layer, where the weights associated with this layer are a 2-D matrix whose size is the number of words in the lexicon by N dimensions. Thus, the values associated with each word dimension in this matrix correspond to its embedding. Just as with the number of hidden layer units, the value of N can be freely chosen; it is common to see it set between 100-300. Like all other weights the word embeddings are updated during training to optimize the model predictions.

Alternatively, it is also possible to utilize embeddings from models specifically trained to optimize the representation of words. Words with similar meanings are expected to have similar vector representations. For example, techniques like word2vec (Mikolov et al., 2013) learn to encode words by predicting other words that appear nearby in the same sequence. This is based on the hypothesis of distributional semantics, which theorizes that words observed in similar contexts often have similar meanings. The resulting embeddings can be plugged into a model for a particular task to provide the model with some semantic information about words. For a language model, for instance, this may make it easier for the model to generalize predictions for a word in a particular sequence to similar sequences. For example, the sequences "I went to the hospital because I felt" and "I visited the doctor because I was" contain words with similar vectors and thus would be expected to have some similarity in their hidden state representations in an RNN LM. This enables the model to predict that "sick" is a probable next word in both sequences, even if one of the sequences has not been directly observed in the corpus. Word embeddings are now a standard framework across all NLP tasks, and we utilize them extensively throughout the work in this thesis.

In this thesis, I apply a few different variations of neural models to narrative text, all of which borrow from the same concepts introduced in this chapter. Chapter 4 presents an *encoder-decoder* approach that learns a mapping between temporally related sequences in a story. Chapter 5 demonstrates a classifier that predicts the likelihood of a particular ending for a given story by learning to distinguish between correct and incorrect endings, using an RNN to encode the sentences in the story. Chapter 6 evaluates feed-forward and recurrent models for predicting discrete categories associated with outcome sequences in a user-driven interactive narrative, comparing them to simpler machine learning techniques. Finally, Chapters 7 and 8 employ a version of the RNN LM depicted here for the purpose of free-text story continuation. The general design of each model is conveyed in those chapters, but their technical details can be referenced here.

Chapter 4

Choice of Plausible Alternatives

The human brain has evolved the capacity to impose a narrative, complete with chronology and cause-and-effect logic, on whatever it encounters, no matter how apparently random.

Robin Marantz Henig

In the first part of this thesis, I examine the task of closed-choice narrative continuation: predicting 'what happens next' in a story when a fixed list of candidates is given. I take advantage of some recent evaluation schemes that have emerged for this task, which have drawn new attention to story generation research. In Roemmele et al. (2011), I presented one such framework I developed, the Choice of Plausible Alternatives (COPA), which focuses on commonsense reasoning about events that are causally related. It uses a binary-choice format to elicit a prediction for either the cause or effect of a given event, where all events are communicated in natural language. COPA can easily be framed as a narrative prediction task. It is highly related to the Story Cloze Test featured in the next chapter: both involve predicting what is likely to follow in a prototypical real-life scenario according to commonsense expectations. The distinction is that the Story Cloze Test involves predicting the ending of a story, whereas COPA items involve predicting relations

between pairs of sentences not explicitly presented in a narrative context. Moreover, COPA emphasizes not just events that are likely to temporally co-occur in a scenario, but those where one is the likely cause of the other. As mentioned in Chapter 1, this causal coherence between events is a prominent feature of narrative text. Accordingly, following previous work on COPA by Gordon et al. (2011), we leverage story corpora for this task.

As will be detailed in the next chapter, the state-of-the-art approaches to the Story Cloze Test involve neural networks trained to distinguish between correct and incorrect endings of stories (Cai et al., 2017; Schwartz et al., 2017b). A neural approach has not yet been applied to COPA, so this is what we pursue in the current work. In particular, we evaluate a neural encoder-decoder model that predicts the probability that a particular sequence follows another in a story. Our experiments explore a few different variables for configuring this approach. First, we examine how to extract temporally related sequence pairs provided as input to the model. Second, we vary the use of feed-forward versus recurrent layers in the architecture of the model. Third, we assess different vector-based representations of the sequence pairs. Finally, we compare our model using different narrative corpora for training, including a newly released corpus of stories (ROCStories) related to the Story Cloze Test. Our results are presented in comparison to existing systems applied to COPA, which involve lexical co-occurrence statistics gathered from web corpora. Our best-performing model achieves an accuracy of 66.2% on the COPA test set, which falls short of the currently known state-of-the-art[1] of 71.2% (Sasaki et al., 2017). Interestingly, this best result utilizes the ROCStories corpus for training, which is several orders of magnitude smaller than the datasets used in existing approaches. Applying our model to these larger datasets actually

[1]As of February 2018

yields significantly worse performance, suggesting that the model is sensitive to the density of commonsense knowledge contained in its training set. We conclude that this density is far more influential to COPA performance than just data quantity, and further success on the task will depend on methods for isolating commonsense knowledge in text.

4.1 Task Design

The Choice of Plausible Alternatives (COPA) is composed of 1,000 items, where each item contains three sentences, a premise and two alternatives, as well as a prompt specifying the relation between them. The items are divided equally into development and test sets of 500 items each. The goal is to select which alternative conveys the more plausible cause (or effect, depending on the prompt) of the premise sentence. Half of the prompts elicit the more plausible effect of the premise event, while the other half ask for the more plausible cause of the premise.

1. **Premise:** The homeowners disliked their nosy neighbors. *What happened as a result?*

   Alternative 1:* They built a fence around their property.

   Alternative 2: They hosted a barbecue in their backyard.

2. **Premise:** The man fell unconscious. *What was the cause of this?*

   Alternative 1:* The assailant struck the man in the head.

   Alternative 2: The assailant took the mans wallet.

Above are examples of COPA items, where the correct alternative is starred. For all items, both alternatives refer to temporally related events that could be found within the same story, but the correct one is intended to convey a more

coherent causal relation. I was the single author of COPA questions. This process was one of guided intuition, where I manually examined events from a story corpus as candidate premise sentences for COPA items. I looked for causes or effects of a premise (as plausible alternatives) as well as events either independently co-occurring with or in direct opposition to that event (as implausible alternatives). All sentences consisted of a single clause with a past tense verb, and proper nouns were avoided. Each item was then validated by two other raters to ensure human accuracy was as close to 100% as possible. See Roemmele et al. (2011) for further details about the authoring and validation process.

4.2 Existing Approaches

4.2.1 Pointwise Mutual Information (PMI)

In Roemmele et al. (2011), we presented a baseline approach to COPA that focused on lexical co-occurrence statistics computed from story corpora. The general idea is that a causal relation between two story events can be modeled by the proximity of the words that express the events. This approach counts the number of times two words co-occur within the same context, i.e. within a certain N number of words of each other in a story. For a given corpus, these word pair frequencies along with the overall frequency of each word can be used to compute an overall score for the causal association between two sequences. To do this, each word in the first sequence is paired with each word in the second sequence. Then the Pointwise Mutual Information (PMI) statistic (Church and Hanks, 1990) is calculated for each word pair $(w1, w2)$ based on how often the words occur in the same context relative to how often they occur independently:

$$PMI(w1, w2) = \frac{count(w1, w2)}{count(w1) * count(w2)} \tag{4.1}$$

In this work, PMI is an asymmetric measure in that only instances of $w1$ followed by $w2$ are counted as co-occurrences, not instances where their order is reversed. This captures the order of causality in the sequence pair, based on the assumption that causes more often appear before their effects in stories relative to the reverse. Given the PMI scores for each pair of words in the sequences $S1$ and $S2$, causality is a sum of those scores normalized by the length of each sequence:

$$causality(S1, S2) = \frac{\sum_{w1 \in S1} \sum_{w2 \in S2} PMI(w1, w2)}{|S1| * |S2|} \tag{4.2}$$

For a given COPA item, the predicted alternative is the one that has the higher causality score with regard to the premise. Since the scores are asymmetric is assuming $S1$ is the cause of $S2$, COPA items that elicit the more plausible effect assign the premise and alternative to $S1$ and $S2$ respectively, whereas this assignment is reversed for items that ask for the cause of the premise.

Using this model, Gordon et al. (2011) gathered word counts from a corpus of one million stories extracted from personal weblogs. These stories were largely non-fiction stories about daily life events written from the first-person perspective. A co-occurrence between two words was counted when they appeared within 25 words of one another in the same story. This model obtained 65.2% accuracy on the COPA test set.

4.2.2 Causal Net

The PMI approach assumes a causal relation between events can be captured to some degree by their temporal co-occurrence in a story. Luo et al. (2016)

introduced a variation that alternatively focuses on explicit mentions of causality in a given corpus. To do this, they extracted sequences matching lexical templates that signify causality, e.g. A leads to B, B results from A, B due to A, where A is the *cause* event and B is the *effect* event. Each cause word a in sequence A is paired with each of the corresponding effect words b in sequence B, and these pairs are counted as co-occurrences in the same manner as the PMI approach. Instead of applying the simple PMI equation (Equation 4.1) to a word pair, however, they propose a measure of the *causal strength* between them. This measure distinguishes between *necessary* causality CS_{nec} and *sufficient* causality CS_{suf}:

$$\begin{aligned} CS_{nec}(a,b) &= \frac{p(a,b)}{p^{\alpha}(a) * p(b)} \\ CS_{suf}(a,b) &= \frac{p(a,b)}{p(a) * p^{\alpha}(b)} \\ CS(a,b) &= CS_{nec}(a,b)^{\lambda} * CS_{suf}(a,b)^{1-\lambda} \end{aligned} \tag{4.3}$$

where $p(a,b)$ is the normalized frequency of the word pair in the corpus, and α is a constant. Intuitively, CS_{nec} captures the degree to which a must appear in order for b to appear, while CS_{suf} captures the degree to which a alone will result in the occurrence of b. These two factors are combined into a single measure of causal strength for a word pair, where the weight of each factor can be adjusted according to the parameter λ. The overall causality between two sequences is scored as the average causal strength between all pairs of words in the sequences, analogous to Equation 4.2.

Luo et al. applied this approach to extract causal pairs from a corpus of approximately 1.6 billion web pages. They achieved 70.2% accuracy on the COPA test

set, significantly outperforming the existing PMI result. Sasaki et al. (2017) evaluated the same approach on a smaller corpus of web documents, ClueWeb[2], which contains 700 million pages. They discovered that including multiword phrases in the causal pairs boosted accuracy to 71.2%. Both results indicate that causal knowledge can be extracted from large web data as an alternative to story corpora. Rather than assuming that causality is implicitly conveyed by temporally related sequences, they relied on explicit mentions of causality to filter data relevant to COPA. Still, a lot of causal knowledge in stories is not templated. Consider the sequence "John starts a pot of coffee because he is sleepy", for example. This sequence would be extracted by the CausalNet approach since it contains one of the designated lexical markers of causality ("because"). However, the sequence "John is sleepy. He starts a pot of coffee" expresses the same causal relation and yet would not be captured by the templates. Even though the text does not explain that "sleepy" and "coffee" are causally related, we know by people's ability to answer COPA items that they can readily infer this. Using a large web corpus can possibly compensate for missing these instances, since there will still be many that convey the same relations using explicit mentions of causality. However, it still means that a lot of causal information is potentially being overlooked.

4.2.3 Other Approaches

COPA was a shared task in the 6th International Workshop on Semantic Evaluation (SemEval 2012) (Gordon et al., 2012). There, Goodwin et al. (2012) presented a linear classifier that predicted COPA alternatives based on features associated with bigram PMI counts, temporal links given by the knowledge representation

[2]lemurproject.org/clueweb12/

framework TimeML (Pustejovsky et al., 2003), causal dependencies given by manually crafted patterns, and sentiment polarity. Their best result obtained 63.4% test accuracy. Jabeen et al. (2014) used the existing PMI model to detect verbal patterns associated with causality, which resulted in 58.8% accuracy when applied to a corpus of Wikipedia texts.

Other work associated with COPA includes Blass and Forbus (2017), who proposed an *analogical chaining* approach to prediction, which uses case-based reasoning based on formal knowledge structures. This work focuses on demonstrating a deep semantic representation of COPA items rather than evaluating accuracy across the entire COPA test set. Additionally, Zhang et al. (2016) incorporated COPA items into a dataset where the candidate answers for commonsense-related questions are annotated with ordinal judgments about their plausibility.

4.3 Neural Network Approach

As mentioned above, our work initiates the exploration of neural approaches to COPA. We focus here on an *encoder-decoder* architecture. Originally applied to machine translation (Cho et al., 2014), encoder-decoder models have been extended to other sequence modeling tasks like dialogue generation (Serban et al., 2016; Shang et al., 2015) and poetry generation (Ghazvininejad et al., 2016; Wang et al., 2016). We propose that this technique could be similarly useful for our task in establishing a mapping between cause-effect sequence pairs. This direct modeling of co-occurrence between sequences is unique from the previous work, which relied on co-occurrence between pairs of individual words.

4.3.1 Sequence Segmentation

The inputs and outputs for the encoder-decoder model are each word sequences. Given a corpus of stories as the training set for a model, we first segmented each story by clausal boundaries. This was done heuristically by analyzing the dependency parse of each sentence. Words whose dependency label was an adverbial clause modifier (ADVCL; e.g. "After I got home, I *got* a text from her."), conjunct (CONJ; "I dropped the glass and the glass *broke*."), or prepositional complement (PCOMP; "He took me to the hospital to *seek* treatment.") were detected as the heads of clauses distinct from the main clause. All contiguous words dependent on the same head word were segmented as a separate clause. These particular labels do not capture all clausal boundaries (for example, relative clauses are not detected), but they are intended to distinguish sequences that may refer to separate narrative events (e.g. "I dropped the glass" is segmented from "and the glass broke"). This is somewhat analogous to the segmentation performed by Luo et al. (2016) which splits cause and effect clauses according to lexical templates. The difference is that the parsing labels we use for segmentation do not explicitly refer to causality. We did not perform an intrinsic evaluation of this procedure in terms of how often it correctly segmented narrative events. Instead, we evaluated its impact on the prediction task by comparing it to traditional segmentation based on sentence boundaries for the same model.

4.3.2 Sequence Pairs

After segmenting the stories, we joined neighboring segments (i.e. clauses or sentences) into pairs. We manipulated the temporal window within which these pairs were joined, by pairing all segments within N segments of each other. For a given segment at position t in a story, pairs were established between all segments in

$segment_t, \ldots, segment_{t+N}$. For example, when N=1, a pair was formed with the next segment only $(segment_t, segment_{t+1})$; when N=2, pairs were formed between $(segment_t, segment_{t+1})$ and $(segment_t, segment_{t+2})$. By doing this, we intended to examine the proximity of causal information in a story according to its impact on COPA prediction. Gordon et al. (2011) analogously evaluated this by varying the number of words within which PMI word pairs were formed, regardless of sentence or clause boundaries.

4.3.3 Encoder-decoder Models

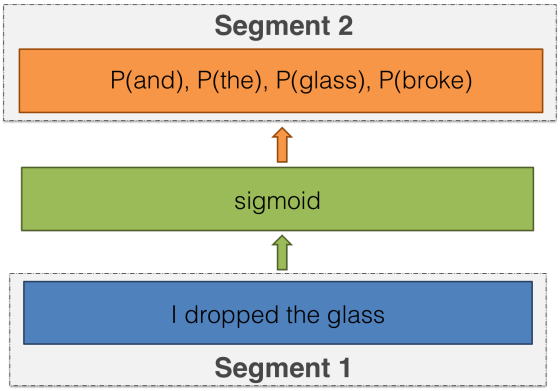

Figure 4.1: FFN encoder-decoder model

We examined two types of encoder-decoder models: one with feed-forward (FFN) layers and one with recurrent (RNN) layers. The latter is often referred to as a sequence-to-sequence model (Sutskever et al., 2014). See Chapter 3 for the technical details of feed-forward and recurrent neural networks. The theoretical motivation for comparing these models is to determine the importance of accounting for word order in the input and output sequences in terms of its effect on COPA prediction. The existing COPA approaches only accounted for word order to the extent of capturing word pairs within the same context of N words (though Sasaki et al. (2017) also recognized commonly occurring multi-word expressions).

The FFN model, displayed in Figure 4.1, ignores word order. It is very simple: both the input and output segments are collapsed into flat n-dimensional vectors of word counts, so the hidden (encoder) layer observes all words in each segment in parallel. On the output (decoder) layer (which has sigmoid activation like the encoder), the FFN computes a score for each word that indicates the probability that that word will appear anywhere in output segment.

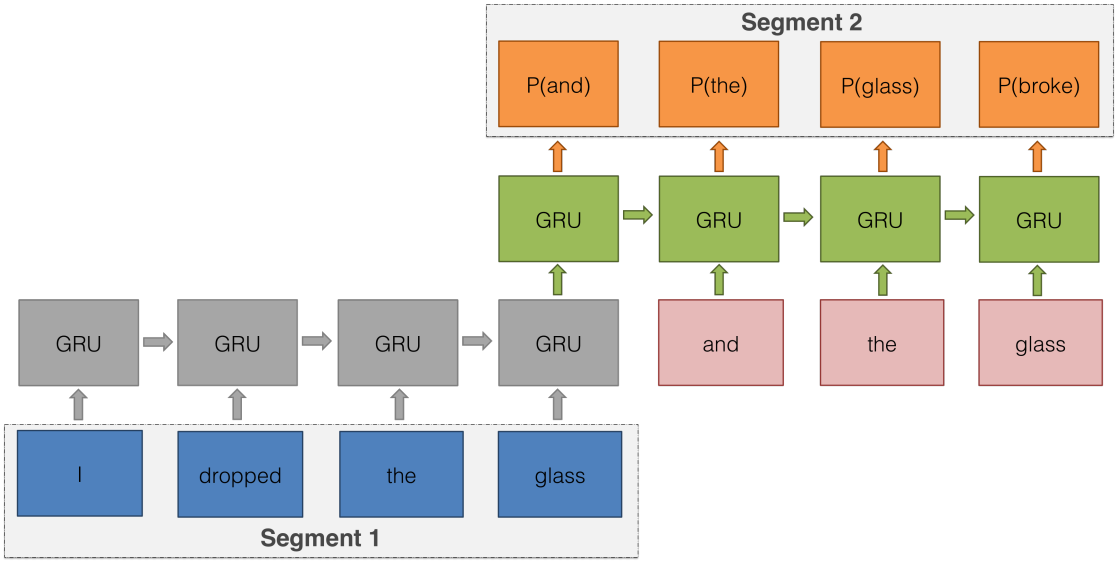

Figure 4.2: RNN encoder-decoder model

In contrast, the RNN (Figure 4.2) accounts for the order of words in the segments. In particular, it uses a recurrent (encoder) layer with GRU units to iteratively encode the input sequence, and another recurrent (decoder) layer to represent output segment. The final hidden state of the encoder layer after observing the whole input is provided as the initial hidden state to the decoder. The decoder then iteratively computes a representation of the output sequence that is conditioned upon the input segment. For each timepoint in this decoder layer, a dense softmax layer is applied to predict a probability distribution over each word being observed in the segment at that particular timepoint. This model is a variation of

the RNN language model illustrated in Chapter 3 (Figure 3.3), with the distinction that the input and output sequences here are represented by different RNN layers, so the entire input sequence is observed before the model predicts the output sequence. Both the FFN and RNN encoder-decoders are trained through the same procedure outlined in Chapter 3, using the cross-entropy loss function to maximize the output word probabilities observed during training.

4.4 Initial Experiments

4.4.1 ROCStories Corpus

The PMI and CausalNet approaches to COPA made use of large web corpora. Based on the proposal in Gordon et al. (2011) that stories are a good source for the commonsense knowledge needed to answer COPA questions, we examined a recently created dataset of stories that has yet to be utilized for COPA prediction, the ROCStories corpus[3] (Mostafazadeh et al., 2016). This corpus is also central to work described in Chapter 5. It consists of 97,027 five-sentence narratives authored via crowdsourcing. In contrast to weblog stories, these stories were written with the specific objective to minimize discourse complexity and explicate prototypical causal and temporal relations between events in salient everyday scenarios. COPA items also target these latent commonsense relations, so the ROCStories appear to be particularly suitable for this domain. Table 4.1 shows some examples of stories in this corpus and corresponding COPA items that address the same causal knowledge. A disadvantage of the ROCStories corpus is that it is notably smaller than the datasets used in previous work, which consist of millions of texts.

[3] cs.rochester.edu/nlp/rocstories

ROCStories Story	COPA Item
Susie went away to Nantucket. She wanted to relax. When she got there it was amazing. The waves were so relaxing. Susie never wanted to leave.	**Premise:** The man went away for the weekend. What was the cause of this? **Alt 1*:** He wanted to relax. **Alt 2:** He felt content.
Albert wanted to enter the spelling bee, but he was a bad speller. He practiced every day for the upcoming contest. When Albert felt that he was ready, he entered the spelling bee. In the very last round, Albert failed when he misspelled a word. Albert was very proud of himself for winning the second place trophy.	**Premise:** The girl received a trophy. What was the cause of this? **Alt 1*:** She won a spelling bee. **Alt 2:** She made a new friend.
Anna was lonely. One day, Anna went to the grocery store. Outside the store, she met a woman who was giving away kittens. Anna decided to adopt one of those kittens. Anna no longer felt lonely with her new pet.	**Premise:** The woman felt lonely. What happened as a result? **Alt 1:** She renovated her kitchen. **Alt 2*:** She adopted a cat.
April is fascinated by health and medicine. She decided to become a doctor. She studied very hard in college and medical school. April graduated at the top of her medical school class. April now works in a hospital as a doctor.	**Premise:** The woman wanted to be a doctor. What happened as a result? **Alt 1:** She visited the hospital. **Alt 2*:** She went to medical school.

Table 4.1: Examples of stories in ROCStories corpus and similar COPA items

4.4.2 Procedure

We applied the methodology outlined in Section 4.3 to pairs of sequences from the ROCStories corpus. Our first set of experiments varied segmentation (clause versus sentence boundaries), distance between segments (N=1 to N=4), and the type of encoder-decoder (FFN or RNN). Note that N=4 is the maximum setting when using sentence boundaries since there are five sentences in each story, so here pairs will be formed between all sentences in the story. For all experiments, we filtered all grammatical words (i.e. all words except for adjectives, adverbs, nouns, and verbs) and lemmatized all segments in the pairs, consistent with Luo et al.

(2016). COPA items intentionally do not contain proper nouns, so we excluded them as well. In order to map each word in the segments to a discrete word index, we assembled a lexicon that included each word occurring at least five times in the corpus, which totaled 9,299 words in the ROCStories corpus. All other words were mapped to a generic <UNKNOWN> token.

The encoder and decoder layers of the FFN and RNN models each consisted of 500 dimensions. The RNN had an additional word embedding layer of 300 nodes in order to transform discrete word indices in the input segments into distributed vectors. They were both trained for 50 epochs using the Adam optimizer with a batch size of 100 pairs. After each epoch, we evaluated the model on the COPA development set and saved the weights that obtained the highest accuracy on these items.

4.4.3 Results

Table 4.2 shows the results of these different configurations in terms of COPA accuracy. We include the results on the development set as a reference because they tended to vary from the test results. Most notably, the FFN outperformed the RNN universally, suggesting that the order of words in the segments did not provide a strong signal for prediction beyond the presence of the words themselves. Among the FFN results, the model trained on clauses with N=4 obtained the highest accuracy on the development set (66.0%), and was tied for the highest test accuracy with the model trained on clauses with N=3 (66.2%), though this result was only trivially better than the test result for the sentence segment model with N=4 (66.0%). The model with N=4 was trained on three times as many pairs as the model with N=1, which perhaps contributes to the former obtaining better results. Still, because of the improvement, we can probably assume that some

causal relations are distributed across segments that are not immediately adjacent. The impact of clause segmentation is less clear from these results, given that the 66.2% accuracy using clause units is only trivially better than the corresponding result with sentence segmentation (66.0% for N=4).

			FFN		RNN	
Segment	**N**	**# Pairs**	**Dev**	**Test**	**Dev**	**Test**
Sentence	1	389,680	64.8	64.4	63.4	54.4
	2	682,334	65.2	65.4	61.2	57.6
	3	877,963	63.8	63.8	60.2	55.4
	4	976,568	63.8	66.0	59.4	55.6
Clause	1	539,342	64.2	63.6	59.4	56.8
	2	981,677	65.2	65.0	59.2	54.6
	3	1,327,010	65.4	**66.2**	63.4	58.0
	4	1,575,340	66.0	**66.2**	61.2	56.6

Table 4.2: Accuracy by segmentation unit and pair distance (N) for the FFN and RNN encoder-decoders trained on the ROCStories corpus

4.4.4 Other Findings

Alternative Input Representations

Model	**Dev**	**Test**
FFN (above)	66.0	**66.2**
FFN GloVe	65.0	61.6
FFN ConceptNet	61.6	62.4
FFN Skip-thought	66.8	63.8

Table 4.3: Accuracy of FFN trained on ROCStories with different input representations

In the FFN model evaluated above, the input segments were simply represented as bag-of-words vectors with the frequency of each word contained in the segment. The model internally learns the distributed vector representation of these

segments in the hidden layer. Alternatively, the segments can be represented in terms of word embeddings learned by a separately trained model, as explained in Chapter 3. Using pretrained embeddings gives models some initial information about relations between words that may facilitate them to more readily learn how they are causally related. Moreover, they may be useful for recognizing similarity between stories that use different words to express similar concepts. For example, knowledge contained in a story about a dog chasing a squirrel can generalize to a story about a cat chasing a mouse, which may be captured by the similarity of the embeddings between "dog" and "cat" (and their analogous relations to "squirrel" and "mouse"). We experimented with three sets of embedding representations. First, we encoded the words in each input segment as the sum of their GloVe embeddings[4] (Pennington et al., 2014), which represent words according to a global log-bilinear regression model trained on word co-occurrence counts in the Common Crawl corpus. We also did this using ConceptNet embeddings[5] (Li et al., 2013), which apply the word2vec skip-gram model (Mikolov et al., 2013) to tuples that specifically define commonsense knowledge relations (e.g. soak in hotspring CAUSES get pruny skin). Lastly, we used skip-thought vectors[6] (Kiros et al., 2015), which compute one embedding representation for an entire sentence and thus represent the sentence beyond just the sum of its individual words. Analogous to how word embedding models are trained to predict words near a given target word in a text, the skip-thought vectors are trained by predicting nearby sentences. The provided model is trained on the BookCorpus dataset[7], which we

[4]nlp.stanford.edu/projects/glove/

[5]ttic.uchicago.edu/ kgimpel/commonsense.html

[6]github.com/ryankiros/skip-thoughts

[7]yknzhu.wixsite.com/mbweb

also use as a training set for the encoder-decoder in our subsequent experiments described below.

We trained the FFN model on the ROCStories corpus with each of these three sets of embeddings. Because they obtained the best performance in the previous experiments, we configured the models to use clause segmentation and distance N=4 in constructing the pairs. Table 4.3 shows the results of these models, compared alongside the best result from above with the standard bag-of-words representation. Neither the GloVe nor ConceptNet word embeddings performed better than the bag-of-words vectors (61.6% and 62.4% test accuracy, respectively). The sentence (skip-thought) vectors performed better than the bag-of-words representation on the development set (66.8%), but this did not scale to the test set (63.8%).

Phrases

Model	**Dev**	**Test**
FFN (above)	66.0	**66.2**
FFN Phrases	62.6	64.8

Table 4.4: Accuracy of FFN trained on ROCStories with explicit phrase representations

As mentioned above, Sasaki et al. (2017) found on that modeling multi-word phrases as individual words was helpful for the CausalNet approach. The RNN encoder-decoder has the opportunity to model phrases by capturing sequential dependencies between words, but Table 4.2 indicated this model was not successful relative to the FFN model. To assess whether the FFN model would benefit from phrase information, we merged all phrases in the training corpus into individual word tokens in the same manner as Sasaki et al., using their same list of

phrases. We again filtered all tokens that occurred fewer than five times in the data, which resulted in the vocabulary increasing from 9,299 words to 10,694 when the phrases were included. We trained the same FFN model in Table 4.2 that achieved the best result (clause segmentation, N=4, and the simple bag-of-words input representation). The test accuracy, relayed for clarity in Table 4.4 alongside the above best result, was 64.8%, indicating there was no benefit to modeling phrases in this particular configuration.

Comparison with Existing Approaches

Model	**Dev**	**Test**
FFN (above)	66.0	**66.2**
PMI	60.0	62.4
CausalNet	50.2	51.8

Table 4.5: Accuracy of PMI and CausalNet trained on ROCStories

To establish a comparison between our encoder-decoder approach and the existing models applied to the same dataset, we trained the PMI model on the ROCStories corpus. In this case, rather than using a fixed word window, we computed the PMI counts for all words in a story, which generally corresponds to using distance N=4 among sentence segments in the encoder-decoder model. Table 4.5 shows that this approach had 62.4% test accuracy, so our new FFN encoder-decoder approach outperformed it on this particular dataset. For completeness, we also applied the CausalNet approach to this dataset. Its poor performance (51.8%) is unsurprising, because the lexical templates used to extract causal pairs only matched 4,964 sequences in the ROCStories. This demonstrates that most of the causal information contained in these stories is conveyed implicitly.

4.5 Experiments on Other Datasets

Gordon et al. (2011) found that the PMI approach trained on personal stories in blogs yielded significantly better COPA accuracy than the same model trained on books in Project Gutenburg[8], despite that the latter had far more text than the former. Beyond this, there has been limited exploration of the impact of different training datasets on COPA prediction, so we were motivated to examine this in this work. Thus, we applied our FFN encoder-decoder model to the following datasets:

Visual Storytelling (VIST): 50,200 five-sentence stories[9] authored through crowdsourcing in support of research on vision-to-language tasks (Huang et al., 2016b). Participants were prompted to write a story from a sequence of photographs depicting salient "storyable" events.

CNN/DailyMail corpus: 312,085 bullet-item summaries[10] of news articles, which have been used for work on reading comprehension and automated summarization (Chen et al., 2016; See et al., 2017).

CMU Book and Movie Plot Summaries (CMU Plots): 58,862 plot summaries[11] extracted from Wikipedia, which have been used for story modeling tasks like automatically inferring attributes and relations of characters (Bamman et al., 2013; Srivastava et al., 2016).

BookCorpus: 8,032 fiction novels uploaded by authors to the website smashwords.com (link provided above; full corpus contains 11,000 books). Our subset included books from a variety of genres, including Adventure, Fantasy, Historical

[8] gutenberg.org/

[9] visionandlanguage.net/VIST/

[10] github.com/danqi/rc-cnn-dailymail

[11] cs.cmu.edu/~ark/personas/; cs.cmu.edu/~dbamman/booksummaries.html

Fiction, Horror, Mystery/Thriller, Romance, Science Fiction, and Young Adult Fiction. This dataset is also associated with the skip-thought sentence vectors evaluated in Section 4.4.4.

Blog Stories: 1 million weblog stories used in the COPA experiments by Gordon et al. (2011) identified above.

ClueWeb Pairs: Approximately 150 million sequence pairs extracted from the ClueWeb corpus using the lexical templates method in Sasaki et al. (2017). We utilized the pairs in this dataset directly as they were given (i.e. the first clause in each pair was the input segment, the second clause was the output segment).

4.5.1 Procedure and Results

Dataset	**# Pairs**	**Dev**	**Test**
ROCStories-Half	762,130	64.0	62.6
VIST	854,810	58.2	49.2
ROCStories-Full	1,575,340	66.0	**66.2**
CNN/DailyMail	3,255,010	59.4	51.8
CMU Plots	6,094,619	57.8	51.0
ClueWeb Pairs	157,426,812	60.8	61.2
Blog Stories	222,564,571	58.4	57.2
BookCorpus	310,001,015	58.2	55.0

Table 4.6: Accuracy of the FFN encoder-decoder on different datasets

We trained the FFN model with the best-performing configuration from the ROCStories experiments (clause segmentation, N=4, bag-of-words input representation). After determining that the lexicon used in the previous experiments included most of the words (93.5%) contained the COPA development set, we re-used this same lexicon to avoid the inefficiency of assembling a new one for each separate corpus. We also trained a model on the initial 45,502 stories in

the ROCStories corpus (ROCStories-Half) to further analyze the impact of this dataset.

Table 4.6 shows the results for these datasets compared alongside the ROCStories result from above (ROCStories-Full). They are listed in ascending order of the number of training pairs they contain. As shown, none of the other datasets reach the level of test accuracy of the model trained on the ROCStories (66.2%). Even the model trained on only the initial half of this corpus outperforms the others (62.6%). The next closest result is for the ClueWeb Pairs, which had 61.2% accuracy despite containing 100 times the number of pairs as the ROCStories dataset. The ClueWeb pairs obtained 71.2% accuracy when used in the CausalNet approach, so the encoder-decoder model is apparently not as effective in utilizing the causal knowledge contained in this dataset. The larger Blog Stories and BookCorpus datasets also did not have much impact, despite that the Blog Stories obtained 65.2% accuracy in the PMI approach. One speculative explanation for this is that our approach is highly dependent on the *density* of COPA-relevant knowledge contained in a dataset. As mentioned above, authors of the ROCStories were instructed to emphasize the most obvious possibilities for 'what happens next' in prototypical scenarios. These expectations align with the correct answers to COPA questions. In naturally occurring stories, these obvious occurrences are considered boring; often stories are salient for describing events that violate commonsense expectations (Schank and Ableson, 1995). Thus, they may show greater diversity in 'what happens next' relative to the ROCStories. Consider this sequence taken from the CMU Plots corpus:

> *Beginning several months after the events in Blade Runner, Deckard has retired to an isolated shack outside the city, taking the replicant Rachael with him in a Tyrell transport container, which slows down the*

replicant aging process. He is approached by a woman who explains she is Sarah Tyrell, niece of Eldon Tyrell, heiress to the entire Tyrell Corporation and the human template (templant) for the Rachael replicant.

This story certainly references commonsense knowledge that makes it interpretable, but it looks very different from the ROCStories examples in Table 4.1. For one thing, there is more syntactic complexity. But it also contains a lot of knowledge that is not particularly informative for COPA: for example, that Rachael is a replicant, the Tyrell transport container slows down replicant aging, and Sarah Tyrell is the replicant template for Rachael. The amount of non-commonsense knowledge in these stories may have been more distracting for our encoder-decoder architecture than for the previously evaluated approaches. Despite all being related to narrative, the VIST, CNN/DailyMail, and CMU Plots datasets were also ineffective on the test set with regard to this model. In general, there is a large gap between the development and test accuracy, which was also observed in the ROCStories results. It is likely there is an overfitting effect here, especially since there are only 500 items in the COPA development and test sets, respectively. These items were authored to address a wide breadth of commonsense scenarios, which could result in sparsity that limits generalizability between items.

4.6 Conclusion

In summary, we pursued a neural encoder-decoder approach for predicting 'what happens next' in the COPA framework. To our knowledge this is the first work to evaluate a neural model for this task. Our best model obtained 66.2% COPA accuracy. This is lower than the current state-of-the-art at 71.2%, but our experiments point to some opportunities for future work. We demonstrated the usefulness of the

ROCStories corpus for this task, as our model appeared to benefit from its density of commonsense knowledge. The gap between 66.2% and 71.2% is not dramatic in light of the massive size difference between the datasets. It does not seem likely that the CausalNet result can be scaled to human-level performance just by training on an even larger web corpus. At the same time, the ROCStories corpus is a crowdsourced dataset and thus will not grow naturally over time, so it may not be practical to rely exclusively on these types of specially authored resources either. The CausalNet approach proposed a useful way to isolate commonsense knowledge in text by relying on lexical cues, but because so much information about causality is not marked by specific lexical items, it still overlooks a lot that is relevant to COPA. Automatically detecting more latent linguistic features associated with the expression of causal knowledge in text is a significant research challenge. Our experiments with clause segmentation addressed this to some degree by examining syntactic boundaries between events that may be causally related. More advanced approaches to delineating related story events could have more impact.

COPA is a general evaluation framework for research on commonsense reasoning through natural language understanding. Here, I examined it through the lens of closed-choice narrative prediction. In the next chapter, I address a related framework, the Story Cloze Test, which focuses more explicitly on narrative by eliciting a prediction for the most plausible ending of a given story.

Chapter 5

The Story Cloze Test

Every story is organic, and every story finds its own ending.

T. C. Boyle

In the previous chapter, I addressed the task of modeling commonsense causality in text, which I examined as an instance of closed-choice narrative continuation. In this chapter, I focus on a related closed-choice prediction task, the Story Cloze Test[1] introduced by Mostafazadeh et al. (2016). The Story Cloze Test arose recently both out of new interest in story modeling as an NLP task, as well as the need to establish standard evaluation resources in this domain. In contrast to COPA where predictions are made from single sentences that are not explicitly framed within a narrative context, the Story Cloze Test is specifically a story continuation task. I describe a set of approaches for performing this task, based on our work in Roemmele et al. (2017b). Our best result obtains 67.2% accuracy on the test set, outperforming the top baseline of 58.5% presented by Mostafazadeh et al. This approach is below the currently published state-of-the-art[2] of 75.2% achieved by Schwartz et al. (2017b), but it provides some interesting insights into the task and also has the opportunity to scale beyond the current result, as I discuss below. I first report the results of some models that were evaluated on the other closed-choice prediction tasks in this thesis. I then illustrate our main approach

[1] cs.rochester.edu/nlp/rocstories

[2] As of August 2017

that achieves the best result, which uses a recurrent neural network (RNN) with a binary classifier to distinguish correct story endings from artificially generated incorrect endings. We compare the performance of this model when alternatively trained on different story encodings and different strategies for generating incorrect endings.

5.1 Task Design

In the Story Cloze Test, given the beginning four sentences of a story, the task is to choose which of two given sentences best completes the story. Like COPA items, the stories convey a variety of commonsense causal and temporal relations between events from which the best ending can be predicted. Also akin to COPA, the binary-choice format enables performance to be evaluated straightforwardly in terms of prediction accuracy. Released along with the Story Cloze Test was a corpus of stories (referred to as the ROCStories corpus, introduced in the previous chapter) from which cloze items were derived. All items were authored through the crowdsourcing process detailed in Mostafazadeh et al. (2016), in which authors were instructed to write five-sentence stories about everyday life experiences that convey stereotypical expectations about the events contained in these scenarios. For the cloze items, authors observed the first four sentences of a given story (the context) and wrote a new 'correct' ending and 'incorrect' ending for the story. Table 4.1 in Chapter 4 showed some examples of stories in the ROCStories corpus. Here, Table 5.1 shows some examples of the accompanying Story Cloze Test items. There are 97,027 narratives in the ROCStories corpus and 3,742 items in the Story Cloze Test divided equally between development and test sets.

Initial Story (Context)	Correct Ending	Incorrect Ending
Rick grew up in a troubled household. He never found good support in family, and turned to gangs. It wasn't long before Rick got shot in a robbery. The incident caused him to turn a new leaf.	He is happy now.	He joined a gang.
Laverne needs to prepare something for her friend's party. She decides to bake a batch of brownies. She chooses a recipe and follows it closely. Laverne tests one of the brownies to make sure it is delicious.	The brownies are so delicious Laverne eats two of them.	Laverne doesn't go to her friend's party.
Sarah had been dreaming of visiting Europe for years. She had finally saved enough for the trip. She landed in Spain and traveled east across the continent. She didn't like how different everything was.	Sarah decided that she preferred her home over Europe.	Sarah then decided to move to Europe.
Ignacio wants to play a sport while he is in college. Since he was a good swimmer, he decides to try out for swim the team. Ignacio makes it onto the team easily. At the first swim meet, Ignacio wins second place!	Ignacio won a silver medal.	Ignacio gave up swimming.

Table 5.1: Examples of Story Cloze Test items

5.2 Initial Approaches

Mostafazadeh et al. (2016) applied several unsupervised baselines to the Story Cloze Test. We began our exploration of this task by evaluating three additional approaches borrowed from other narrative prediction tasks.

Average Maximum Similarity (AveMax): The AveMax model is a slight variation on a baseline presented in Mostafazadeh et al. that selects the candidate ending with the higher average word2vec embedding similarity to the initial story. It is currently implemented to predict story continuations from user input in the DINE application (Bellassai et al., 2017), which is detailed in the

next chapter. Instead of selecting the candidate ending most similar to the context in terms of their respective word embeddings, this method iterates through each word in the ending, finds the word in the context with the most similar embedding, and then takes the mean of these maximum similarity embeddings: $\frac{\sum_{word1 \in context} \max_{word2 \in ending} sim(word1, word2)}{|context|}$. Intuitively, this favors endings that have exact word overlap with the context. We evaluated this method using both word embeddings from the GoogleNews dataset and the ROCStories corpus (see 5.3.1 below for details on these embeddings).

Pointwise Mututal Information (PMI): The PMI model was introduced in the last chapter, where it was shown to have moderate success on COPA. To review, this model relies on lexical co-occurrence counts to compute a 'causality score' for how likely one sentence is to follow another in a story. We applied the same approach to the Story Cloze Test to select the final sentence with the higher causality score of the two candidates. We evaluated word counts from two different sources: the Blog Stories corpus of one million stories extracted from personal weblogs (as was used in Gordon et al., 2011) and the ROCStories corpus.

Encoder-decoder: We applied the neural feed-forward (FFN) encoder-decoder model that was featured in Chapter 4 as our best-performing approach on COPA. This model was adapted to the current task by encoding the initial sentences of a story as the input sequence and the corresponding ending sentence as the decoded output sequence. We again used the ROCStories corpus for training.

5.2.1 Results

Table 5.2 shows the accuracy of these approaches on both the development and test sets of the Story Cloze Test. The AveMax model with the GoogleNews embeddings (55.2% test accuracy) performs comparably to the word2vec similarity model in

Mostafazadeh et al. (53.9%). The PMI approach performs at the same level as the current best baseline of 58.5%, and the counts from the ROCStories are just as effective (59.9%) as those from the much larger Blog Stories corpus (59.1%). Despite that the encoder-decoder model trained on the ROCStories performed better on COPA than the PMI model, this was not the case for this task (58.4% accuracy for the encoder-decoder). Based on these results, we explored alternative approaches to this task more targeted towards specifically predicting the ending of a story.

	Dev	**Test**
AveMax		
GoogleNews WordEmb	0.553	0.552
ROC WordEmb	0.548	0.547
PMI		
Blog Stories	0.585	0.591
ROCStories	0.581	**0.599**
Encoder-decoder	0.584	0.584

Table 5.2: Accuracy of initial approaches on the Story Cloze Test

5.3 Binary Classification Approach

Because the encoder-decoder approach did not perform very far above the 50% random baseline, we pursued an alternative neural architecture for the Story Cloze Test. Specifically, we investigated an approach based on binary classification of the correctness/incorrectness of endings. As described, while the Story Cloze Test provides a correct and incorrect ending to choose from, the ROCStories corpus only contains the correct ending for a given story. So our strategy was to create a new training set with binary labels of 1 for correct endings (positive examples) and 0 for incorrect endings (negative examples). Each story in the corpus was considered

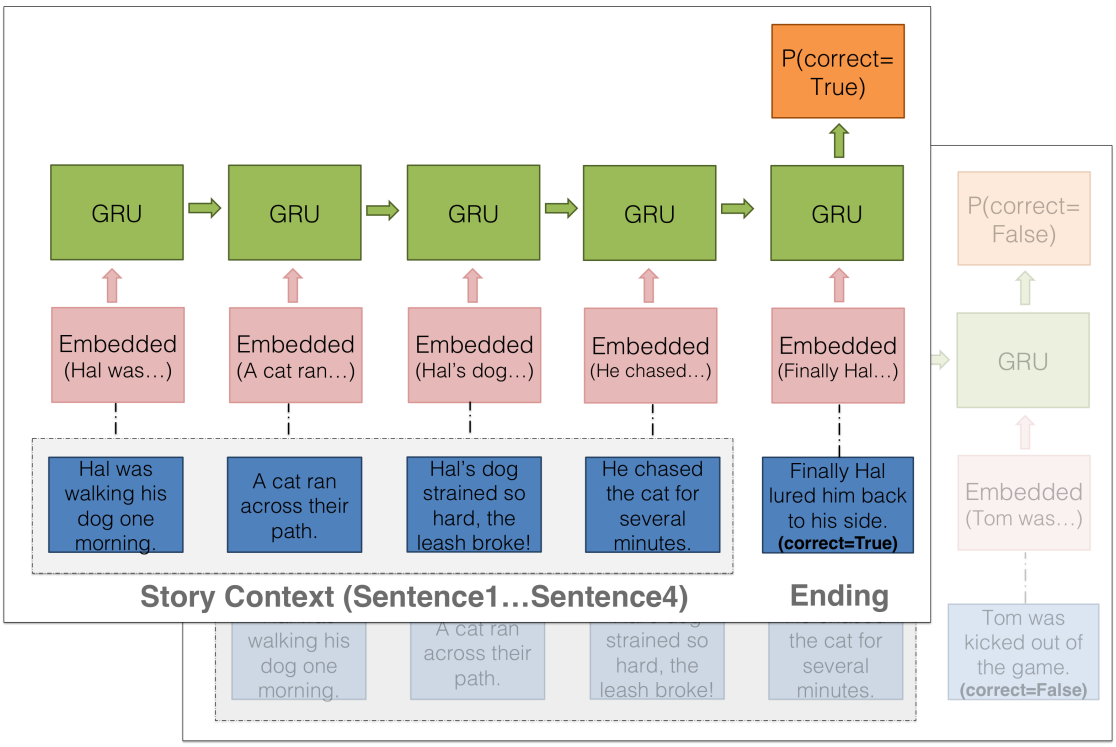

Figure 5.1: RNN-based Binary Classifier for the Story Cloze Test, with an example of a positive input (correct ending) and a negative input (incorrect ending)

a positive example. Given a positive example, we generated a negative example by replacing its final sentence with a designated incorrect ending. As described below, we generated more than one negative ending per story, so that each positive example had multiple negative counterparts. Our methods for generating negative examples are defined below in Section 5.3.2. Our approach was to train a classifier to distinguish between these positive and negative examples.

The binary classifier is integrated with an RNN. Figure 5.1 shows the general architecture of the model. It takes the context sentences and ending for a particular story as input and then returns the probability of that ending being correct, using the ending labels as feedback during training. The diagram shows an example of a correct ending and incorrect ending. To clarify, a single input consists of a context

and only one ending with its label (0 for correct=False and 1 for correct=True), rather than the model observing both the incorrect and correct endings for a given context at the same time. We append the ending to the context sentences, and then transform each of the five sentences into a vectorized (embedded) representation as conveyed in the next section. Each sentence is sequentially fed as a timestep into a single 1000-node GRU (Cho et al., 2014) hidden layer. The values of the final hidden state are given to a top feed-forward layer composed of one node with sigmoid activation. The potentially surprising aspect of this design is that the ending is encoded as the final timestep in the RNN that reads the context. An alternative architecture would be to represent the ending in a separate hidden layer, and then have the top layer compare the context and ending representations to predict the correctness of the latter. Instead, the top layer only observes the final GRU state after reading the entire story including the ending, so the GRU is designed to model the correctness of the transition from the context to the ending. A binary cross-entropy objective function is applied to train the network to maximize the probability of positive examples being correct. All experiments used RMSProp (Hinton et al., 2012) with a batch size of 100 to optimize the model over 10 training epochs. After training, given a cloze test item, the model predicted a probability score for each candidate ending, and the ending with the higher score was selected as the response for that item.

5.3.1 Story Representations

We employed distributed vector representations of story sentences in our models, as we explored in Chapter 4. This was further motivated by the top performing baseline in Mostafazadeh et al. that used embeddings to select the candidate story ending with the higher cosine similarity to its context.

Word Embeddings: We first tried encoding stories with word-level embeddings using the word2vec model (Mikolov et al., 2013). We compared two different sets of vectors: 300-dimension vectors trained on the 100-billion-word GoogleNews dataset[3] (referred to below as GoogleNews WordEmb) and 300-dimension vectors that we trained on the ROCStories corpus itself (ROC WordEmb). The latter were trained using the gensim word2vec library[4], with a window size of 10 words and negative sampling of 25 noise words. All other parameters were set to the default values given by the library. By comparing these two sets of embeddings, we intended to determine the extent to which our models can rely only on the limited training data provided for this task. In our classification experiments we averaged the embeddings of the words in each sentence, resulting in a single vector representation of the entire sentence.

Sentence Embeddings: The second embedding strategy we used was the skip-thought model (also applied in Chapter 4), which produces vectors that encode an entire sentence. We evaluated two sets of sentence vectors: 4800-dimension vectors trained on the 11,000 books in the BookCorpus described in Chapter 4 (BookCorpus SentEmb), and 2400-dimension vectors we trained[5] ourselves on the ROCStories corpus (ROC SentEmb). Mostafazadeh et al. also used the BookCorpus vectors in a baseline that measured vector similarity between the story context and candidate endings (which obtained 55.2% accuracy).

Context	Correct	Type	Incorrect
Hal was walking his dog one morning. A cat ran across their path. Hal's dog strained so hard, the leash broke! He chased the cat for several minutes.	Finally Hal lured him back to his side.	**Rand**	Tom was kicked out of the game.
		Back	A cat ran across their path.
		Near	His dog had to wear a leg cast for weeks.
		Near	His dog is too fast and runs off.
		Near	Rod realized he should have asked before petting the dog.
		LM	When she woke up, she realized he had no dog noises.
		LM	When he got to the front, he saw a dog, squirrel, and dog.
		LM	When he got to the front office, he found a cat in the ditch.
John woke up sick today. He washed his face in the bathroom. John went into the kitchen to make some soup. He put a bowl of soup into the microwave.	John dropped the soup when he grabbed it from the microwave.	**Rand**	She waited for months for her hair to grow back out.
		Back	He put a bowl of soup into the microwave.
		Near	Dan returned to the couch and watched a movie with his snack.
		Near	The doctor gave him medicine to get better.
		Near	Finally, he ate it.
		LM	He brushed his teeth and ate it for a while, he was sad.
		LM	He put the bowl in his microwave, and went to the kitchen.
		LM	He brushed her teeth, but the candles didn't feel so he didn't have any.

Table 5.3: Examples of generated negative endings

5.3.2 Incorrect Ending Generation

We examined four different ways to generate the incorrect endings for the classifier.

Table 5.3 shows examples of each.

[3]code.google.com/archive/p/word2vec

[4]radimrehurek.com/gensim/models/word2vec.html

[5]We used the same code and default parameters available at github.com/ryankiros/skip-thoughts

Random (Rand): First, we simply replaced each story's ending with a randomly selected ending from a different story in the training set. In most cases this ending will not be semantically related to the story, so this approach would be expected to predict endings based strictly on lexical overlap with the context.

Backward (Back): The Random approach generates negative endings with divergent lexical items from the context. However, these examples may not represent the items in the Story Cloze Test, where the endings generally both have some degree of semantic coherence with the context sentences. To generate negative examples in the same semantic space as the correct ending, we replaced the fifth sentence of a given story with one of its four context sentences (i.e. a backward sentence). This results in an ending that is semantically related to the story, but is typically incoherent given its repetition in the story.

Nearest-Ending (Near): The Nearest-Ending approach aims to find endings that are very close to the correct ending by using an ending for a similar story in the corpus. Swanson and Gordon (2012) presented this model in an interactive storytelling system, as outlined in Chapter 2. Given a story context, we retrieved the most similar story in the corpus (in terms of bag-of-words cosine similarity), and then projected the final sentence of the similar story as the ending of the given story[6]. Multiple endings were produced by finding the N most similar stories. The negative examples generated by this scheme can be seen as 'almost' positive examples with likely coherence errors, given the sparsity of the corpus. This is in line with the cloze task where both endings are plausible, but the correct answer is more likely than the other.

Language Model (LM): Separate from the binary classifier, we trained an RNN language model (Mikolov et al., 2010) on the ROCStories corpus. As detailed

[6]We used the gensim library to build a similarity index: radimrehurek.com/gensim/

in Chapter 3, the LM learns a conditional probability distribution indicating the chance of each possible word appearing in a sequence given the words that precede it. We gave the LM the context of each training story and had it produce a final sentence by sampling words one by one according to the predicted distribution. Multiple sentences were generated for the same story by sampling the N most probable words at each timestep. The LM had a 200-node embedding layer and two 500-node GRU layers, and was trained using the Adam optimizer with a batch size of 50. This approach has an advantage over the Nearest-Ending method in that it leverages all the stories in the training data for generation, rather than predicting an ending based on a single story. Thus, it can generate endings that are not directly observed in the training corpus. Like the Nearest-Ending approach, an ideal LM would be expected to generate positive examples similar to the original stories it is trained on. However, we qualitatively found that the LM-generated endings were relevant to the story context but had less of a commonsense interpretation than the provided endings, so we judged them to be less correct.

5.3.3 Experiments

We trained a classifier for each type of negative ending and additionally for each type of embedding, shown in Table 5.4. For each correct example, we generated multiple incorrect examples. We found that setting the number of negative samples per positive example near 6 produced the best results on the development set for all configurations, so we kept this number consistent across experiments. The exception is the Backward method, which can only generate one of the first four sentences in each story. For each generation method, the negative samples were kept the same across runs of the model with different embeddings, rather than

re-sampling for each configuration. After discovering that our best development results came from the Random endings, we also evaluated combinations of these endings with the other types of endings to see if they could further boost the model's performance. The samples used by these combined-method experiments were a subset of the negative samples generated for the single-method results.

5.3.4 Results

Table 5.4 shows the accuracy of all configurations, with the best test result within each group in bold. The best result using the GoogleNews word embeddings (61.5%) was slightly better than that of the ROC word embeddings (58.8%). Among the single-method results, the word embeddings were outperformed by the best result of the skip-thought (sentence) embeddings (63.2%), suggesting that the skip-thought model may capture more information about a sentence than simply averaging its word embeddings. For this reason we skipped evaluating the word embeddings for the combined-ending experiments. One caveat to this is the smaller size of the word embeddings relative to the skip-thought vectors. While it is unusual for word2vec embeddings to have more than a thousand dimensions, to be certain that the difference in performance was not due to the difference in dimensionality, we performed an ad-hoc evaluation of word embeddings that were the same size as the ROC sentence vectors (2400 nodes). We computed these vectors from the ROCStories corpus in the same way described in Section 5.3.1, and applied them to our best-performing data configuration (Rand-3 + Back-1 + Near-1 + LM-1). The result (57.9%) was still lower than that produced by the corresponding ROC sentence embeddings (66.1%), supporting our idea that the skip-thought embeddings are a richer sentence representation. Interestingly, though the BookCorpus sentence vectors obtained the best result overall (67.2%),

	Dev	Test
Rand-6		
GoogleNews WordEmb	0.625	0.585
ROC WordEmb	0.605	0.584
BookCorpus SentEmb	0.645	**0.632**
ROC SentEmb	0.639	0.631
Back-4		
GoogleNews WordEmb	0.529	0.540
ROC WordEmb	0.528	0.553
BookCorpus SentEmb	0.545	0.539
ROC SentEmb	0.548	**0.560**
Near-6		
GoogleNews WordEmb	0.641	0.615
ROC WordEmb	0.585	0.588
BookCorpus SentEmb	0.649	**0.621**
ROC SentEmb	0.632	0.615
LM-6		
GoogleNews WordEmb	0.524	0.534
ROC WordEmb	0.523	**0.544**
BookCorpus SentEmb	0.520	0.507
ROC SentEmb	0.514	0.512
Rand-4 + Back-2		
BookCorpus SentEmb	0.662	**0.669**
ROC SentEmb	0.664	0.664
Rand-4 + Near-2		
BookCorpus SentEmb	0.636	**0.641**
ROC SentEmb	0.650	0.609
Rand-4 + LM-2		
BookCorpus SentEmb	0.624	0.607
ROC SentEmb	0.640	**0.653**
Rand-3 + Back-1 + Near-1 + LM-1		
ROC WordEmb (2400)	0.599	0.579
BookCorpus SentEmb	0.656	**0.672**
ROC SentEmb	0.680	0.661

Table 5.4: Accuracy of binary classification methods on the Story Cloze Test

they performed on average the same as the ROC sentence vectors (mean accuracy of 61.1% versus 61.3%, respectively), despite that the former have more dimensions

(4800) and were trained on several more stories. This might suggest it helps to model the unique genre of stories contained in the ROCStories corpus for this task.

The best results in terms of data generation incorporate the Random endings, suggesting that for many of the items in the Story Cloze Test, the correct ending is the one that is more semantically similar to the context. Not surprisingly, the Backward endings have limited effect on their own (best result 56%), but they boost the performance of the Random endings when combined with them (best result 66.9%). We expected that the Nearest-Ending and LM endings would have an advantage over the Random endings, but our results didn't show this. The best result for the Nearest-Ending method was 62.1% compared to 63.2% produced by the Random endings. The LM endings fared particularly badly on their own (best result 54.4%). We noticed qualitatively that the LM seemed to produce very similar endings across different stories, which possibly influenced this result. The best result overall (67.2%) was produced by the model that sampled from all four types of endings, though it was only trivially higher than the best result for the combined Random and Backward endings (66.9%). Still, we see opportunity in the technique of using generative methods to expand the training set. We only generated incorrect endings in this work, but ideally this paradigm could generate correct endings as well, given that a story has multiple possible correct endings. It is possible that the small size of the ROCStories corpus limited our current success with this idea, so in the future we plan to pursue this using a much larger story dataset.

5.4 Comparison with Other Approaches

The Story Cloze Test was a shared task at the 2017 Linking Lexical, Sentential, and Discourse-level Semantics (LSDSem) workshop[7] (Mostafazadeh et al., 2017), so it has now garnered some attention in the NLP community. Some of the other submitted systems also explored the strategy of generating incorrect endings by sampling them from other stories, and then training a classifier to predict the probability that an ending is correct. In particular, as we explored in our work with the Nearest-Ending and LM approaches, Bugert et al. (2017) retrieved incorrect endings from other stories that were lexically similar to the correct endings in terms of their word embeddings, and Mihaylov and Frank (2017) sampled wrong endings that were similar to the target story context in terms of noun overlap. These systems outperform ours with 71.2%, and 72.42% accuracy, respectively. However, both these systems incorporated the cloze items in the development set for training, as well as for retrieving the incorrect endings. Our system exclusively observed the ROCStories corpus for data augmentation and training rather than any items in the Story Cloze Test itself.

Mentioned previously, the existing best approach overall (75.2%) by Schwartz et al. (2017b) used a combination of n-gram style features and a word-based RNN LM trained to predict the probability that an ending is correct. Cai et al. (2017) presented a system with competitive accuracy (74.7%) that used an attention-augmented hierarchical RNN with bidirectional LSTM layers to predict correctness. Like the systems highlighted above, these systems also relied on the development set of the Story Cloze Test for training. Unlike the ROCStories, the cloze

[7] coli.uni-saarland.de/ mroth/LSDSem/

items provide human-authored incorrect endings. This seems to be highly beneficial, despite that there are only 1,871 instances in this set. We attempted to overcome this in our work by manufacturing incorrect endings from the correct ones. However, the strong performance of these other systems suggests that a lot of information can be gained from observing the difference between the correct and incorrect endings given specifically in the evaluation. Another curious finding of this other work is that the correct ending can be predicted in some cases without considering the story context. In particular, Cai et al. found their same model achieved 72.5% accuracy when it only observed the endings. We did not evaluate it here, but it is possible that our model also paid more attention to the independent features of the endings in predicting their correctness, since both the context and ending were encoded into the same RNN layer.

5.5 Conclusion

Schwartz et al. (2017b) more closely examined the phenomenon that correct endings in the Story Cloze Test could be predicted without observing their context. They found that simple 'style features' (e.g. character and POS n-grams) of the correct and incorrect endings differed systematically, such that a binary classifier trained only on these features obtains 64.5% accuracy. They conclude that this is a function of the writing task, in that authors approached writing correct and incorrect endings in cognitively different ways. This is an interesting psychological finding, but it is less ideal if the Story Cloze Test is specifically intended to evaluate models of temporal and causal coherence between story events. To serve this goal, the candidates would ideally be authored in a way that makes it harder to distinguish their correctness based on these superficial features. Interestingly,

Schwartz et al. applied their same model based on these features to COPA, which resulted in only 53.2% accuracy, suggesting that COPA items are written in a way that mostly avoids this confounding effect.

Even though our result on the Story Cloze Test is not the current state-of-the-art, it still offers some findings that are informative for future work on this task. First, we discovered that sentence (skip-thought) embeddings appear to capture more information about a sentence than just the combination of its word embeddings. These representations may be useful for narrative modeling tasks in general. Moreover, these representations may not necessarily require large datasets to be meaningful, given that the vectors trained on the ROCStories contributed almost as greatly to the same model trained on the vastly larger BookCorpus. Second, there is an opportunity to adapt story datasets to be particularly suitable for classifiers that discriminate good and bad endings for stories. Surprisingly, even the naive method of using lexically unrelated sentences as negative endings provides some signal for this task. Even though we did not observe any benefit from more constrained methods for generating incorrect endings here, the success of approaches that rely on a very small set of human-authored incorrect endings indicates the potential of this strategy. Scaling this approach will require methods for generating incorrect endings that mimic the features of the human-authored ones. Though there is a concern that the Story Cloze Test may contain biases that allow it to be "gameable" without story knowledge, the current best system still performs well below the human gold standard (75.2% versus 100%), so there is a lot to this task that has not yet been modeled. Significant progress is likely to require much deeper knowledge of a story than what is being captured through the current approaches. Though this deeper knowledge could potentially come from vector representations like skip-thought vectors, for instance, they will likely need

to demonstrate that they can transcend surface-level lexical features to simulate something closer to commonsense reasoning.

Chapter 6

Data-driven Interactive Narrative Engine

Computers are useless. They can only give you answers.

Pablo Picasso

The previous two chapters introduced the task of closed-choice narrative continuation and illustrated it within two frameworks, COPA and the Story Cloze Test. In this chapter, I examine this same task in the context of an interactive user application. Chapter 2 outlined the domain of interactive narrative, where systems generate stories with direction from user input. Most existing interactive narrative systems do not receive user input in the form of unconstrained natural language. More commonly, users will define actions using a formal syntax akin to a programming language, or they select an input from a list of pre-authored sentences. The Data-driven Interactive Narrative Engine[1] (DINE) (Cychosz et al., 2017) avoids imposing this constraint, instead enabling users to provide natural-language input to elicit the next segment of a story. Similarly, DINE seeks to empower authors by not requiring them to be aware of the underlying system for predicting continuations. This enables them to focus on the writing task itself. This is not the case for many traditional interactive fiction platforms, where authors must also serve as

[1] dine.ict.usc.edu

programmers in parsing user input. In line with COPA and the Story Cloze Test, the obvious goal of DINE is to predict a continuation that fits coherently with the input. In this chapter, I discuss some approaches to DINE prediction, some of which were used in the previous chapters. Overall our experiments suggest that DINE items have a specific authoring design that distinguishes them from COPA and the Story Cloze Test. Consequently, the approaches used for these other tasks are largely outperformed by simpler machine learning models on DINE. DINE also has some unique requirements as a user application; in particular, prediction models must be scalable to new scenarios for which user training data is not readily available. For this reason an unsupervised approach that can be integrated into any newly authored DINE story is ideal. In this work, we compare unsupervised and supervised approaches and find that the latter obtain better prediction performance, which is a gap to be addressed by future work.

6.1 Application Design

In other parts of this thesis, the terms *user* and *author* are often conflated to mean the same person. However, they are different in DINE: the author creates the story scenario, and the user interacts with it. A DINE scenario consists of a sequence of *pages*. Each page consists of a *setup* and a list of potential *outcomes*. The text in the setup presents the user with a scenario and elicits an initial input from the user. The input triggers the system to present an outcome which continues the story and prompts the user to specify further inputs leading to new outcomes. For each outcome they define, authors can provide a list of example inputs expected to deliver that outcome, where each input typically consists of a single sentence. An author can also link an outcome to a new page so that when the user sees

that outcome, they are sent to another page in the scenario with a new setup and outcome list. Alternatively, authors can specify that a particular outcome should end the scenario. Figure 6.1 shows an example of a DINE page for a story called *Pull Over / Sleep Under*, where the protagonist (the user) is a truck driver trying to stay awake during an overnight drive. The setup appears at the beginning, and examples of possible user inputs are shown with the outcomes that are expected to be triggered by those inputs. The final outcome is designated to end the scenario. See Cychosz et al. (2017) for further details about DINE authoring.

6.2 Dataset

To compare the prediction accuracy of different models in DINE, we crowdsourced participants to play DINE scenarios and annotated their inputs with gold-standard outcomes. One challenge to this data collection process was that in order for participants to interact with the application, some method needed to already be in place for producing outcomes. A common solution to this in other work is the Wizard-of-Oz approach, where system responses are produced by a human rather than an algorithm, typically unbeknownst to users. This method affords reliable real-time data annotation, but is expensive in requiring human intervention for every system interaction. Therefore we pursued an alternative strategy where an automated model was used to deliver outcomes, and inputs were later annotated with their gold-standard outcomes, without regard to the outcome selected by the model during the user interaction. The model used for this crowdsourcing process was the Blog Stories PMI model that was discussed in detail in Chapter 4. For each potential outcome, the PMI score of the user input and the first six[2] words of

[2]This parameter was selected based on informal testing before data collection.

Pull Over / Sleep Under

(setup) I had to get my rig to Albuquerque before the shop opened at 8AM the next morning. Mendleton was already riding me hard for late deliveries. One more bad turnover and I was out of a job. I'd lose my rig, my paycheck and any hope in hell of making rent next month. But I was so exhausted. I'd been driving for 15 hours straight and had 8 more in front of me. My eyes were heavy, my head drooping, as I contemplated my options. If I didn't pull over, I could fall asleep at the wheel but if I did: nothing, but the tip of Mendleton's finger as he showed me the door, would be waiting for me when I woke. I looked around my truck to find things to occupy me. There was an almost empty pack of cigarettes, an energy drink, half of a sad sandwich from earlier in the day, a sole stick of chewing gum and that was it. That was all I had to prevent my 18-wheeler from driving off the road. Or of course I could pull over and- I didn't know what to do.

(*input*) >`I rolled down the window.`

(*outcome*) As I rolled down the windows to usher in the night air, I felt disheartened. It was muggy out. Lukewarm and goopy like soup left untended and allowed to congeal. It settled around me, heavy, making me even more drowsy than before. The road swam, slow and sluggish, before my eyes.

(*input*) >`I drank my energy drink.`

(*outcome*) As I gulped the energy drink down in long, greedy pulls, I felt hopeless. I had tried this before. Coffee. Five-hour Energy. Even caffeine pills once. It would work but only for an hour or two. I had eight sitting between me and the drop off point. The caffeine boost wouldn't hold that long. No way. No how. After a certain point, nothing would replicate the effect of a good night's sleep on my state of mind and in an hour, maybe two, I'd be faced with the same problem. Only it would be compounded this time by the comedown from caffeine burning itself off and tiring me out.

(*input*) >`I finished my sandwich.`

(*outcome*) Eating the sandwich only made me more drowsy. I felt it sitting heavy at the bottom of my belly. My mouth tasted like soggy bread and stale cheese. And my eyes, flickering shut and open, a signal, warning me that I had to choose and choose soon before my body gave over and the choice was made for me.

(*input*) >`I pulled off the highway at the next exit.`

(*outcome*) As I pulled over to sleep for the night, I felt a sense of trepidation. Sure, sleeping would ensure that I didn't harm anyone but it also meant that there was no way I'd get the contents of my truck to Albuquerque on time. Mendleton wasn't going to be happy about that. No matter what the rationale behind my decision was, Mendleton wasn't going to be happy... THE END

Figure 6.1: A DINE story with example user inputs

each outcome was calculated, and the outcome with the highest score was shown to the user.

To create the DINE evaluation set, two authors wrote a total of 25 DINE scenarios, which altogether consisted of 30 pages with an average of 9.1 outcomes per page. The content of these stories varied, spanning across science fiction, romance, psychological horror, and other genres. There were a few different design patterns that informally emerged through these authors' efforts. In particular, the authors explored three types of page setups. In a *mystery* scenario, the user is presented with a problem that they need to solve, e.g. "How do I get out of this locked room?". A *decision* scenario presents the user with a choice, e.g. "Should I intervene when I see a parent harming a child?". The scenario in Figure 6.1 is an example of a decision scenario. Finally, a *task* setup gives explicit cues to the user about what they should do, e.g. "I just woke up and I am expected to do my morning stretches". Ultimately, there were 14 mystery scenarios, 5 decision scenarios, and 6 task scenarios in this dataset. In addition to examining DINE prediction across all pages, our experiments below also evaluate how prediction performance is affected by this setup design (Section 6.4.1).

As mentioned above, when writing an outcome, DINE authors have the option of supplying example user inputs intended to yield that outcome. Authors provided four example input sentences for each outcome in this dataset (a total of 1,068 inputs across all pages), which serve as 'fake' annotated inputs. 393 participants were recruited to play these scenarios, and these participants provided a total of 2,368 inputs across all scenarios. Afterwards, each input was labeled with the most appropriate outcome as judged by the author of the corresponding scenario, disregarding the original outcome that was predicted during the interaction. Authors annotated 209 inputs (8.83%) as 'nonsense' (e.g. not interpretable

English), and 583 inputs (24.6%) as lacking an appropriate outcome from the existing candidates that had been authored. To determine the inter-rater agreement of these annotations, each of the two authors also annotated 910 inputs from the other author's scenarios. Moderate agreement based on Cohen's Kappa (Cohen, 1960) was found for inputs labeled with a specific available outcome (κ=0.637), as well as when distinguishing between inputs labeled as nonsense, lacking an available outcome, or having some available outcome (κ=0.651). Ultimately the 792 user inputs without an annotated outcome were eliminated from the evaluation set, so the dataset used in the experiments below consisted of 1,068 'fake' annotated inputs and 1,576 annotated user inputs. See Bellassai et al. (2017) for further details about the development of this evaluation set.

6.3 Prediction Models

Relative to COPA and the Story Cloze Test where each item involves a different set of candidates to select from, the DINE task has the advantage that there is one set of candidate outcomes fixed across different inputs for a given page. Thus, it can naturally be treated as a supervised classification task. However, DINE is set up to enable rapid, straightforward authoring of new scenarios. Because it is less practical to annotate data for each new page, a prediction model that scales readily to new pages with unseen outcomes is ideal. We compared the performance of some supervised models specifically trained to predict the outcomes for a single page to unsupervised approaches that do not directly observe the outcomes of a page during training. Unlike the Story Cloze Test, the story context preceding a particular user input is not observed during DINE prediction, because scenarios are authored with the intention that this context should not influence the predicted

outcome. In this way, DINE prediction is analogous to COPA where the input is a single sentence or clause.

6.3.1 Supervised Approaches

Our supervised models are page-specific, such that each model is trained to specifically predict outcomes for a single page. We carried out three different schemes for training, which are labeled as *Author*, *User* and *Both* in Table 6.1. In the first (Author), the fake author-supplied inputs were used to train the model and the real annotated user inputs were held out for prediction. In the second (User), only the real user inputs were used for training and evaluation, and the results were obtained using a 4-fold evaluation method. Finally, both the fake and real inputs were used for training (Both), and the model was evaluated on the real user inputs, again with 4-fold evaluation. The supervised models are briefly outlined below[3]. For all models except the encoder-decoder, outcomes are represented simply as abstract classes corresponding to numerical labels, rather than in their sequence form. User inputs were lemmatized and filtered to contain content words only (adjectives, adverbs, nouns, and verbs). All words occurring at least once in either the training inputs or outcomes were explicitly represented in the lexicon of the models.

Logistic Regression Classifier (LR): The multi-class Logistic Regression model shown in Figure 6.2 was trained predict outcome classes from inputs represented as bag-of-words vectors containing word frequency counts.

Logistic Regression Classifier with Tf-idf Weighting (Tfidf LR): Using the same LR model, we applied tf-idf weighting (Manning et al., 2008) to the

[3]All supervised models were implemented with Keras (keras.io), and trained for 50 epochs using the Adam optimizer

word counts in the input vectors, which discounted the value of words occurring frequently across all inputs.

Logistic Regression Classifier with Word Embeddings (WordEmb LR): As an alternative to the simple bag-of-words representation, we encoded the inputs in the LR model as the sum of their individual GloVe word vectors, as was evaluated for COPA.

Logistic Regression Classifier with Sentence Embeddings (SentEmb LR): Because they were shown to be highly useful for both COPA and the Story Cloze Test, we applied skip-thought sentence vectors to encode user inputs in the LR model.

Multilayer Perceptron Classifier (MLP): Our MLP classifier (Figure 6.2) was the same as the LR with the addition of a 200-node hidden (sigmoid) layer between the input and output layers, intended to more deeply encode the features of the user inputs. The inputs were represented as bag-of-words vectors as in the simple LR model.

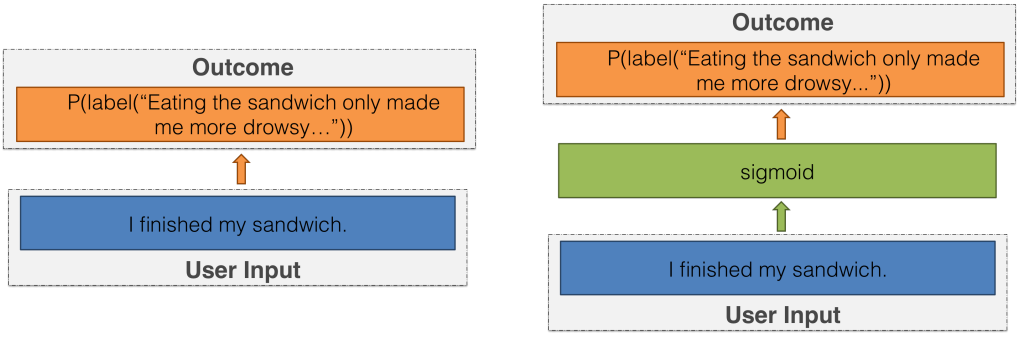

Figure 6.2: Logistic Regression (left) and Multilayer Perceptron (right) classifiers

Recurrent Neural Network Classifier (RNN): The RNN classifier (Figure 6.3) observed inputs as sequences and mapped each word in a sequence to a 100-node word embedding learned during training (not the embeddings used in the WordEmb LR). The embedded sequences were processed by a 200-node GRU

recurrent layer, and the final state of each sequence was used to predict the outcome class.

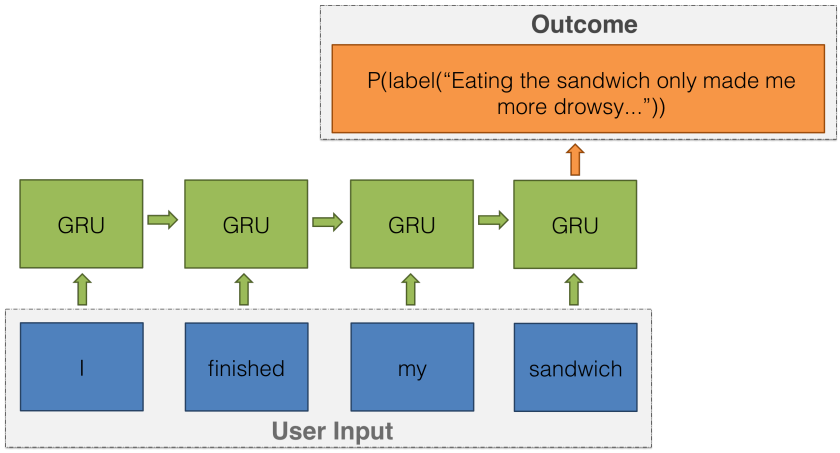

Figure 6.3: RNN classifier

Feed-forward Encoder-decoder (Enc-dec): We used the same feed-forward encoder-decoder model presented in Chapter 4 for the COPA task, relayed here in Figure 6.4. Here the outcomes were predicted based on their first sentence only. The model may look the same as the MLP classifier in Figure 6.2, but the key difference is that the encoder-decoder model directly observes the text of the outcomes. In contrast, MLP classifier only sees outcomes as discrete classes (e.g. label("Eating the sandwich...") corresponds a numerical index) and does not actually know which words they contain. In the results below, we report the performance for the model that represents user inputs as the sum of their GloVe word embeddings, which here outperformed the bag-of-words and skip-thought vectors also evaluated in the COPA experiments. We used a 200-node hidden layer as the encoder.

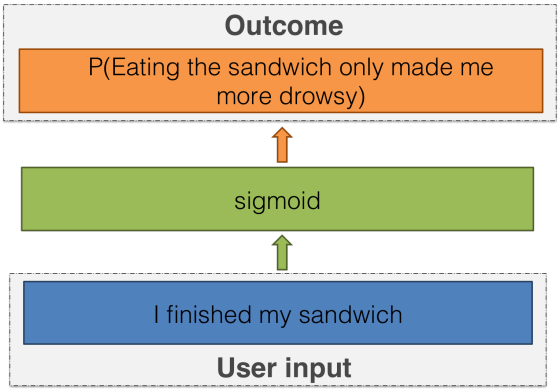

Figure 6.4: Feed-forward encoder-decoder

6.3.2 Unsupervised Approaches

We then applied some unsupervised approaches motivated in Chapters 4 and 5. For these models, each outcome was represented by its first sentence only.

Pointwise Mutual Information (PMI): We selected the outcome with the highest mean PMI score for the given user input. This is the model that was used during data collection.

Word Embedding Similarity (WordEmbSim): We predicted the outcome with the highest cosine similarity to the input in terms of averaged GloVe word embeddings.

Average Maximum Word Embedding Similarity (AvgMaxSim): Given a particular outcome, for each word in an input, we computed its cosine similarity with each word in the outcome and selected the maximum similarity score of these pairs, as in Chapter 5. We averaged these scores across the input words, and predicted the outcome with the highest average.

Sentence Embedding Similarity (SentEmbSim): We predicted the outcome with the highest cosine similarity to the input in terms of their skip-thought sentence embeddings.

	Author		User		Both	
	Acc	F1	Acc	F1	Acc	F1
LR	0.353	0.339	0.409	0.371	0.496	0.468
Tfidf LR	0.431	0.426	0.432	0.412	0.528	0.514
WordEmb LR	**0.433**	0.421	**0.474**	0.449	**0.532**	0.528
SentEmb LR	0.383	0.385	0.415	0.376	0.471	0.452
MLP	0.431	0.419	0.414	0.398	0.494	0.490
RNN	0.364	0.361	0.403	0.385	0.454	0.448
Enc-dec	0.367	0.350	0.321	0.281	0.441	0.426

Table 6.1: Accuracy and F1 scores of supervised models on the DINE evaluation set according to training set (author inputs only, annotated user inputs only, or both)

	Acc	F1
Majority	0.268	0.118
PMI	0.263	0.257
WordEmbSim	0.314	0.302
AvgMaxSim	**0.318**	0.301
SentEmbSim	0.243	0.229
ROC Enc-dec	0.148	0.081

Table 6.2: Accuracy and F1 scores of unsupervised models on the DINE evaluation set

ROCStories Encoder-decoder (ROC Enc-dec): In addition to evaluating the encoder-decoder used in COPA as a supervised approach trained directly on each DINE page (referred to in Table 6.1 as Enc-dec), we evaluated the best-performing COPA model trained on the ROCStories dataset. We applied this already-trained model as an unsupervised approach to determine whether its usefulness on COPA would generalize to DINE.

6.4 Results

Table 6.1 shows the accuracy and F1 scores for the supervised models averaged across all pages in the dataset, with the best accuracy for each training dataset (Author, User, and Both) in bold. Table 6.2 shows the corresponding results for the unsupervised models, where the test set included all annotated user inputs. This table also includes the baseline of just selecting the most frequent (majority) outcome associated with the test items for the page.

The main finding is that the supervised models make more accurate predictions than the unsupervised models, likely because the former are trained to predict page-specific outcomes. All supervised models exceed the baseline of just selecting the most frequent (majority) outcome in the annotations for each page (26.8% accuracy and 11.8% F1 score). The models trained on the combination of the author-provided inputs and the annotated user inputs (Both) performed the best, with the maximum result obtained by the LR with GloVe embeddings (53.2% accuracy and 52.8% F1 score). We can conclude that both types of data are useful for increasing prediction accuracy. When only one type of data is provided, the models benefited slightly more from the actual user inputs (best result 47.4% accuracy and 44.9% F1) than the author-provided examples (best result 43.3% accuracy and 42.1% F1). The neural classifiers (MLP and RNN) had no advantage over the simple LR, counter to the current thinking that deeper models are always better. This is likely due to how few training examples there were. LR performance was boosted by both tf-idf weighting as well as using pre-trained embeddings as word features. The sentence (skip-thought) embeddings were less effective in this setting. The Enc-dec model fared poorly relative to the others, suggesting that modeling outcomes as abstract classes is a better strategy for a supervised approach with limited training data than relying on the words themselves. This is not

possible in the COPA and Story Cloze Test since the candidate continuations are unique to each item.

Among the unsupervised approaches, AvgMaxSim performed the best (31.8% accuracy and 30.1% F1), though it was comparable to just using the average word embedding similarity (31.4% accuracy and 30.2% F1). In terms of accuracy, these models only moderately outperformed the majority baseline (26.8% accuracy), and PMI accuracy was actually the same as this baseline (26.3%). Interestingly, the encoder-decoder trained on the ROCStories (ROC Enc-dec) did not generalize to DINE prediction, performing well below the baseline (14.8% accuracy and 8.1% F1). The performance discrepancy between the AvgMaxSim model and the best supervised model trained on the author example inputs (43.3%) suggests that even having these few example inputs is helpful, even if annotated user data is not available. The problem with this is that it can easily become burdensome for authors to spend time anticipating user inputs, so it is not practical to try to scale performance by increasing the number of examples they provide. An unsupervised approach that can also leverage a small set of author examples when available may be ideal. Accordingly, Bellassai et al. (2017) addressed this in a very simple way by integrating the example inputs into the AvgMaxSim approach to model the similarity between the user inputs and these examples, rather than just the similarity between the user inputs and outcomes. For a given user input, the score for an outcome is whichever sequence has the highest similarity to the input, either the outcome text itself or one of its corresponding example inputs: $\max_{ex \in examples}(sim(user\_input, ex), sim(user\_input, outcome))$. Since example inputs are specifically intended to mimic real user inputs, it is not surprising that this boosts the accuracy of the unsupervised approach. It obtains 38.4%

	Mystery		Decision		Task	
	Acc	F1	Acc	F1	Acc	F1
WordEmb LR	0.404	0.391	0.383	0.377	**0.544**	0.540
AvgMaxSim	0.301	0.286	0.312	0.277	0.444	0.422

Table 6.3: Accuracy of best supervised model (WordEmb LR) and unsupervised model (AvgMaxSim) for each setup design category

accuracy, which is the best unsupervised result for DINE so far. This approach is currently implemented in the live DINE application.

6.4.1 Analysis of Setup Design

As highlighted in Section 6.1, each DINE scenario in these experiments had a setup design classified as either *mystery*, *decision*, or *task*. We were curious whether this variable had an impact on prediction accuracy. Table 6.3 shows the mean accuracies for the best-performing supervised and unsupervised model (WordEmb LR and AvgMaxSim, respectively) grouped by setup design. The result reported for WordEmb LR is for the Author configuration where only the example inputs are used for training.

Both models obtain much better performance on the task scenarios than the on mystery and decision ones. The discussion in Cychosz et al. (2017) points to a potential explanation for this. The task scenarios impose more constraints on the users such that only one particular action following a given outcome will produce a coherent interaction. The example in Section 6.1 illustrates this: "I just woke up and I am expected to do my morning stretches" specifically prompts the user to write something that contains the word lemma "stretch". This in turn produces an outcome that frequently restates the user's input and then contains another explicit clue for what the user should write next (e.g. "After I finish my stretches, I need

to do the laundry"). As a result, there is likely more consistency in the user inputs associated with an outcome, e.g. inputs that produce the outcome "After I finish my stretches, I need to do the laundry" are very likely to contain "stretch", making it a reliable cue for the supervised model to predict this outcome. Accordingly with this example, these cue words are often echoed in their corresponding outcomes, which is the likely reason the AvgMaxSim approach also performs better for the task scenarios. Cychosz et al. explain that the other design types require authors to anticipate a wider range of user inputs at a given point in the interaction. For example, in the scenario alluded to in Section 6.1 that sets up the user to break out of a locked room without any specific cues for how to do so, the authors must guess what users will find reasonable to write (e.g. look around for a key, scream, pound on the walls) by authoring outcomes associated with these expected inputs. Here, there is a greater chance a user will specify something not anticipated by the author, which may result in an incoherent prediction.

6.5 Conclusion

In theory, DINE is the same as the COPA task implemented inside of an interactive user application. However, this interactivity seems to influence the DINE authoring process to be different from that of the offline evaluation frameworks in the previous chapters. DINE authors write outcomes with an idea of which user inputs should trigger them, but they ultimately have no control over the predictions made by the model. Our authors developed scenarios by informally interacting with them and qualitatively observing the predictions made by the PMI model, then revising the scenarios based on these test interactions. It is therefore possible that their authoring decisions were influenced by the performance of this model, causing them

to write outcomes in a way that favored correct predictions. It would be interesting to determine if authors would have designed items differently if they had observed a different model during authoring. It is hard to draw conclusions about this from the current work given that the PMI accuracy was still low according to the gold-standard annotations, and it was ultimately not the best-performing model. Our experiments showed that the PMI model that obtained 65.2% accuracy on COPA was significantly outperformed by the AvgMaxSim model on DINE. We discovered that AvgMaxSim only obtains 53.0% test accuracy on COPA, which suggests that lexical similarity between inputs and outputs is less informative for COPA than it is for DINE. Accordingly, DINE authors seem to have focused on directing users towards the use of specific words contained in the intended outcomes. The encoder-decoder model trained on the ROCStories corpus that had moderate success on COPA was not effective at all on DINE, as further evidence that DINE and COPA items have different properties.

To summarize, as discussed in Section 6.4, prediction approaches that rely exclusively on user training data are impractical for DINE, because of the chicken-and-egg problem that data collection requires a prediction system to already be in place. The results show that example inputs provided by authors themselves are useful for training supervised models, given that these models outperform the fully unsupervised models. Future work will focus on utilizing this limited amount of supervision in a model that can still meet the expectations of users without it.

Chapter 7

Free-text Story Continuation

Writing is like driving at night in the fog. You can only see as far as your headlights, but you can make the whole trip that way.

E. L. Doctorow

The previous chapters examined the task of narrative continuation when choices for how to continue the story are provided. In the second part of this thesis, I focus on the analogous task of generating a new continuation of a story when no pre-authored candidates are given. Though these tasks are clearly related in both requiring knowledge of 'what happens next' in a story, free-text generation has some unique challenges that will be addressed here. Our work on this task is motivated by a particular application: an interface that provides assistance for story writing by automatically generating suggestions for the next sentence in a story. This application is the focus of Chapter 8. In the current chapter, I demonstrate how an RNN can be used for free-text story generation in a continuation framework. I compare this approach to the case-based reasoning model (Swanson and Gordon, 2012) referenced in Chapter 2, which previously facilitated one of the first interactive applications for free-text story continuation.

A large focus of my work on this task is specifically on the problem of evaluation. In closed-choice prediction, the task is to select which next sentence in the story is most probable from the provided candidates, so performance can be

evaluated simply in terms of accuracy. Outside of this constrained framework, there are numerous 'correct' possibilities for how to continue a given story. As with other language generation tasks, it is common to rely on human judgments of quality; for instance, by asking people to rate stories on different dimensions (e.g. "on a scale of 1-5, how {coherent, creative, interesting, etc.} is this story?") (McIntyre and Lapata, 2009; Pérez y Pérez and Sharples, 2001; Swanson and Gordon, 2009). However, human evaluations can be time-consuming and costly to carry out, particularly since they must be repeated for each new set of generated content. Moreover, these measures do not necessarily provide insight into the specific characteristics that influenced annotators' judgments, as annotators might not even be explicitly aware of them. While evaluating generation quality in a fully automated way is likely as difficult as the generation task itself, progress on this research would greatly benefit from tools that can provide some indication of generation quality without manual analysis. This chapter describes our work in Roemmele et al. (2017a), which explores the use of automated linguistic analyses to compare sentences generated by different models to the corresponding sentences in the human-authored stories, which we treat as a gold standard for narrative quality. These metrics involve straightforward NLP techniques that have been used in other work on story generation and writing quality evaluation. We apply these analyses to compare sentences generated by the RNN and a case-based reasoning model as well as two relevant random baselines. In the subsequent chapter, we extend this work by applying these same analyses to story continuation in an interactive user application.

7.1 Task Design

As defined above, free-text story continuation involves generating the next sequence in a given story, as opposed to systems that independently generate a full story. This task is useful for evaluation because continuations generated by different models for the same story can be directly compared. In this work, we performed this story continuation task on stories from the Children's Book Test[1] (CBT; Hill et al., 2016). The CBT framework contains children's novels authored between 1850 and 1950 and freely available through Project Gutenberg. Each book is divided into passages of 21 sentences. The intended task is to use the first 20 sentences (the context) to predict a word that is missing from the 21st sentence given a set of candidate words. We did not directly attempt this task in this work, but instead used the context of the passage to generate a new 21st sentence. We performed generation on only the items in the validation and test sets, which consist of a total of 18,000 passages with 440 average words per context. Table 7.1 shows two examples of story contexts (in italics) that come from Andrew Lang's *The Grey Fairy Book* and Lucy Maud Montgomery's *The Golden Road*. We used the actual 21st sentence contained in each item as a gold standard with which to compare our models, based on the assumption that this sentence is a high-quality continuation of the story.

As can be observed from the examples, the stories in the CBT framework are very different from the ROCStories corpus we utilized for our closed-choice prediction experiments. The ROCStories depict stereotypical, ordinary, mundane everyday experiences in highly standardized language. These stories are less representative of traditional features of literary narrative that are more reflected in the

[1] fb.ai/babi/

CBT stories: imagery, metaphor, setting development, theme, dialogue, tension, and suspense, for example[2]. Since the long-term goal of this particular work is to provide creative authoring support, we were motivated to select stories associated with these creative elements of narrative.

7.2 Generation Models

In this work, we evaluated two different models that both take an initial story (context) as input and generate the next sentence: a case-based reasoning (CBR) model and a recurrent neural network (RNN) model. We also considered two baseline methods to further inform our interpretation of the analyses, described below. Table 7.1 shows examples of sentences produced by each model.

7.2.1 Training Data

For training the models, we used a different dataset from the CBT stories. This dataset also consists of fiction stories, but from the domain of fiction-writing websites instead of classic literature. One motivation for this difference is that in interactive generation systems, a user's specific story genre might not be known in advance, so this capacity for domain adaptation can be particularly important. To assemble this dataset, we gathered stories from websites including fictionaut.com, ficwad.com, wattpad.com, writerscafe.org, and various other sites containing fiction uploaded by authors themselves. These stories cover a wide range of genres related to fantasy, horror, romance, and science fiction. Many of them are fan fiction stories that depict characters and settings from existing works (e.g. *Harry Potter*, *Naruto*, and *Twilight*). This dataset consists of 607,627 stories, with a total

[2]en.wikipedia.org/wiki/Fiction_writing

Papa,' she said, 'it is not artificial, it is REAL!' 'Ugh!' said all the ladies-in-waiting, 'it is real!' 'Let us see first what is in the other casket before we begin to be angry,' thought the Emperor, and there came out the nightingale. It sang so beautifully that one could scarcely utter a cross word against it. 'Superbe! charmant!' said the ladies-in-waiting, for they all chattered French, each one worse than the other. 'How much the bird reminds me of the musical snuff-box of the late Empress!' said an old courtier. 'Ah, yes, it is the same tone, the same execution!' 'Yes,' said the Emperor; and then he wept like a little child. 'I hope that this, at least, is not real?' asked the Princess. 'Yes, it is a real bird,' said those who had brought it. 'Then let the bird fly away,' said the Princess; and she would not on any account allow the Prince to come. But he was nothing daunted. He painted his face brown and black, drew his cap well over his face, and knocked at the door. 'Good-day, Emperor,' he said. 'Can I get a place here as servant in the castle?'	
R-sent	The music plays, it's my favourite song, But I don't want to sing along.
CBR	A thousand flashbacks appeared in my mind vision as I demanded my brain to find a reasonable explanation for my present state.
1-gram	It and placed peered of why 7:40 and bolted you.
RNN	Yes, he's a 'one'.
Gold	'Yes,' said the Emperor, 'but there are so many who ask for a place that I don't know whether there will be one for you; but, still, I will think of you.
"Cecily, you've got a dreadful cold," said the Story Girl anxiously. "In spite of Peg's ginger tea," added Felix . "Oh, that ginger tea was AWFUL," exclaimed poor Cecily. "I thought I'd never get it down – it was so hot with ginger – and there was so much of it! But I was so frightened of offending Peg I'd have tried to drink it all if there had been a bucketful. Oh, yes, it 's very easy for you all to laugh! You didn't have to drink it." "We had to eat two meals, though," said Felicity with a shiver. "And I don't know when those dishes of hers were washed. I just shut my eyes and took gulps." "Did you notice the soapy taste in the porridge?" asked the Story Girl. "Oh, there were so many queer tastes about it I didn't notice one more than another," answered Felicity wearily. "What bothers me," remarked Peter absently, "is that skull. Do you suppose Peg really finds things out by it?" "Nonsense! How could she?" scoffed Felix, bold as a lion in daylight." She didn't SAY she did, you know," I said cautiously. "Well, we'll know in time if the things she said were going to happen do," mused Peter.	
R-sent	How in the hell am I supposed to say no to that face?
CBR	I sighed and stood
1-gram	You!
RNN	"Course" I said, then nodded.
Gold	"Do you suppose your father is really coming home?"

Table 7.1: Examples of CBT story contexts and continuations generated by each model

of 41,458,210 sentences and 467,023,696 words. For all models, we established a vocabulary of words that occurred at least 25 times in this corpus, which ultimately included 83,292 words. All other words were ignored by the models during training (in the case of the RNN and 1-gram baseline, they were all mapped to a single <UNKNOWN> token).

7.2.2 Case-based Reasoning (CBR)

Chapter 2 introduced case-based reasoning, which is a general AI problem-solving approach where a new problem is solved by consulting a known solution for an existing problem (Aamodt and Plaza, 1994). In the context of story generation, CBR is used to establish an analogy between a new story and an existing story so that the existing story can inform the generation of the new story (Turner, 1993a). Swanson and Gordon (2012) applied this paradigm to produce new sentences in a story by retrieving them from a corpus (the 'case library'). The CBR approach functions as a nearest-neighbors classifier: given the most recent sentence in a new story, the system finds the existing sentence in the corpus that is most similar to the new sentence. It then looks at the story in which the existing sentence appears and retrieves the sentence that immediately follows it. The rationale is that because of the similarity between the new sentence and the existing sentence, what appears after the existing sentence in its story is also a reasonable prediction for what happens next in the new story. To compute similarity, each sentence is encoded as a bag-of-words vector, whose values are the number of times each word in the corpus vocabulary (lexicon) occurs in that sentence. Then the similarity between two sentences is equal to their vector cosine similarity (Manning et al., 2008). This approach is also familiar from Chapter 5, where it was labeled as the Nearest-Ending method, used to augment the training data for the Story Cloze

Test models. For the current task, we used the web fiction dataset described in Section 7.2.1 as the case library. To generate the next sentence for a given story context, the model examines this dataset to find the sentence most similar to the last one in the context and returns the sentence that follows it in its corresponding story.

7.2.3 Recurrent Neural Network (RNN)

We use the RNN language model[3] with GRU units detailed in Chapter 3 (illustrated in Figure 3.3) to generate story continuations. To review, the RNN learns a conditional probability distribution of each word occurring in a story given the words that precede it. This distribution is computed through a set of nonlinear functions (the hidden layer) that maintain a representation of the story up to a word at a particular timepoint in the sequence. The input to the first hidden layer is a word sequence where each word is encoded as a vector of real values (a word embedding). The output of the uppermost hidden layer is passed to a top (softmax) prediction layer which gives the probability distribution of all possible next words in the sequence. Training occurs by minimizing cross-entropy loss such that the parameters of the model are optimized to increase the predicted probabilities of the true words that actually appear a story. After training, the learned distributions can then be used to generate new words in a given story by iteratively sampling from the probabilities of all potential next words in the story. This model was also applied in Chapter 5, where it was used to augment the training data for the Story Cloze Test models in the same way as the CBR (Nearest-Ending) model.

In this work, we used an RNN with a 300-dimension embedding layer and two 500-dimension GRU layers. We tokenized the stories into lowercased words, and

[3]Code available at: github.com/roemmele/narrative-prediction

all punctuation was treated in the same way as word tokens. During training, the model processed entire stories word by word in batches of 50 stories at a time, using the Adam algorithm for optimization. To use the trained model to generate a new sentence for a given context, we fed the context into the model and sampled a word from the resulting probability distribution for the next word. We appended this word to the story as the beginning of the next sentence, and continued adding words to the sentence until an end-of-sentence punctuation token ('.', '!', and '?') was generated. Because some of our analyses in Section 7.3 require us to present sentences as regular strings rather than lists of tokens, we 'detokenized' the sentences using some heuristics for punctuation formatting, capitalization, and merging contractions.

The RNN has some theoretical advantages over the CBR model. First, it is a productive model, meaning that it can generate sequences that do not directly appear in its training data. Whereas there is an exponential number of sequences the RNN can produce through all possible combinations of words in the vocabulary, the CBR model is limited to the sentences it has observed in the corpus. In this way, the RNN is arguably a better model of computational creativity, since natural language has this same productivity that leads authors to produce novel content. Another important advantage of the RNN model is that it considers an entire story when generating the next sentence; in contrast, the CBR model only observes the most recent sentence. Consequently, the RNN has more opportunity to refer to events or entities that appeared earlier on in the story. Quite obviously human authors frequently refer to story elements that previously appeared further back than the most recent sentence, so any ideal generation model will have this same capacity. However, it is important to keep in mind that the CBR model retrieves human-authored sentences, which may be favorable in practical ways over the

RNN-generated ones. It is interesting to compare these particular models on the same task because while they both rely on a data-driven approach, they assemble sequences from different units of generation (sentences versus words) and thus are likely to produce distinct types of sequences.

7.2.4 Baselines

We also considered two baseline models as additional comparisons in our analyses, both of which randomly generate sentences without regard to the story context. The first baseline (R-sent) simply selects a random sentence from the training corpus. The second baseline is a unigram language model (1-gram), which like the RNN generates sentences word by word. Its probability distribution is just the relative frequency of each word in the training corpus, so each word is sampled independently from the previous word during generation. By including these in the analyses, we show the expected performance on the evaluation metrics even when there is minimal signal in the model.

7.3 Automated Linguistic Analyses

We applied a set of automated linguistic analyses to examine differences in the quality of the generated sentences within their story context. Each metric is listed below with an explanation of its relevance to this evaluation task. We focused on features used in previous work on story generation (in particular, the Story Cloze Test) and evaluating writing quality. The metrics can be broadly categorized into two types: 1) metrics that analyze the generated sentence in isolation from its context (Story-Independent), and 2) those that evaluate the sentence with reference to the context (Story-Dependent). Intuitively, the first type of analysis captures

how well-written the sentence is by itself, while the second determines how apt the sentence is for that particular story. We can reasonably expect both dimensions to be important for this task.

7.3.1 Story-Independent Metrics

Sentence Length: Sentence length is an extremely simple feature that can reliably discriminate between text genres, authors, and other characteristics like overall readability (Flesch, 1948; Graesser et al., 2004; Karlgren and Cutting, 1994). Moreover, the length of a candidate ending in the Story Cloze Test was found to influence its correctness (Bugert et al., 2017; Schwartz et al., 2017a). Length is simply the number of words in each generated sentence (Metric 1).

Grammaticality: Grammaticality is an obvious feature of high-quality writing. To judge the grammaticality of generated sentences, we used Language Tool[4] (Miłkowski, 2010), a rule-based system that detects various grammatical errors. The system provided an overall grammaticality score (Metric 2) for each sentence, equal to the proportion of total words in the sentence deemed to be grammatically correct.

Lexical Diversity: High-quality writing has been found to contain a larger set of unique words and phrases, and avoids overly repetitious use of the same phrases (Burstein and Wolska, 2003; Crossley et al., 2011; Kao and Jurafsky, 2012). We analyzed the number of unique words (types) in the generated sentences relative to the number of total word occurrences (tokens), known as the type-token ratio (Metric 3). A single type-token ratio was computed for each model from the entire set of sentences generated by that model. Because our models were only aware of words that occurred 25 or more times in the training data, we only counted words

[4]Code at: pypi.python.org/pypi/language-check

in this vocabulary in the ratio (in contrast to the R-sent and CBR models, the RNN and 1-gram models never had the opportunity to generate words not in this vocabulary). We also measured the number of unique phrases in the same way, by computing the total proportion of unique trigrams to the total number of trigram occurrences in the generated sentences (Metric 4), again only considering trigrams where all tokens were contained in the training vocabulary.

Lexical Frequency: Related to lexical diversity, writing quality has been found to correlate with the use of less common words (Burstein and Wolska, 2003; Crossley et al., 2011). We measured the average log frequency of the words in each generated sentence (Metric 5), where the frequencies were Good-Turing smoothed counts taken from the 3-billion-word Reddit Comment Corpus[5]. To keep this metric consistent with the others where higher scores are hypothesized as more favorable, we report the negative (inverse) log frequency, so that higher numbers indicate lower word frequency.

Syntactic Complexity: Writing quality is also associated with greater syntactic complexity (Beers and Nagy, 2009; McNamara et al., 2010; Pitler and Nenkova, 2008; von Glasersfeld, 1970; Yang et al., 2015). We examined this feature in terms of the number and length of syntactic phrases in the generated sentences. Phrase length was approximated by the number of children under each head verb/noun as given by the dependency parse. We counted the total number of noun phrases (Metric 6) and words per noun phrase (Metric 7), and equivalently the number of verb phrases (Metric 8) and words per verb phrase (Metric 9). To account for the effect of sentence length on these measures (i.e. longer sentences may naturally contain more and longer phrases), we divided all measures for each sentence by its length.

[5]Available in spaCy: spacy.io/docs/api/token

7.3.2 Story-Dependent Metrics

Lexical Cohesion: While it may seem quite obvious that words that appear in the same context in coherent text tend to be related in meaning, NLP research at large has benefited greatly from shallow metrics that quantify this lexical cohesion (Foltz et al., 1998; Lapata and Barzilay, 2005). Accordingly, these features were found to be relevant to prediction in the Story Cloze Test, as correct endings tended to have higher lexical similarity to their contexts (Mihaylov and Frank, 2017; Mostafazadeh et al., 2016; Flor and Somasundaran, 2017). First, and most simply, we computed the overall proportion of overlapping words between the context and generated sentence according to their Jaccard similarity (Jaccard, 1912) (Metric 10). We filtered this measure to include only words tagged as adjectives, adverbs, interjections, nouns, pronouns, proper nouns, and verbs (with the exception of pronouns, these are the categories associated with content words). Second, we examined similarity in terms of word embeddings, specifically utilizing the GloVe embedding vectors also used in the closed-choice prediction experiments. We computed the cosine similarity between the means of the embeddings for the content words in the generated sentence and its context, respectively (Metric 11).

Alternatively, in contrast to computing similarity at the word level, we also looked at similarity between full sentence vectors given by the skip-thought model. Our experiments in Chapter 5 discovered that this model provided useful story representations as judged by their impact on the Story Cloze Test, by representing sentences according to their relation with adjacent sentences. We used the same 4800-dimension sentence vectors trained on the 11,000 books in the BookCorpus to encode each of the sentences in the context as well as the generated sentence. We then computed the cosine similarity between the mean of the context sentence vectors and the vector for the generated sentence (Metric 12).

Style Consistency: Automated measures of writing style have been used to predict the success of fiction novels (Ganjigunte Ashok et al., 2013; Pennebaker et al., 2015). Moreover, Schwartz et al. (2017a) found that simple n-gram style features could distinguish between correct and incorrect endings in the Story Cloze Test. Since proficient authors exhibit style consistency across a particular text (Gamon, 2004), we similarly posit that generated sentences should match the style of their contexts. We examined the similarity in style between the context and generated sentence in terms of their distributions of coarse-grained part-of-speech (POS) tags, using the same approach as Ireland and Pennebaker (2010). The similarity for each POS category was quantified as $1 - \frac{|pos_{context} - pos_{gen\_sent}|}{pos_{context} + pos_{gen\_sent}}$, where pos is the proportion of words with that tag. We computed the score for each category (adverbs, adjectives, conjunctions, determiners, nouns, pronouns, prepositions and punctuation), and then averaged these scores (Metric 13). In addition to the category distribution of individual words, we also looked at style similarity in terms of POS trigrams (Argamon et al., 1998). To do this, we computed the Jaccard similarity between the POS trigrams in each generated sentence and those in the corresponding context (Metric 14).

Entity Coreference Rate: Similar to the expectation of lexical cohesion between sentences in a text, a generated sentence should mention entities that have been previously introduced in the story. Entity coreference has been used in existing work for automatically judging coherence (Barzilay and Lapata, 2008; Elsner and Charniak, 2008). It is a particularly important factor in story coherence, since events in stories are linked by common characters, locations, and objects (Elsner, 2012). To compute an entity coreference rate, we found the proportion of entities (equivalent to noun phrases) in the generated sentence that co-referred

to an entity in the corresponding context[6] (Metric 15). Higher coreference rates indicate more entity coherence between the generated sentence and context.

Sentiment Similarity: The relation between the sentiment of a story and a candidate ending in the Story Cloze Test predicted its correctness (Flor and Somasundaran, 2017; Goel and Singh, 2017; Bugert et al., 2017). We applied sentiment analysis to the context and generated sentence using the tool[7] described in Staiano and Guerini (2014), which provides a valence score for each of 11 emotions in a given text. For each emotion, we computed the inverse distance between the scores for the context and generated sentence: $\frac{1}{(1+|score_{context} - score_{gen\_sent}|)}$. We averaged these values across all emotions to get one overall sentiment similarity score (Metric 16).

7.4 Results and Discussion

Table 7.2 shows the mean metric scores across all 18,000 generated sentences for each model compared to the original (gold) sentences contained in the CBT stories. Differences between models were statistically evaluated using two-sample Monte Carlo permutation tests, with significance shown at $p < 0.005$ due to Bonferroni adjustment (there are 10 model comparisons, so the alpha level of 0.05 is adjusted to $0.05/10 = 0.005$).

There are several notable results to highlight in this table. First, the sentences generated by the models were much shorter than the corresponding gold sentences, which may reflect the domain difference between the training corpus and the CBT stories. Overall, the gold sentences most often obtained the highest scores on

[6]Using Stanford CoreNLP: stanfordnlp.github.io/CoreNLP

[7]github.com/marcoguerini/DepecheMood/releases

	R-sent	**CBR**	**1-gram**	**RNN**	**Gold**
Story-independent metrics					
1. Sentence length	13.36	15.56*‡§	13.67	13.10	28.84*†‡§
2. Grammaticality	0.957‡	0.961*‡	0.925	0.992*†‡⋆	0.982*†‡
3. Type-token ratio	0.057†§⋆	0.042§⋆	0.057†§⋆	0.010	0.020§
4. Unique trigram ratio	0.776†§⋆	0.491§⋆	0.946*†§⋆	0.307	0.418§
5. Inverse word frequency	7.078§	7.122*‡§	7.038§	6.056	7.399*†‡§
6. # of noun phrases	0.238‡§	0.239‡§	0.192	0.227‡	0.225‡
7. Noun phrase length	0.149⋆	0.141⋆	0.144⋆	0.143⋆	0.087
8. # of verb phrases	0.190†‡⋆	0.186‡⋆	0.164	0.191†‡⋆	0.181‡
9. Verb phrase length	0.367†‡⋆	0.346‡⋆	0.261⋆	0.403*†‡⋆	0.219
Story-dependent metrics					
10. Jaccard similarity	0.004‡	0.005*‡	0.003	0.006*†‡	0.036*†‡§
11. GloVe similarity	0.227‡	0.228‡	0.192	0.227‡	0.246*†‡§
12. Skip-thought similarity	0.682	0.713*	0.718*†	0.733*†‡	0.799*†‡§
13. Word POS similarity	0.503‡	0.541*‡§	0.442	0.501‡	0.698*†‡§
14. Trigram POS similarity	0.028‡	0.034*‡§	0.016	0.031*‡	0.070*†‡§
15. Entity coreference rate	0.440‡	0.456*‡	0.328	0.536*†‡	0.644*†‡§
16. Sentiment similarity	0.976‡	0.979*‡	0.966	0.980*†‡	0.986*†‡§

Statistical significance, $p < 0.005$: *greater than R-sent; †greater than CBR; ‡greater than Unigram; §greater than RNN; ⋆greater than Gold

Table 7.2: Mean scores on metrics for generated sentences and gold sentences

the metrics, which suggests that these metrics are correlated with writing quality. Among the other story-independent features, the order of the model scores was very mixed. One unexpected result was that the RNN had a higher overall grammaticality score than the gold sentences. Since it is probably not the case that the gold sentences are ungrammatical, it is worth exploring whether there were unique features in the gold sentences that the Language Tool scorer consistently recognized as ungrammatical.

The gold sentences had a lower type-token and unique trigram ratio than all models except for the RNN. The random baselines demonstrated the highest scores on these measures, which is unsurprising for the 1-gram model since it obeys no constraints on which word combinations qualify as grammatical. The contrast

between the CBR and gold sentences may reflect writing style differences between classic literary authors and self-published authors in the present day. The fact that the RNN had even lower lexical diversity than the gold sentences is an interesting consideration that may have to do with the probability distribution learned by the RNN, where perhaps probability was largely concentrated under a narrow set of words. On the other hand, the findings for word frequency favored the human-authored sentences as expected, as these sentences did use less frequent words. Along with having a smaller vocabulary, the RNN model tended towards more common words relative to the other models.

In terms of syntactic complexity, the gold sentences appeared to contain far more noun and verb phrases than the other models, but this was not the case once sentence length was taken into account. While it was expected that the 1-gram sentences had little syntactic complexity (since the model has no knowledge of syntax), it was surprising that the gold phrases were also much shorter on average than the phrases generated by the other models. The RNN model was notable for its verb phrases, which were longer and more frequent than those in the other sentences.

It is intriguing that significant differences emerged between the R-sent and CBR models on the story-independent metrics, since these sentences come from the same corpus and therefore would be expected to have similar features. It is likely that the CBR model selected sentences with specific features not evenly distributed across the corpus at large; for example, the CBR sentences were longer, more grammatical, contained rarer words, and had less verb phrase complexity.

The results for the story-dependent analyses are more consistent across metrics. For all measures, the gold sentences scored the highest: they were more semantically related to their story contexts, were more stylistically similar in terms of

part-of-speech categories, were more likely to co-refer to context entities, and better matched the sentiment of their contexts. This, along with the low performance of the random baselines on these measures, suggests that scores on these metrics correlate positively with story coherence.

The story-dependent metrics are particularly useful for comparing the CBR and RNN models. In Section 7.2, we discussed how in contrast to the CBR model, the RNN can in theory observe the entire story context and produce sentences more targeted to that unique context. We observe some practical evidence for this in these results: the RNN sentences were more semantically related to their contexts in terms of Jaccard and skip-thought similarity, more frequently referred to context entities, and had greater sentiment similarity to their contexts. However, the CBR sentences still demonstrated greater stylistic similarity to their contexts than the RNN sentences.

7.5 Conclusion

Overall our results suggest that automated linguistic analyses can capture meaningful differences between generation models in a story-continuation framework. For the story-independent analyses, it may not necessarily be the goal of a system to maximize scores on these metrics. In contrast, the story-dependent analyses suggested that higher scores on these metrics do indicate higher quality. In general, the human-authored sentences help interpret the comparison between the models. If the goal is to make the generated sentences more like the human-authored ones, then progress can be evaluated in terms of the similarity in scores between the gold sentences and the generated ones.

We measured generation quality according to existing features that have been used in story prediction and writing quality evaluation, but these are relatively shallow analyses. Our metrics do not directly address some of the more complex linguistic dimensions specific to the domain of narrative, such as character development, plot structure, and suspense. Moreover, given our interest in applying this work to evaluate interactive story generation in a creativity support context (introduced in the next chapter), analyses that focus specifically on linguistic creativity (Zhu et al., 2009) are important to explore in future work. Automatically modeling these types of features is an extremely difficult language understanding problem, for which developing effective metrics still requires much research. However, keeping in mind our intended application of this work, we see any characteristics that make generated content more appealing to authors as relevant to creativity. The next chapter further explores this perspective by analyzing what authors find helpful for advancing a story according to these features.

Chapter 8

Creative Help

Creativity is the power to connect the seemingly unconnected.

William Plomer

At the intersection between research on natural language generation, computational creativity, and human-computer interaction is the vision of tools that directly collaborate with people in authoring creative content. Chapter 2 reviewed the recent work on creative language generation, through which this ambition has started to come to fruition. Our application Creative Help[1] (Roemmele and Gordon, 2015) explores this vision for story writing. Creative Help functions as a story writing assistant, where authors receive 'help' through automated suggestions for new sentences in an ongoing story. The application has a functionality that tracks author's modifications to suggestions, by which user judgment of quality can be elicited implicitly. This approach is related to rewriting tasks in other NLP domains where annotators edit sentences to improve their perceived quality (Sakaguchi et al., 2016), enabling the features of the modified sequence to be compared to those of the original. In Roemmele and Gordon, we produced Creative Help suggestions using the CBR nearest-neighbors similarity approach illustrated in the previous chapter. In this current work, rather than suggesting human-authored sentences taken from existing stories, we apply an RNN language

[1] fiction.ict.usc.edu/creativehelp/

model to dynamically generate unique sentences word-by-word. Our previous work also compared different configurations of the suggestion model according to the similarity between a suggestion and modification, based on the idea that more helpful suggestions will receive fewer edits. Here, we focus on quantifying suggestions according to the automated linguistic analyses in Chapter 7, and examining how these features correlate with authors' modifications. We propose that this approach is useful for identifying the aspects of generated content authors implicitly find most helpful for writing stories. It can inform the evaluation of future creativity support systems in terms of their ability to maximize features associated with helpfulness.

8.1 Related Applications

Automated support for creative story writing is an emerging application of narrative generation research. The previously discussed Say Anything application (Swanson and Gordon, 2012) was a precursor to this task, by being one of the first tools to enable a user to collaboratively write a story with an automated system. The applications demonstrated by Manjavacas et al. (2017), Khalifa et al. (2017), and Clark et al. (2018) are similar to Creative Help in using an RNN language model for generation in a creativity support framework. Manjavacas et al. focus on a 'temperature' variable used to vary the level of randomness in the RNN probability distribution. Their interface enables users to control the temperature setting to explore its influence on a generated sentence. They plan to evaluate this system as future work. Khalifa et al. presented a similar system, DeepTingle, which autocompletes the next word in an author's story using an RNN specifically trained on books written by one particular author. Participants rated excerpts of the

original author's text compared to RNN-generated ones in terms of grammaticality, coherence, and interestingness. Clark et al. conducted a thorough qualitative analysis of authors' interactions with this type of application, and presented some recommendations for interface features such as allowing users to control the level of surprise in the suggestions and offering different generation formats (e.g. sentences versus keywords). Our work is unique from these in that we quantitatively evaluate authors' interactions with the application by analyzing their edits to the generated sentences, in particular in terms of their linguistic features.

8.2 Interface

Creative Help has a simple interface: authors see a text box where they can start typing a story. They are instructed that they can type *help*\ at any point while writing in order to generate a suggestion for a new sentence in the story. The suggestion appears in place of the *help*\ string. Figure 8.1 shows an example with the suggestion returned by the help request underlined. The author can freely modify this sentence like any other text that already appears in the story. There is no minimum or maximum requirement on receiving suggestions.

As soon as the suggested sentence appears to the author, the application starts tracking any edits the author makes to it. Once one minute has elapsed since the author last edited the sentence, the application logs the modified sentence alongside the original version of the sentence in a database. See Roemmele and Gordon (2015) for further details about this tracking and logging functionality. The result of authors' interactions with the application is a dataset aligning each generated suggestion to its corresponding modification along with the story context that precedes the help request.

Figure 8.1: Creative Help interface with a generated suggestion

8.3 Generation

8.3.1 Motivation

We use the same approach to generating sentences with the RNN LM that was described in Chapter 7, i.e. we randomly sampled a word from the LM probability distribution and appended it to the story, and iteratively continued doing this until a sentence boundary was detected. In that chapter we discussed the advantages the RNN model has over the previously integrated CBR approach for producing suggestions, in particular that it can generate sentences that have not been directly observed in other stories. The approach of randomly sampling from the probability distributions yields some unpredictability in what will be generated. Some researchers have theorized that randomness plays a large role in human creativity, on the basis that creativity involves making sense out of unpredictable combinations of ideas (Sweller, 2009). For example, in the book *The Creative Mind: Myths and Mechanisms*, Boden (2004) explains:

> "Randomness is widely seen as incompatible with creativity. If Mozart had written his dice-music by randomly choosing every note (instead

> of carefully constructing sets of alternative bars), the composition of minuets would have been as improbable as the writing of Hamlet by the legendary band of monkeys-with-typewriters in the basement of the British Museum . . . However, randomness did play a part when the dice-music was actually played. Moreover, random genetic mutations are seen as essential for the creation of new species. And random muscular tics are used as the seeds of exciting musical improvisations, by a jazz-drummer suffering from a neurological disease."

Accordingly, the capacity of computational systems to simulate randomness may be particularly relevant for automated creative assistants (Dartnall, 2013; Liapis et al., 2016). Boden goes on to address this:

> "What is useful for creativity in minds and evolution is useful for creative computers too. A convincing computer model of creativity would need some capacity for making random associations and/or transformations . . . Indeed, some creative programs (such as Cohen's and Johnson-Laird's) rely on random numbers at certain points, and genetic algorithms can produce order out of chaos."

To be clear, systems that leverage randomness are not necessarily models of human creativity, but they may be valuable in supporting human creativity. In the context of assistance for creative writing, injecting randomness may serendipitously result in a sentence that presents an interesting word sequence to the author. Veale (2012) explains that seemingly nonsensical language challenges people to think deeper about its meaning, as demonstrated by people's ability to interpret figurative utterances. Noam Chomsky's famous sentence "Colorless green ideas sleep furiously" is an example of how nonsensical language can still be processed

syntactically. In fact, as silly as it seems, there have been attempts to actually interpret this sentence's meaning, e.g. Chao (1997). Of course, Boden makes clear that meaning is unlikely to be derived from purely random associations, which most often result in unintelligibility. "Furiously sleep ideas green colorless", for instance, is far more difficult to parse than its counterpart, making it harder to search for an interpretation. Thus, a model for creative writing support should ideally recognize the constraints of natural language enough to produce interpretable utterances. By training the RNN on a corpus of stories, it observes some of these rules, but models them probabilistically rather than as fixed constraints. This leaves some opportunity to produce sequences the author does not anticipate, which may be appealing for that reason.

The RNN model has an additional advantage over the previously implemented CBR approach in terms of space and time efficiency. In this work, we do not directly compare these two models, because the CBR model in our experiments in Roemmele and Gordon (2015) caused users to wait a long time (10-20 seconds) for a suggestion to be returned after a help request. This latency is due to how the CBR model works: it searches through the entire corpus for the sentence that is most similar to the user's most recent sentence, so a larger corpus results in slower retrieval. The RNN avoids this problem because the size of its parameters (e.g. the number of nodes in the layers) is the same regardless of the number of stories in its training corpus. It only stores the lexicon associated with the corpus, which is dramatically smaller than the total set of sentences. Consequently, suggestions are returned to the user in 1-2 seconds on average. Efficiency is a significant aspect of the usability of interactive applications, so this new version of Creative Help is preferable from this perspective.

8.3.2 Model Setup

The RNN LM was trained on 8,032 books in the BookCorpus that was introduced in the previous chapters, which contains self-published fiction novels across a variety of genres. We divided the books into their respective 155,400 chapters. In total this dataset consisted of a little over half a billion words. We established a vocabulary of all words occurring at least 25 times in the text, which resulted in 64,986 unique words being included in the model.

In this work, we handled proper names uniquely by replacing each of them with a token indicating their entity type and a unique numerical identifier for that entity. For example, the sentence "Tom and Lisa met one night at a swanky New York party next to Tom's apartment" was represented as "<PERSON1> and <PERSON2> met one night at a swanky <LOCATION1> party next to <PERSON1>'s apartment". During generation, we maintained a list of all entities mentioned prior to the help request. When the model generated one of these entity tokens, we replaced it with an entity of the corresponding type and numerical index in the story (e.g. if <PERSON1> was generated, it was replaced with "Tom"). If no such entity type was found in the story, we randomly sampled an entity token from a list of entities found in the training data.

Just as in Chapter 7, we set up the RNN LM with a 300-dimension word embedding layer and two 500-dimension GRU layers. During training, the model observed chapters in batches of 125 at a time, and the Adam algorithm was used for optimization. We trained the model for one single iteration through all chapters.

When a Creative Help request was made, the model read all text prior to the help request, and we sampled from the resulting probability distribution to generate a new word. We repeated this process to generate 35 tokens, and we then

filtered all tokens after the first detected sentence boundary[2]. In some cases, no sentence boundary was detected so all 35 words were included in the returned sentence. Finally, we 'detokenized' the sentence using some heuristics for punctuation formatting, capitalization, and merging contractions.

8.4 Experiment

We set up a task where we recruited people via social media, email, and Amazon Mechanical Turk (AMT) to interact with Creative Help. When participants navigated to the site via the provided link, they saw the following instructions:

Creative Help is an experimental application intended to help people write stories by automatically suggesting new sentences in the story. We are looking for participants to spend 15 minutes using the app to write a story about anything you choose. At any point while writing, simple type \help\ to generate a new sentence. You are welcome to edit, add to, or delete this suggestion just like any other text in your story. The point of the task is to experiment with asking for \help\ but you are not required to make a certain number of requests. It's ultimately your story, so you choose what to do with the suggestions. The app will track your story, but your identity is completely anonymous to the researchers.

Participants were also told that after fifteen minutes, the application would notify them that they could complete the task by taking them to a short questionnaire. Upon agreeing to these instructions, users were presented with a text box where they could start writing and making help requests. When the user typed \help\, a suggestion appeared in place of the help string. After fifteen

[2]Based on spaCy's sentence segmentation tool: spacy.io

minutes, the user was provided with the questionnaire link, but they could continue writing with no maximum time limit and proceed to the questionnaire when ready. AMT workers were compensated $3.00 for their participation after completing the questionnaire. Ultimately, 139 users participated in the task, resulting in suggestion-modification pairs for 940 help requests.

8.5 Analysis and Results

8.5.1 Descriptive Analysis of Suggestions

Table 8.1 shows some examples of the generated suggestions along with the authors' resulting modifications. To provide a brief overview of the content contained in the suggestions, Table 8.2 shows the most common lemmatized words generated by the model according to grammatical (part-of-speech) categories as well the most common bigrams and trigrams, with the total count of each item in parentheses. The total number of words generated across all suggestions was 16,025.

The statistics provide a shallow characterization of the model's writing style and preference for topics. It commonly conveys both narrated and quoted dialogue, as indicated by the frequency of the verbs *say, tell, ask, whisper* and *talk*. It is surprisingly eager to introduce a *monster* into a story. It refers frequently to body parts (*hand, eye, body, face, head, blood*), elements of a residence (*home, door, room, bed*), and *time* (along with specifically *day* and *night*). It likes abstract nouns (*way, idea, mind, thing, life, truth*). It often writes from the first-person perspective, given that *I* is by far the most frequent pronoun (though some of these occurrences might be included in quoted dialogue). In terms of pronouns, it is more likely to refer to female than male characters, indicated by the relative frequency of *she* and *her* versus *he* and *him*. However, *man* and *father* also show

Initial Story (Context)	Suggestion	Modification
I knew it wasn't a good idea to put the alligator in the bathtub. The problem was that there was nowhere else waterproof in the house, and Dale was going to be home in twenty minutes.	I needed to know, too, and I was glad I was feeling it.	I needed to know how upset he would be if he found out about my adoption spree.
It was a dark and stormy night. The old man from another planet had a story to tell, and this is the story he told. And there was a note of the note. I picked up the note with trepidation, and read in a quavering voice: "Leave this house. You are in danger.	I was shaking, so I was worried about the tears that I would need to save.	I was shaking, so I was worried about the tears that I would need to save.
My brother was a quiet boy. He liked to spend time by himself in his room and away from others. It wasn't such a bad thing, as it allowed him to focus on his more creative side. He would write books, draw comics, and write lyrics for songs that he would learn to play as he got older.	He'd have to learn to get in touch with my father.	He had an ok relationship with my parents, but mostly because they supported his separation.
The air was cold and the sky was grey when she clumsily rose from her bed. Two more days until her birthday, but she didn't feel much like celebrating this year. She focused her mind on the hot coffee and donuts that her boss brought every Friday, and wrapped an extra scarf around her neck to combat the chill. "Just keep going" she thought to herself. "Just keep going until you get another letter.	Although she's tempted, she couldn't quite believe it was there.	Although she was tempted, she didn't check the mailbox as she walked to the street.
Today was no ordinary day for Mr. Beetle. He had had a long conversation with his daughter. She was a piece of work. She had just finished drinking from a dirty puddle that had an old diaper in it. Sheesh, thought Mr. Beetle, even beetles should have standards. Not to mention her strange affinity for spicy corn-based snacks.	He knew that the job about him was a way to anyway.	But none of that was the subject of their long conversation this day.

Table 8.1: Examples of generated suggestions and corresponding modifications with their preceding story context

Verbs	be(147), know(110), go(105), would(73), want(70), say(69), think(63), look(56), tell(44), stay(41), feel(37), ask(33), come(30), happen(28), get(24), wait(23), have(22), find(21), whisper(20), talk(18), leave(18), take(17), see(17), pull(16), open(16), sit(15), need(15), hold(15), expect(14)
Nouns	time(59), way(42), monster(36), thing(33), hand(29), man(27), door(27), world(25), eye(24), night(23), life(22), father(22), girl(21), body(20), person(19), voice(18), day(18), face(17), room(16), blood(15), bed(15), mind(14), idea(14), head(14), chance(14), truth(13), word(12)
Pronouns	I(638), she(346), it(212), you(203), he(145), her(113), they(77), me(77), him(74), we(53)
Adjectives	sure(58), able(45), good(28), little(20), sorry(19), right(17), ready(17), real(15), glad(14), troubled(12), beautiful(12), afraid(12), wrong(11), long(11), new(10), strong(9), true(8), alive(8)
Adverbs	not(218), maybe(27), away(10), probably(6), home(6), forward(6), especially(6), right(5)
Bigrams	not know(26), not sure(25), be go(22), not want(18), be sorry(14), not go(11), feel like(10), not expect(8), look like(7), not think(6), monster say(6), not handle(5), good idea(5)
Trigrams	not be able(8), able to stay(8), want to know(7), be a good(6), want to stay(5), able to handle(5)

Table 8.2: Most common generated words/phrases in the suggestions

up as characters. The model has an interesting preoccupation with being *sure* (or *not sure*), as well as the capacity to do something (*able, not be able, able to handle, not handle*). Apologies are common (*sorry*). References to cognitive and emotional states also appear often, indicated by the adjectives *glad, troubled*, and *afraid* as well as the verbs *want, feel* and *think*. The model makes value judgments about things being *good, right, wrong,* and *true*. Negations (*not*) are overwhelming common, particularly in the context of *not know*, *not sure*, *not want*, *not go*, *not expect*, *not think*, and *not handle*. Moreover, the model tends to hedge with the

adverb *maybe* as well as *probably*. Finally, the act of transferring or coordinating locations also shows up (*go, leave, stay, come, wait, away*).

8.5.2 Qualitative Feedback

Table 8.3 shows some of the authors' written feedback on the questionnaire about their experience with the application. Overall, many authors found the task interesting and expressed enthusiasm about future applications that provide creative writing support. They indicated these applications could be useful for dealing with writers' block and facilitating the pace of writing. However, the most common piece of feedback was that the suggestions were not coherent. In some cases, users found them to be completely nonsensical. Users indicated differences in how they approached the task. Some said they already had a story mentally planned out and this caused them to reject the suggestions, while others expressed that they were willing to divert their story in order to accommodate the suggestions. In terms of content, a few authors remarked that the suggestions had erotic material, consistent with the frequency of romance-related words displayed in Table 8.2. This probably reflects the large proportion of romance novels in the training corpus. Also consistent with the pattern of dialogue-related terms, a few users reported that the suggestions too often consisted of dialogue and this didn't necessarily fit with their particular story. Finally, a few users expressed that they wanted the suggestions to be less generic and more concrete in their word choices, and this possibly corresponds to the frequency of abstract nouns we discovered.

8.5.3 Quantitative Analysis

Given the resulting dataset of suggestion-modification pairs, we first quantified the degree to which authors edited the suggestions. As we did in Roemmele and

"The best part was how they kept my momentum going. When I wanted to stop writing, I'd press the magic button and it'd give me a wave of motivation and energy to keep going. I never used the actual phrases that I was given because many of them were too ridiculous and kind of out of context, but they were really perfect in terms of giving me an actual physical action that could come next– especially since I think that's where my weak point is in writing."
"I tried a constrained writing exercise of coming up with one sentence on my own and then using a help sentence in between each one. It worked quite well, although if writing an actual story that I expected to publish, I would take the time to fix the grammar of the suggested sentence and also edit it to use more precise language, especially in the adjectives department."
"More interesting than helpful, in the sense of shifting the focus from just writing a story to playing around with how the system would/could contribute at the price of narrative coherence. But I can see how this would be a good tool for much more sporadic use in case of writing block."
"It's very interesting. I thought it was funny, but I'll admit I had a whole story going in my head and I tried not to stray from it even after the addition of the \help\ sentence. I may have been able to keep things going, or at least modify the help sentences to create something."
"The suggestions didn't generally fit my story at all. I just wrote around them, changing story direction to incorporate them. Quite a few were borderline nonsensical, as I pointed out within the story I wrote."

Table 8.3: Sample of participant feedback

Gordon (2015), we calculated the similarity between each suggestion and corresponding modification in terms of Levenshtein edit distance (Levenshtein, 1966). Edit distance counts the number of operations required to transform one string into another, where an operation is an addition, deletion, or substitution of a single character. We computed this similarity as $1 - \frac{distance(suggestion,modification)}{max(|suggestion|,|modification|)}$, so higher values indicate more similarity. The mean similarity score for the current dataset was 0.695 (SD=0.346), indicating that authors most often chose to retain large parts of the suggestions instead of fully rewriting them. It is certainly possible to use other metrics besides Levenshtein to quantify similarity, such as similarity in terms of word embeddings. Vector-based measures may capture more latent

similarity where the modification alters the surface text to alternatively express the same story event or idea conveyed by the suggestion.

Comparison between Suggestions and Modifications

We applied the automated linguistic analyses from the previous chapter to compare the features of the suggestions before and after being modified by the authors. Our motivation was that human-authored continuations were used as a gold standard in the previous chapter. The Creative Help task is different in that authors use the generated sentences as a reference for this gold standard rather than independently writing their own. However, we expected the authors' modifications to generally improve the quality of the suggestions and that these improvements would be reflected by the metrics, in the same way the human-authored sentences were shown to differ from the generated sentences in the previous chapter. We very briefly review each metric here; see Chapter 7 for their full description and their relevance to story generation. Metrics 1-9 analyze the generated sentences by themselves (story-independent metrics); Metrics 10-16 analyze them with regard to their context (story-dependent). The context for the story-dependent metrics is all text that appears before the suggestion. For these metrics, we omitted sentences that did not have a context, i.e. those that appeared as the first sentence in the story. There were 910 suggestions with a context in this dataset.

Sentence Length: the number of words in each suggestion (Metric 1).

Grammaticality: the proportion of grammatically correct words in the suggestion (Metric 2).

Lexical Diversity: the proportion of unique words (types) in the suggestions relative to the number of total word occurrences (tokens) (Metric 3), and the total

proportion of unique trigrams relative to the total number of trigram occurrences (Metric 4).

Lexical Frequency: the average inverse log frequency of the words in the suggestions (Metric 5, where higher means less frequent), according to Good-Turing smoothed counts in the Reddit Comment Corpus.

Syntactic Complexity: the total number of noun phrases (Metric 6), words per noun phrase (Metric 7), number of verb phrases (Metric 8), and words per verb phrase (Metric 9) in the sentences, all normalized by sentence length.

Lexical Cohesion: the Jaccard similarity (Metric 10), GloVe word embedding similarity (Metric 11), and skip-thought sentence similarity (Metric 12) between the suggestion and its context.

Style Consistency: the similarity in the distributions of POS tags (Metric 13) and Jaccard similarity of POS trigrams (Metric 14) between the suggestion and its context.

Entity Coreference Rate: the proportion of entities in each suggestion that corefer to an entity in the corresponding context (Metric 15).

Sentiment Similarity: the inverse distance between the sentiment scores of the suggestion and its context based on the DepecheMood sentiment analyzer.

Table 8.4 shows the average scores for each set of sentences. Gray rows indicate features that varied significantly between the generated suggestions and corresponding modifications, which we determined through two-sample Monte Carlo permutation tests with $p < 0.025$. Authors' edits introduced more unique words (Metric 3) and rare words (Metric 5) to the suggestions. The authors increased the syntactic complexity of the suggestions: their edits resulted in an increase in noun phrases and verb phrases as well as longer verb phrases. The automated

	Suggested	Modified
1. Sentence length	15.67	15.11
2. Grammaticality	0.992	0.992
3. Type-token ratio	0.097	0.124
4. Unique trigram ratio	0.791	0.820
5. Word frequency	6.27	6.48
6. # of noun phrases	0.234	0.244
7. Noun phrase length	0.130	0.132
8. # of verb phrases	0.189	0.233
9. Verb phrase length	0.220	0.286
10. Jaccard similarity	0.056	0.059
11. GloVe similarity	0.912	0.913
12. Skip-thought similarity	0.722	0.785
13. Word POS similarity	0.627	0.642
14. Trigram POS similarity	0.051	0.053
15. Entity coreference rate	0.540	0.572
16. Sentiment similarity	0.986	0.985

Table 8.4: Linguistic metric scores for suggestions vs. modifications

grammaticality measure did not detect changes in the number of errors in the sentences, which is especially surprising since many authors gave feedback that there were grammatical errors in the suggestions. As mentioned in the previous chapter, it is possible that the tool we used is not sensitive to all grammatical errors. The authors edited the suggestions to be more similar in terms of skip-thought vectors, but no other differences on the story-dependent measures were significant. This may be because authors did not heavily edit the sentences overall.

Predicting Helpful Suggestions

Beyond examining the difference in features between the suggested and modified sentences, we also wanted to determine if the features of a suggestion would predict how much it is modified by the author. To do this, we ran a Spearman correlation

	ρ
1. Sentence length	-0.082
2. Grammaticality	0.097
3. Type-token ratio	-
4. Unique trigram ratio	-
5. Word frequency	0.058
6. # of noun phrases	0.112
7. Noun phrase length	0.052
8. # of verb phrases	0.001
9. Verb phrase length	-0.022
10. Jaccard similarity	0.017
11. GloVe similarity	0.105
12. Skip-thought similarity	0.258
13. Word POS similarity	-0.037
14. Trigram POS similarity	-0.023
15. Entity coreference	0.134
16. Sentiment similarity	0.107

Table 8.5: Correlation ρ between metric scores for suggestions and similarity to modifications

analysis between the metric scores for the suggestions and their Levenshtein similarity to the corresponding modifications. Features that significantly correlate with Levenshtein similarity can be interpreted as being helpful in influencing authors to make use of the original suggestion in their story (i.e. to apply fewer edits). Table 8.5 shows the resulting correlation coefficients (ρ). Note that the lexical diversity features (Metrics 3 and 4) are not included in these results because they are scored for an entire set of sentences, not on the level of an individual sentence. Statistically significant correlations ($p < 0.005$) are highlighted in gray, indicating that suggestions with higher scores on these metrics were particularly helpful to authors. Suggestion length did not have a significant impact, but grammaticality emerged as a helpful feature. The frequency scores of the words in the suggestions did not significantly influence their helpfulness. In terms of syntactic complexity,

suggestions with more noun phrases were edited less often, but verb complexity was not predictive of edits. For lexical cohesion, the number of overlapping words between the suggestion and its context (Jaccard similarity) was not predictive, but vector-based similarity was an indicator of helpfulness. Similarity in terms of sentence (skip-thought) vectors was especially impactful, which suggests these representations are indeed useful for modeling coherence between neighboring sentences in a story. Neither metric for style similarity predicted authors' edits, but suggestions that more frequently coreferred to entities introduced in the context were more helpful. Finally, sentiment similarity between the suggestion and context was significantly helpful. Overall, the significance of these features supports existing research about their relevance to story generation, and indicates their particular applicability to this story continuation task. These metrics are certainly only a small sample of those that are relevant for characterizing generation quality for this task, in particular since they focus more on linguistic coherence than on creativity. As discussed in Chapter 7, it is extremely difficult to quantify what makes a text creative. Moroever, there are many important features of narrative text that are also challenging to capture automatically, such as imagery, thematic coherence, and suspense. As approaches for modeling these features begin to emerge, they can be readily incorporated in this framework to determine their impact in this application.

8.6 Conclusion

In this work, we integrated an RNN LM model into the Creative Help platform for automated story writing assistance. The novelty of this work is in the analysis of users' interactions with the application. We analyzed the generated sentences in

terms of the automated linguistic analyses that characterized differences between the models in the previous chapter. We showed that authors' modifications significantly changed some of these features of the sentences. If we assume that authors' modifications are a type of gold standard for generation quality, this provides some evidence of the usefulness of these features for evaluating story generation, which is what we proposed in Chapter 7. Furthermore, we revealed that some of these features of the suggestions made authors more likely to incorporate the suggestions into their stories.

This work makes a broader contribution beyond story generation specifically: the demonstrated framework can be scaled to determine the influence of any feature in an automated writing support application where authors can adapt generated content. The objective of this approach is to leverage data from user interactions with the system to establish an automated feedback loop for evaluation, by which features that emerge as helpful can be promoted in future systems.

In terms of the Creative Help application itself, the obvious trajectory of this research is to develop models that are effective in supporting creative story writing. Rather than trying to model how people actually generate creative language, this work examined how a simple machine learning model trained on creative text (specifically, fiction stories) and applied to generate sequences with some degree of unpredictability can be used to simulate creative suggestions. As models become deeper and more sophisticated, there is an important question of which story elements benefit most from being unpredictable. Randomness in the surface language of the story is different from randomness in the plot events of the story, for instance. In this work, these are obviously conflated, since we use a shallow RNN that does not recognize the distinction. We can easily imagine a model that promotes the

unpredictability of these plot events while ensuring the grammaticality of the language used to express these events. This also motivates the need for evaluation metrics that are sensitive to these different layers of generation in determining the impact of their unpredictability on authors' creativity.

Chapter 9

Conclusion

Every story I write adds to me a little, changes me a little, forces me to reexamine an attitude or belief, causes me to research and learn, helps me to understand people and grow.

Octavia E. Butler

This thesis examined the emerging task of narrative continuation, which is embedded in research on story generation. I examined this task largely from the perspective of natural language processing while also interfacing with other components of story generation research such as commonsense reasoning and computational creativity. The work described here is among the first to explore a neural network paradigm for this endeavor. This paradigm departs from many traditional AI approaches in trying to induce a latent representation of a story from its surface text, without having access to any explicitly provided story structure. I demonstrated that this approach is useful for interfacing with the inherent natural language people use to tell stories, without imposing any requirements on them to annotate or constrain their text. This work provides an additional reference point for future work on narrative continuation. Given the explosion of neural-based approaches across AI, this work is particularly useful as a baseline to frame the

progress made by new techniques. To restate the particular contributions of this thesis:

- An approach to predicting story continuations based on cause-effect relations (Chapter 4)
- An approach to predicting the likely ending of a given story (Chapter 5)
- A demonstration of these approaches for facilitating natural-language user input as the vehicle of interaction in a storytelling application, and a comparison of their impact relative to other models (Chapter 6)
- A comparison between a neural approach to generating free-text story continuations and other methods, according to a set of automated linguistic metrics (Chapter 7)
- A demonstration of this approach to free-text generation in an application that provides automated support for story writing (Chapter 8)
- An approach to evaluating generation within this application based on automated linguistic metrics (Chapter 8)

9.1 Summary of Closed-Choice Prediction

In closed-choice prediction, a set of human-authored candidates is provided for the next segment in a story, and a system must select the most suitable continuation among them. On an abstract level, humans have a strong ability to anticipate what is likely to happen next in a scenario, so this task serves as an evaluation framework for automated systems that model this capacity. Here these scenarios are expressed in natural language in this work, so this is also a type of machine

reading comprehension task. Our approach focused on learning probable continuations by reading story text. For predicting cause-effect sequences (COPA) in Chapter 4, we used a neural encoder-decoder model to estimate the likelihood that one sequence follows another in a story by directly modeling word probabilities. For predicting the correct ending of a story (the Story Cloze Test) in Chapter 5, we used an RNN-based binary classifier to distinguish transitions from story contexts to correct and incorrect endings, respectively. Despite the inherent value of these frameworks for evaluation, closed-choice prediction is not just a toy AI task; Chapter 6 shows that it can be integrated into interactive storytelling applications.

We observed that our neural models could perform closed-choice prediction tasks to some degree, but nowhere near the level of human performance. We interpret from this that a lot of knowledge for story continuation can be captured by the hidden layer of the network just by observing stories on a lexical surface level. But there are other informative signals for the task that were not harnessed by our models. We can speculate a few reasons for this. First of all, it is unclear whether the relevant sequences for prediction at test time were contained in the training data in the first place. Obviously, the breadth of commonsense knowledge gleaned from human experiences is massive. Our primary training corpus for the COPA and Story Cloze Test consisted of only 100,000 stories, not enough to capture all the different scenarios about which people share common expectations. In theory, neural networks are designed to account for this sparsity. Moreover, we tried to leverage resources like word embeddings, for example, to generalize knowledge about one situation to similar ones with slightly different details. If we continue to rely on existing story corpora for the story continuation task, then we likely need approaches that can further recognize analogies between stories. Another gap in

our approach is that the models were not given explicit guidance about which components of the stories were particularly important for prediction. For example, the encoder-decoder approach in Chapter 4 learned to maximize the probability of all content words in a sequence given those in the previous sequence, and the best-performing model in Chapter 5 learned to distinguish between words in correct endings and those in random endings from other stories. However, as discussed in those chapters, there were systematic differences between correct and incorrect continuations such that certain words were more important for distinguishing between them than others. For instance, the incorrect continuations in these tasks were unlikely to reference story characters that had not already been introduced in the context. During training, the models would have learned to assign higher probability to continuations that contained familiar characters relative to those that referenced new characters. While this is not wrong, it may have had limited relevance for prediction in this particular setting, since often both the correct and incorrect continuations referenced the same characters from the context. This problem of relevance also applies to other prediction tasks in NLP, so neural mechanisms like attention layers (Bahdanau et al., 2014) have been developed to learn the relative influence of different components of an input sequence on prediction. Incorporating these techniques is a target of future work on this task, but there is still the general problem that the features that distinguish human-defined plausible and implausible continuations are not explicitly given in naturally occurring stories used as training corpora.

9.2 Summary of Free-text Generation

Story continuation via free-text generation is complementary to closed-choice prediction: systems that perform the former should also be able to select correct continuations in the latter. Our approaches to both made use of some of the same techniques. However, the closed-choice prediction framework cannot fully substitute as an evaluation for the free-text generation task. Of course the goal is to similarly generate a plausible continuation for a given story in the free-text case, but there are a great deal of different continuations that could be considered plausible. Eliciting human judgments of quality is a standard evaluation approach, but generation can greatly benefit from more immediate feedback on what constitutes high-quality output. Exploring this idea, in Chapter 7, we generated story continuations using an RNN that modeled probable word sequences from a fiction corpus. We analyzed these continuations according to a selected set of linguistic metrics relevant to story generation. We used these analyses to compare these continuations to those of other generation models as well as human-authored continuations, in order to determine features associated with good continuations. Our work in Chapter 8 examined the RNN generation approach within a user application, where a generated continuation served as a suggestion for the next sentence in an author's story. Here, we used the same linguistic analyses to determine features of the continuations that appealed to authors. We conclude that this type of implicit approach to evaluation is valuable in being efficient, unobtrusive to users, and capable of identifying features of generation quality that users may not be conscious of. The difficulty of this approach is in designing metrics that address important dimensions of quality like coherence and originality, as well as narrative-specific elements like plot, theme, and the development of characters and settings.

A complete model for evaluating story quality faces the same difficulties as the task of generating high-quality stories itself.

In terms of the RNN-generated continuations themselves, our analyses demonstrated that these sequences were largely distinct from human-authored continuations. This was shown by their comparison in Chapter 7 in terms of their relative scores on the linguistic metrics, and also by the qualitative results in Chapter 8 represented by authors' feedback about Creative Help. We can informally conclude that our RNN model was capable of producing continuations that were interpretable and often interesting, but not necessarily coherent. This fits with the overall discussion about the impact of these models. For example, the first movie with an automatically generated script, *Sunspring*[1], was recently released. The generation model was very similar to the one used in this work, and was trained on a corpus of existing movie scripts. The movie received praise for the concept itself, but its appeal was in its absurdity rather than in the possibility of convincing viewers that it was human-authored. Existing research has focused on adding constraint mechanisms to RNNs to improve generation, such as by maintaining a 'check-list' of lexical items that should be referenced in the text (Kiddon et al., 2016) or by explicitly modeling coreferential entities (Ji et al., 2017). These approaches could certainly benefit story continuation as well.

It is important to keep in mind that the requirements of free-text story continuation are application-dependent. As thoroughly discussed in Chapters 4 and 5, continuations that reflect the most obvious thing to happen next in the story are ideal for the Story Cloze Test, but may be considered *too* obvious and thus uninteresting by Creative Help users. From a research perspective, it is necessary

[1] youtu.be/LY7x2Ihqjmc

to theorize about what users will find helpful for these systems, but just as important to avoid foregone conclusions about this. As proposed in Chapter 8, it is plausible that a user would want to brainstorm story ideas by observing "absurd" suggestions that can later be molded into something interpretable. Systems should be able to accommodate this rather than assuming there is some universal ideal standard for any particular dimension like coherence. With regard to the above discussion about evaluation, this of course means that evaluation metrics must be sensitive to these varying requirements as well. There is much to be explored in determining how people will utilize AI-based creativity support tools in general. Will they expect to fully offload creative tasks onto these systems and use the final product, or will they seek to use them as an intermediate step in a new creative process? These questions should inform the ways researchers approach the task of automatically generating creative content.

9.3 Final Outlook

NLP researchers have implemented computational approaches to annotating language data based on linguistic theory, with syntactic parsing being a prime example. Semantic-level annotation schemes such as Rhetorical Structure Theory (Mann and Thompson, 1988) and Abstract Meaning Representation (Banarescu et al., 2013) are also well-established, though their automated application is still the target of ongoing research. Computational models of stories can similarly take advantage of proposed theories about the underlying structure of stories. For example, one of the more well-known theories distinguishes the *fabula* (the underlying narrative structure) of a story from its *sjuzet* (the surface expression of narrative structure) (Propp, 1968, e.g.). However, unlike linguistic annotation, few have proposed

approaches for effectively annotating these differing narrative levels. Among the annotation schemes that have been demonstrated, inter-rater agreement tends to be moderate-to-low (Elson, 2012; Rahimtoroghi et al., 2014). The results of our work suggest that it could greatly benefit from a more abstract representation of the sequence of events in a story, for instance. This was the target of the work on narrative event chains described in Chapter 2 (Section 2.2), but I also discussed how these schemes suppress some of the surface features of the story that are important for generation. Incorporating theoretical work on narrative into computational frameworks is an ongoing challenge that applies broadly across all story generation research.

The conceptual motivation behind neural networks is that their hidden layer eschews the need for annotation of manually selected features. Instead, they are expected to induce this knowledge automatically in the process of learning to make predictions. Even though the hidden layer of our models clearly did learn some degree of story structure (at the word level, at least), it is unclear if these numerical vector representations can be conceptualized according to interpretable labels like fabula or plot. This lack of interpretability is a general drawback of neural approaches, and addressing this problem is an emerging research task[2] (Li et al., 2016; Kádár et al., 2017). Moreover, we ideally want to know why a model predicted the story continuation that it did: what were the cues in the input story that influenced its decision? This insight could inform the authoring design of interactive narratives in applications like DINE; for instance, by telling scenario designers why certain user inputs are yielding incoherent story continuations. It can also be used to advance the architecture of the models themselves, by revealing

[2]I also examined this issue outside the work in this thesis, in the context of a different NLP task: civisanalytics.com/blog/interpreting-visualizing-neural-networks-text-processing

if the model is attending to the wrong features for predicting continuations, for example.

As a more long term vision for future work, another opportunity is to perform text-based narrative continuation from non-text data that convey stories, such as images. Resources for generating stories from images have emerged recently (Huang et al., 2016b). My previous work on modeling story-based perception of moving shapes (Gordon and Roemmele, 2014), outside the scope of this thesis, also pertains to this objective. This work lends itself to interactive applications that fit alongside the ones explored in this thesis. For example, creativity support systems could be augmented with the ability to use images in addition to text as cues for suggestions.

To finally conclude, the ambition of this work is to computationally simulate people's ability to convey knowledge through storytelling in a way that engages and inspires others. The purpose of this is not to replace human storytelling, but to make human storytelling even more compelling. This thesis clearly only scratches the surface of this extremely lofty goal, but it conveys some immediate directions for moving further towards it.

Bibliography

Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1): 39–59, 1994.

Shlomo Argamon, Moshe Koppel, and Galit Avneri. Routing documents according to style. In *Proceedings of the First International Workshop on Innovative Information Systems*, pages 85–92, 1998.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, ICLR 2014, 2014.

Mieke Bal. *Narratology: Introduction to the theory of narrative*. University of Toronto Press, 2009.

David Bamman, Brendan O'Connor, and Noah A Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2013, pages 352–361. Association for Computational Linguistics, 2013.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013.

Robert L Bangert-Drowns. The word processor as an instructional tool: A meta-analysis of word processing in writing instruction. *Review of Educational Research*, 63(1):69–93, 1993.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

Scott F. Beers and William E. Nagy. Syntactic complexity as a predictor of adolescent writing quality: Which measures? Which genre? *Reading and Writing*, 22(2):185–200, 2009.

Jenna Bellassai, Andrew Gordon, Melissa Roemmele, Margaret Cychosz, Obiageli Odimegwu, and Olivia Connolly. Unsupervised Text Classification for Natural Language Interactive Narratives. In *Proceedings of the 10th International Workshop on Intelligent Narrative Technologies*, INT10, 2017.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.

Kim Binsted and Graeme Ritchie. Computational rules for generating punning riddles. *HUMOR-International Journal of Humor Research*, 10(1):25–76, 1997.

Joseph A Blass and Kenneth D Forbus. Analogical Chaining with Natural Language Instruction for Commonsense Reasoning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI-17, 2017.

Margaret A Boden. *The creative mind: Myths and mechanisms*. Psychology Press, 2004.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Eric Brill and Robert C. Moore. An Improved Error Model for Noisy Channel Spelling Correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Michael Bugert, Yevgeniy Puzikov, Andreas Rücklé, Judith Eckle-Kohler, Teresa Martin, Eugenio Martínez-Cámara, Daniil Sorokin, Maxime Peyrard, and Iryna Gurevych. LSDSem 2017: Exploring data generation methods for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, pages 56–61, 2017.

Jill Burstein and Magdalena Wolska. Toward Evaluation of Writing Style: Finding Overly Repetitive Word Use in Student Essays. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics*, EACL 2003, pages 35–42, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Jill Burstein, Martin Chodorow, and Claudia Leacock. Criterion ^{SM} Online Essay Evaluation: An Application for Automated Evaluation of Student Essays. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, IAAI 2003, pages 3–10, 2003.

Zheng Cai, Lifu Tu, and Kevin Gimpel. Pay Attention to the Ending: Strong Neural Baselines for the ROC Story Cloze Task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL 2017, pages 616–622. Association for Computational Linguistics, 2017.

Linda Candy. Computers and creativity support: knowledge, visualisation and collaboration. *Knowledge-Based Systems*, 10(1):3–13, 1997.

Marc Cavazza and Fred Charles. Dialogue Generation in Character-based Interactive Storytelling. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'05, pages 21–26. AAAI Press, 2010.

Marc Cavazza, Fred Charles, and Steven J. Mead. Character-Based Interactive Storytelling. *IEEE Intelligent Systems*, 17(4):17–24, July 2002.

Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, ACL 2008, pages 789–797. Association for Computational Linguistics, 2008.

Yuen Ren Chao. Making sense out of nonsense. *The Sesquipedalian*, 7(32), 1997.

Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2016, pages 2358–2367. Association for Computational Linguistics, 2016.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-8, pages 103–111. Association for Computational Linguistics, 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

In *Proceedings of the NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Elizabeth Clark, Anne Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories. In *Proceedings of the 23rd ACM Conference on Intelligent User Interfaces*, IUI'2018, 2018.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Kate Compton, Ben Kybartas, and Michael Mateas. Tracery: An Author-Focused Generative Text Tool. In *Proceedings of the 8th International Conference on Interactive Digital Storytelling*, ICIDS 2015, pages 154–161, 2015.

Chris Crawford. *Chris Crawford on Interactive Storytelling*. New Riders, 2012.

Scott A. Crossley, Jennifer L. Weston, Susan T. McLain Sullivan, and Danielle S. McNamara. The Development of Writing Proficiency as a Function of Grade Level: A Linguistic Analysis. *Written Communication*, 28(3):282–311, 2011.

Maria Cutumisu, Duane Szafron, Jonathan Schaeffer, Matthew McNaughton, Thomas Roy, Curtis Onuczko, and Mike Carbonaro. Generating ambient behaviors in computer role-playing games. *IEEE Intelligent Software*, 21(5):19–27, 2006.

Margaret Cychosz, Andrew Gordon, Obiageli Odimegwu, Olivia Connolly, Jenna Bellassai, and Melissa Roemmele. Effective scenario designs for free-text interactive fiction. In *Proceedings of the 10th International Conference on Interactive Digital Storytelling*, ICIDS 2017, 2017.

Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

Terry Dartnall. *Artificial intelligence and creativity: An interdisciplinary approach*, volume 17. Springer Science & Business Media, 2013.

Amitava Das and Björn Gambäck. Poetic Machine: Computational Creativity for Automatic Poetry Generation in Bengali. In *Proceedings of the Fifth International Conference on Computational Creativity*, ICCC 2014, pages 230–238, 2014.

Nicholas Davis, Chih-Pin Hsiao, Kunwar Yashraj Singh, Lisa Li, and Brian Magerko. Empirically Studying Participatory Sense-Making in Abstract Drawing with a Co-Creative Cognitive Agent. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, pages 196–207. ACM, 2016.

Natalie Dehn. Story Generation After TALE-SPIN. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, IJCAI '81, pages 16–18, 1981.

Aliya Deri and Kevin Knight. How to make a Frenemy: Multitape FSTs for portmanteau generation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2015, pages 206–210. Association for Computational Linguistics, 2015.

Philip Edmonds and Graeme Hirst. Near-synonymy and lexical choice. *Computational linguistics*, 28(2):105–144, 2002.

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

Micha Elsner. Character-based Kernels for Novelistic Plot Structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 634–644, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. Coreference-inspired Coherence Modeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 41–44, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

David Elson. DramaBank: Annotating Agency in Narrative Discourse. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, LREC 2012. European Language Resources Association (ELRA), 2012.

Rudolph Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32 (3):221, 1948.

Michael Flor and Swapna Somasundaran. Sentiment Analysis and Lexical Cohesion for the Story Cloze Task. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, page 62, 2017.

P. Foltz, W. Kintsch, and T. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307, 1998.

Richard P Gabriel, Jilin Chen, and Jeffrey Nichols. InkWell: A Creative Writer's Creative Assistant. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, pages 93–102. ACM, 2015.

Boris A Galitsky and Sergei O Kuznetsov. A web mining tool for assistance with creative writing. In *Proceedings of the 35th European Conference on Information Retrieval*, ECIR 2013, pages 828–831, 2013.

Michael Gamon. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, pages 611–617. Association for Computational Linguistics, 2004.

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. Success with Style: Using Writing Style to Predict the Success of Novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2013, pages 1753–1764. Association for Computational Linguistics, 2013.

Albert Gatt and Emiel Krahmer. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of AI Research*, 60, 2017.

Matt Gerber, Andrew S Gordon, and Kenji Sagae. Open-domain commonsense reasoning using discourse relations from a corpus of weblog stories. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 43–51. Association for Computational Linguistics, 2010.

Pablo Gervás. Computational Approaches to Storytelling and Creativity. *AI Magazine*, 30:49, 2009.

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4-5):235–242, 2005.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating Topical Poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2016, pages 1183–1191. Association for Computational Linguistics, 2016.

Pranav Goel and Anil Kumar Singh. IIT (BHU): System Description for LSDSem17 Shared Task. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, page 81, 2017.

Travis Goodwin, Bryan Rink, Kirk Roberts, and Sanda M Harabagiu. UTDHLT: COPACETIC system for choosing plausible alternatives. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval 2012, pages 461–466. Association for Computational Linguistics, 2012.

Andrew Gordon, Cosmin Adrian Bejan, and Kenji Sagae. Commonsense Causal Reasoning Using Millions of Personal Stories. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI-11, pages 1180–1185, 2011.

Andrew S Gordon and Melissa Roemmele. An authoring tool for movies in the style of Heider and Simmel. In *Proceedings of the Seventh International Conference on Interactive Digital Storytelling*, ICIDS 2014, pages 49–60. Springer, 2014.

Andrew S Gordon, Zornitsa Kozareva, and Melissa Roemmele. SemEval-2012 Task 7: Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, SemEval 2012, pages 394–398. Association for Computational Linguistics, 2012.

Arthur C Graesser, Danielle S McNamara, Max M Louwerse, and Zhiqiang Cai. Coh-Metrix: Analysis of text on cohesion and language. *Behavior Research Methods*, 36(2):193–202, 2004.

Mark Granroth-wilding and Stephen Clark. What Happens Next? Event Prediction Using a Compositional Neural Network Model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI-16, 2016.

Christina Haas. How the writing medium shapes the writing process: Effects of word processing on planning. *Research in the Teaching of English*, pages 181–207, 1989.

Sarah Harmon. FIGURE8: A Novel System for Generating and Evaluating Figurative Language. In *Proceedings of the Sixth International Conference on Computational Creativity*, ICCC 2015, page 71, 2015.

Gail E Hawisher. Research and recommendations for computers and composition. *Critical perspectives on computers and composition instruction*, pages 44–69, 1989.

F. Hill, A. Bordes, S. Chopra, and J. Weston. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. In *Proceedings of the 4th International Conference on Learning Representations*, ICLR 2016, 2016.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Overview of mini-batch gradient descent. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z. Gajos. ChordRipple: Recommending Chords to Help Novice Composers Go Beyond the Ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, pages 241–250, New York, NY, USA, 2016a. ACM.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, Larry Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '16, pages 1233–1239, 2016b.

Molly E Ireland and James W Pennebaker. Language style matching in writing: synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549, 2010.

Shahida Jabeen, Xiaoying Gao, and Peter Andreae. Using Asymmetric Associations for Commonsense Causality Detection. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, PRICAI 2014, pages 877–883. Springer International Publishing, 2014.

Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11 (2):37–50, Feb 1912.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2017, pages 1830–1839. Association for Computational Linguistics, 2017.

Aditya Joshi, Diptesh Kanojia, Pushpak Bhattacharyya, and Mark Carman. Sarcasm Suite: A Browser-Based Engine for Sarcasm Detection and Generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI-17, 2017.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, pages 2342–2350, 2015.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43 (4):761–780, 2017.

Anna Kantosalo, Jukka M Toivanen, Ping Xiao, and Hannu Toivonen. From Isolation to Involvement: Adapting Machine Creativity Software to Support Human-Computer Co-Creation. In *Proceedings of the Fifth International Conference on Computational Creativity*, ICCC 2014, pages 1–7, 2014.

Justine Kao and Dan Jurafsky. A computational analysis of style, affect, and imagery in contemporary poetry. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 8–17. Association for Computational Linguistics, 2012.

Jussi Karlgren and Douglass Cutting. Recognizing Text Genres with Simple Metrics Using Discriminant Analysis. In *Proceedings of the 15th Conference on Computational Linguistics*, COLING '94, pages 1071–1075, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. DeepTingle. In *Proceedings of the Eighth International Conference for Computational Creativity*, ICCC 2017, 2017.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2016, pages 329–339. Association for Computational Linguistics, 2016.

Joy Kim, Justin Cheng, and Michael S. Bernstein. Ensemble: Exploring Complementary Strengths of Leaders and Crowds in Creative Collaboration. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 745–755, 2014.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*, ICLR 2015, San Diego, May 2015.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, NIPS 2015, pages 3294–3302, 2015.

Sheldon Klein, JF Aeschlimann, and DF Balsiger. Automatic novel writing: A status report. *Wisconsin University*, 1973.

Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Transactions on Graphics*, 36(4):48:1–48:11, 2017.

Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992.

William Labov and Joshua Waletzky. Narrative analysis: Oral versions of personal experience. *Sociolinguistics: The essential readings*, pages 74–104, 2003.

Mirella Lapata and Regina Barzilay. Automatic Evaluation of Text Coherence: Models and Representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1085–1090. Morgan Kaufmann Publishers Inc., 2005.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 7(1):1–170, 2014.

P. David Lebling, Marc S. Blank, and Timothy A. Anderson. Zork: A Computerized Fantasy Simulation Game. *Computer*, 12(4):51–59, 1979.

Michael Lebowitz. Story-telling as planning and learning. *Poetics*, 14(6):483–502, 1985.

Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. Shadowdraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2011*, 30(27), 2011.

Thomas M Leitch. *What stories are: Narrative theory and interpretation*. Pennsylvania State University Press, 1986.

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10:707–710, 1966.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. Story Generation with Crowdsourced Plot Graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 598–604. AAAI Press, 2013.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, NAACL-HLT 2016, pages 681–691. Association for Computational Linguistics, 2016.

Antonios Liapis, Georgios N Yannakakis, Constantine Alexopoulos, and Phil Lopes. Can computers foster human users creativity? Theory and praxis of mixed-initiative co-creativity. *Digital Culture & Education*, 8(2):136–152, 2016.

Maria Teresa Llano, Rose Hepworth, Simon Colton, Jeremy Gow, John Charnley, Mark Granroth-wilding, and Stephen Clark. Baseline Methods for Automated Fictional Ideation. In *Proceedings of the 5th International Conference on Computational Creativity*, ICCC 2014, 2014.

Zhiyi Luo, Yuchen Sha, Kenny Q. Zhu, Seung-won Hwang, and Zhongyuan Wang. Commonsense Causal Reasoning between Short Texts. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning*, KR-2016, 2016.

N. Macdonald, L. Frase, P. Gingrich, and S. Keenan. The Writer's Workbench: Computer Aids for Text Analysis. *IEEE Transactions on Communications*, 30 (1):105–110, 1982.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. DopeLearning: A Computational Approach to Rap Lyrics Generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 195–204. ACM, 2016.

Enrique Manjavacas, Folgert Karsdorp, Ben Burtenshaw, and Mike Kestemont. Synthetic Literature: Writing Science Fiction in a Co-Creative Process. In *Proceedings of the 2017 Workshop on Computational Creativity in Natural Language Generation*, CC-NLG 2017, pages 29–37, 2017.

William C Mann and Sandra A Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. Learning a Probabilistic Model of Event Sequences from Internet Weblog Stories. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*, FLAIRS-21, pages 159–164, 2008.

Hisar Maruli Manurung, Hisar Maruli Manurung, Graeme Ritchie, Graeme Ritchie, Henry Thompson, and Henry Thompson. Towards A Computational Model of Poetry Generation. In *Proceedings of the AISB Symposium on Creative and*

Cultural Aspects and Applications of AI and Cognitive Science, pages 79–86, 2000.

Michael Mateas. *An Oz-centric review of interactive drama and believable agents.* Springer, 1999.

Michael Mateas and Andrew Stern. Façade : An Experiment in Building a Fully-Realized Interactive Drama. In *Proceedings of the Game Developers' Conference: Game Design Track*, volume 2, page 82, 2003.

Neil McIntyre and Mirella Lapata. 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*, ACL '09, pages 217–225, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Danielle S. McNamara, Scott A. Crossley, and Philip M. McCarthy. Linguistic Features of Writing Quality. *Written Communication*, 27(1):57–86, 2010.

James R. Meehan. TALE-SPIN, an Interactive Program That Writes Stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, pages 91–98, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

Todor Mihaylov and Anette Frank. Story Cloze Ending Selection Baselines and Data Examination. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, 2017.

T Mikolov, M Karafiat, L Burget, J Cernocky, and S Khudanpur. Recurrent Neural Network based Language Model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, INTERSPEECH 2010, pages 1045–1048, 2010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, NIPS 2013, pages 3111–3119, 2013.

Marcin Miłkowski. Developing an Open-source, Rule-based Proofreading Tool. *Software: Practice and Experience*, 40(7):543–566, June 2010.

Nick Montfort. *Twisty Little Passages: an approach to interactive fiction.* MIT Press, 2005.

Dan Morris, Ian Simon, and Sumit Basu. Exposing parameters of a trained dynamic model for interactive music creation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, AAAI-08, pages 784–791, 2008.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2016, pages 839–849. Association for Computational Linguistics, 2016.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael William Chambers, and James F. Allen. LSDSem 2017 Shared Task: The Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, pages 1–5, 2017.

Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3834–3846, New York, NY, USA, 2016. ACM.

Eric Nichols, Dan Morris, and Sumit Basu. Data-driven Exploration of Musical Chord Sequences. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 227–236, New York, NY, USA, 2009. ACM.

Hugo Gonçalo Oliveira. PoeTryMe: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21, 2012.

Edward Packard. *Cave of Time (Choose Your Own Adventure #1)*. Bantam, 1982.

James W Pennebaker. Telling stories: The health benefits of narrative. *Literature and medicine*, 19(1):3–18, 2000.

James W. Pennebaker, Cindy K. Chung, Joey Frazee, Gary M. Lavergne, and David I. Beaver. When Small Words Foretell Academic Success: The Case of College Admissions Essays. *PLOS ONE*, 9(12):1–10, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, pages 1532–1543. Association for Computational Linguistics, 2014.

Rafael Pérez y Pérez and Mike Sharples. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139, 2001.

Karl Pichotta and Raymond J. Mooney. Learning Statistical Scripts with LSTM Recurrent Neural Networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2800–2806. AAAI Press, 2016.

Emily Pitler and Ani Nenkova. Revisiting Readability: A Unified Framework for Predicting Text Quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 186–195, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Julie Porteous, Marc Cavazza, and Fred Charles. Applying Planning to Interactive Storytelling: Narrative Control Using State Constraints. *ACM Transactions on Intelligent Systems and Technology*, 1(2):10:1–10:21, December 2010.

Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: using an LSTM for Automatic Rap Lyric Generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2015, pages 1919–1924. Association for Computational Linguistics, 2015.

Keshav Prasad, Kayla Briët, Obiageli Odimegwu, Olivia Connolly, Diego Gonzalez, and Andrew S Gordon. The Long Walk From Linear Film to Interactive Narrative. In *Proceedings of the 10th Workshop on Intelligent Narrative Technologies*, INT10, 2017.

Vladimir Propp. *Morphology of the Folktale*, volume 9. University of Texas Press, 1968.

Mari Carmen Puerta Melguizo, Lou Boves, and Olga Muñoz Ramos. A proactive recommendation system for writing: Helping without disrupting. *International Journal of Industrial Ergonomics*, 39(3):516–523, 2009.

James Pustejovsky, José M Castano, Robert Ingria, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the AAAI Spring Symposium on New Directions in Question-Answering*, 2003.

Elahe Rahimtoroghi, Thomas Corcoran, Reid Swanson, Marilyn A Walker, Kenji Sagae, and Andrew Gordon. Minimal narrative annotation schemes and their applications. In *Proceedings of the 7th Workshop on Intelligent Narrative Technologies*, INT7, 2014.

W Michael Reed. The Effectiveness of Composing Process Software : An Analysis of Writer's Helper. *Computers in the Schools*, 6(1-2):67–82, 1989.

Mark O. Riedl and R. Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

Mark O Riedl, Andrew Stern, Don Dini, and Jason Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, 4(2):23–42, 2008.

David L Roberts, Mark J Nelson, Charles L Isbell, Michael Mateas, and Michael L Littman. Targeting specific distributions of trajectories in MDPs. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI'06. AAAI Press, 2006.

Melissa Roemmele and Andrew S Gordon. Creative Help: A Story Writing Assistant. In *Proceedings of the 15th International Conference on Interactive Digital Storytelling*, ICIDS 2015. Springer International Publishing, 2015.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *Proceedings of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, pages 90–95, 2011.

Melissa Roemmele, Andrew S Gordon, and Reid Swanson. Evaluating Story Generation Systems Using Automated Linguistic Analyses. In *Proceedings of the SIGKDD 2017 Workshop on Machine Learning for Creativity*, 2017a.

Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew Gordon. An RNN-based Binary Classifier for the Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, pages 74–80, 2017b.

Melissa Roemmele, Paola Mardo, and Andrew Gordon. Natural-language Interactive Narratives in Imaginal Exposure Therapy for Obsessive-Compulsive Disorder. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology—From Linguistic Signal to Clinical Reality*, pages 48–57, 2017c.

Christian Roth, Christoph Klimmt, Ivar E Vermeulen, and Peter Vorderer. The experience of interactive storytelling: comparing Fahrenheit with Façade. In *Proceedings of the Tenth International Conference on Entertainment Computing*, ICEC 2011, pages 13–21. Springer, 2011.

Jonathan P Rowe, Eun Young Ha, and James C Lester. Archetype-driven character dialogue generation for interactive narrative. In *Proceedings of the International Workshop on Intelligent Virtual Agents*, IVA 2008, pages 45–58. Springer, 2008.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (Third edition)*. Pearson, 2009.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. Reassessing the Goals of Grammatical Error Correction: Fluency Instead of Grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182, 2016.

Shota Sasaki, Sho Takase, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. Handling Multiword Expressions in Causality Estimation. In *Proceedings of the 12th International Conference on Computational Semantics–Short papers*, IWCS 2017, 2017.

Roger C Schank and Robert P Ableson. Knowledge and memory: The real story. *In. Rober S. Wyer (Ed.), Knowledge and memory: The real story*, pages 1–85, 1995.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A Smith. The Effect of Different Writing Tasks on Linguistic Style: A Case Study of the ROC Story Cloze Task. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, CoNLL 2017, 2017a.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, Noah A Smith, and Computer Science. Story Cloze Task: UW NLP System. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, LSDSem 2017, pages 52–55, 2017b.

Abigail See, Peter J Liu, and Christopher D Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2017, pages 1073–1083, 2017.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building End-to-end Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3776–3783. AAAI Press, 2016.

Burr Settles. Computational Creativity Tools for Songwriters. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to*

Linguistic Creativity, CALC '10, pages 49–57, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Lifeng Shang, Zhengdong Lu, and Hang Li. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53th Annual Meeting of Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP'15, pages 1577–1586, 2015.

Manu Sharma, Santiago Ontañón, Manish Mehta, and Ashwin Ram. Drama management and player modeling for interactive fiction games. *Computational Intelligence*, 26(2):183–211, 2010.

Ben Shneiderman. Creating Creativity: User Interfaces for Supporting Innovation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):114–138, March 2000.

Mei Si, Stacy C. Marsella, and David V. Pynadath. Directorial Control in a Decision-Theoretic Framework for Interactive Narrative. In *Proceedings of the 2nd International Conference on Interactive Digital Storytelling*, ICIDS '09, pages 221–233, Berlin, Heidelberg, 2009. Springer-Verlag.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M Mitchell. Inferring Interpersonal Relations in Narrative Summaries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI-16, pages 2807–2813, 2016.

Jacopo Staiano and Marco Guerini. Depeche Mood: a Lexicon for Emotion Analysis from Crowd Annotated News. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, NAACL-HLT 2014, pages 427–433, 2014.

Franz K Stanzel. *A Theory of Narrative*. Cambridge University Press, 1979.

Amanda Stent and Srinivas Bangalore. *Natural Language Generation in Interactive Systems*. Cambridge University Press, 2014.

Oliviero Stock and Carlo Strapparava. The act of creating humorous acronyms. *Applied Artificial Intelligence*, 19(2):137–151, 2005.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM Neural Networks for Language Modeling. In *Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association*, INTERSPEECH 2012, pages 194–197, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, NIPS 2014, pages 3104–3112, 2014.

Reid Swanson and Andrew S Gordon. A Comparison of Retrieval Models for Open Domain Story Generation. In *Proceedings of the AAAI 2009 Spring Symposium on Intelligent Narrative Technologies II*, pages 119–126, 2009.

Reid Swanson and Andrew S. Gordon. Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. *ACM Transactions on Interactive Intelligent Systems*, 2(3):1–35, 2012.

John Sweller. Cognitive bases of human creativity. *Educational Psychology Review*, 21(1):11–19, 2009.

Chuanqi Tan, Furu Wei, Li Dong, Weifeng Lv, and Ming Zhou. Solving and Generating Chinese Character Riddles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2016, pages 846–855. Association for Computational Linguistics, 2016.

Mariet Theune, Sander Faas, Er Faas, Anton Nijholt, and Dirk Heylen. The Virtual Storyteller: Story Creation by Intelligent Agents. *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment Conference*, pages 204–215, 2003.

David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive Storytelling : A Player Modelling Approach. In *Proceedings of the 3rd Artificial Intelligence and Interactive Digital Entertainment Conference*, AIIDE 2007, pages 43–48, 2007.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop. *COURSERA: Neural networks for machine learning*, 2012.

Tom Trabasso. Causal cohesion and story coherence. In *Proceedings of the Annual Meeting of the American Educational Research Association*, New York, NY, 1982.

Scott R Turner. A case-based model of creativity. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, 1993a.

Scott R Turner. *MINSTREL: A computer model of creativity and storytelling*. University of California at Los Angeles, 1993b.

Kees van Deemter, Brigitte Krenn, Paul Piwek, Martin Klesen, Marc Schröder, and Stefan Baumann. Fully Generated Scripted Dialogue for Embodied Agents. *Artificial Intelligence*, 172(10):1219–1244, June 2008.

Tony Veale. *Exploding the Creativity Myth: The Computational Foundations of Linguistic Creativity*. Bloomsbury Academic, 2012.

Tony Veale and Yanfen Hao. Comprehending and Generating Apt Metaphors: A Web-driven, Case-based Approach to Figurative Language. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1471–1476. AAAI Press, 2007.

Ernst von Glasersfeld. The problem of syntactic complexity in reading and readability. *Journal of Literacy Research*, 3(2):1–14, 1970.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. Chinese song iambics generation with neural attention-based model. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI-16, pages 2943–2949, 2016.

Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. LyriSys: An Interactive Support System for Writing Lyrics Based on Topic Transition. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, pages 559–563. ACM, 2017.

Joseph Weizenbaum. ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of the ACM*, 9(1):36–45, 1966.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Weiwei Yang, Xiaofei Lu, and Sara Cushing Weigle. Different topics, different discourse: Relationships among writing topic, measures of syntactic complexity, and judgments of writing quality. *Journal of Second Language Writing*, 28:53 – 67, 2015.

R Michael Young, Mark O Riedl, Mark Branly, Arnav Jhala, RJ Martin, and CJ Saretto. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51–70, 2004.

Boliang Zhang, Hongzhao Huang, Xiaoman Pan, Heng Ji, Kevin Knight, Zhen Wen, Yizhou Sun, Jiawei Han, and Bulent Yener. Be appropriate and funny: Automatic entity morph encoding. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL 2014, pages 706–711. Association for Computational Linguistics, 2014.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. Ordinal common-sense inference. In *Proceedings of the Sixteenth Conference on Theoretical Aspects of Rationality and Knowledge*, TARK 2017, 2016.

Xingxing Zhang and Mirella Lapata. Chinese Poetry Generation with Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, pages 670–680. Association for Computational Linguistics, 2014.

Xiaojin Zhu, Zhiting Xu, and Tushar Khot. How Creative is Your Writing? A Linguistic Creativity Measure from Computer Science and Cognitive Psychology Perspectives. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 87–93, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.