Guiding Neural Story Generation with Reader Models

Xiangyu Peng, Kaige Xie, Amal Alabdulkarim, Harshith Kayam, Samihan Dani, Mark O. Riedl
Georgia Institute of Technology
`{xpeng62,kaigexie,amal,hkayam3,sdani30,riedl}@gatech.edu`

Abstract

Automated storytelling has long captured the attention of researchers for the ubiquity of narratives in everyday life. However, it is challenging to maintain coherence and stay on-topic toward a specific ending when generating narratives with neural language models. In this paper, we introduce Story generation with Reader Models (StoRM), a framework in which a *reader model* is used to reason about the story should progress. A reader model infers what a human reader believes about the concepts, entities, and relations about the fictional story world. We show how an explicit reader model represented as a knowledge graph affords story coherence and provides controllability in the form of achieving a given story world state goal. Experiments show that our model produces significantly more coherent and on-topic stories, outperforming baselines in dimensions including plot plausibility and staying on topic. Our system also outperforms outline-guided story generation baselines in composing given concepts without ordering.

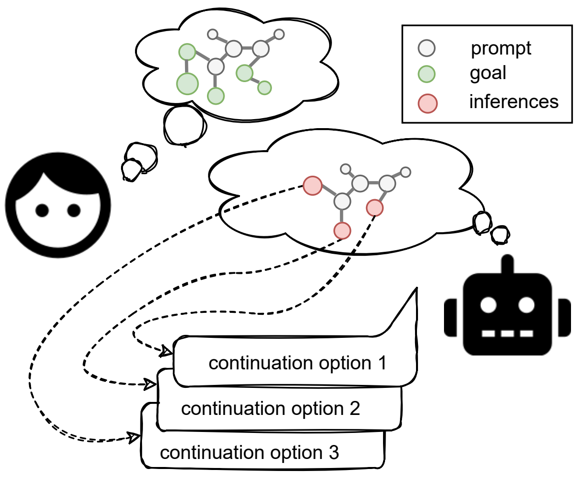


Figure 1: The overview of StoRM system. Our goal is to build a story world covering all the given concepts 🟢. 1. A set of concepts 🟢 and a prompt ○ starts the story generation process. 2. The system builds a goal story world and prompt story world with knowledge graph. 3. The system infers a graph of concepts 🔴. 4. A language model generates continuation options on inferences. 5. *Topk* continuations minimize its difference with goal story world are added to story.

1 Introduction

Automated Story Generation is the challenge of designing an artificial intelligence system that can generate a natural language text that is perceived by readers as a story. Automated Story Generation has been a topic of research since the beginning of the field of artificial intelligence, and has undergone a number of transformations in this time.

Early work on story generation used symbolic planning (Meehan, 1976; Lebowitz, 1987; Cavazza et al., 2003; Porteous and Cavazza, 2009; Riedl and Young, 2010; Ware and Young, 2010; Ware and Siler, 2021). These systems would be provided with a description of the initial world state—usually a list of predicates—and a goal—a description of what predicates should be true to be successful. These approaches had two benefits. First, the plots tended to be coherent because of logical constraints on the actions. Second, the plots were guaranteed to end in a state in which the goal held. However, these systems require substantial knowledge engineering of logical constraints, limiting their generality, and don't always generate plot or stories in natural language.

Recently, neural language modeling approaches (Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2018; Clark et al., 2018; Yao et al., 2019; Rashkin et al., 2020; Fan et al., 2019; Ammanabrolu et al., 2021a) have been applied to story generation because they circumvent the need for manual knowledge engineering and tend to produce relatively fluent, varied, and naturalistic language. Language models are, however, not goal-directed. That is, one cannot natively provide both a context prompt and a goal to be achieved after an arbitrary number of continuations. Further, language models struggle with maintaining story coherence—the logical progression of events—and may also become repetitive. Large, pre-trained language models improve fluency and generalization but do not provide goal-directedness and stories generated can still be perceived as lacking in coherence in the sense that they meander without direction.

In this paper we consider the challenge of coherent and controllable text generation for neural language model based story generation. We hypothesize that neural language models, while powerful text-completion systems, are not natively well-suited for coherent story generation because a neural network trained with a cross-entropy loss function is unlikely to model the unfolding context of a story the same way as a human reader. Studies of human reader comprehension (Zwaan and Radvansky, 1998) show that readers comprehend stories by tracking the relations between entities and events in ways that can be expressed as a graph. The perceived coherence of a story is a function of the connectedness of this graph (Graesser et al., 1994). Ensuring the causality between sentences can significantly improve the coherence of stories (Peng et al., 2021).

Inspired by cognitive science, we aim to augment neural language models with a **reader model** in which a story generation system infers a graph of concepts, entities, and relations that a reader is likely to believe about the story world as they read an incrementally generated story. The reader model enables the story generation algorithm to explicitly reason about the entities and relations and generate story continuations that use those entities to move the story forward; a reader can track how entities and relations change over time and thus perceive stories as more coherent. We use large language models to produce the continuation text of the story generation. However instead of providing the previous story as context, our algorithm selects one or more entity from the world model and uses template filling to generates candidate continuations. This is a deviation from the typical way in which large language models are used to generate text continuations from a context prompt and possibly some additional external biases.

The reader model doesn't guarantee controlled text generation or goal-directedness but provides a means for directing the generation process. In addition to a starting context prompt, we require a goal to be given in the form of what the graphical reader's model of the story world should be at the end of the story. In that way, the goal provides a rough outline of the entities and relations that need to be present in the story but without providing particulars about everything that must be in the story or the ordering in which they must occur.

Our contributions are as twofold: (1) we propose an automated story generation model with Reader Models (**StoRM**) which maintain coherence and controllability of generated stories at the same time; and (2) we conduct a thorough experimental study against strong baselines that shows that StoRM produces significantly more coherent and goal-directed story.

2 Related Work and Background

Instead of symbolic story generation systems (Meehan, 1976; Lebowitz, 1987; Cavazza et al., 2003; Porteous and Cavazza, 2009; Riedl and Young, 2010; Ware and Young, 2010; Pérez and Sharples, 2001; Peinado and Gervás, 2005; Turner, 2014; Ware and Siler, 2021), we focus on related work using neural networks—recurrent and transformer-based—to produce stories (Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2018; Clark et al., 2018). These language model is trained to approximate the distribution $P_\theta(tok_n|tok_{<n})$. They produce story by sampling a sequence of tokens from the distribution.

Among prior works, there are a couple of story generation models that are highly related to our proposed framework, in terms of the following two dimensions: the generation controllability and the usage of commonsense knowledge. The controllability in story generation focuses on how to enable the generation process to adhere to the user's inputs. Fan et al. (2018) proposes a hierarchical story generation framework that divides the generation process into two levels of hierarchy: generating a writing prompt (premise) and then transforming it into a passage of text conditioned on the prompt. Similarly, Plan-And-Write (Yao et al., 2019) conducts generation in two steps: planning a story outline based on a title (topic), then generating a story based on the storyline. Plot Machines (Rashkin et al., 2020) accepts as an input an un-ordered outline of concepts and conditions a language model.

Commonsense knowledge plays an important role in story generation. The most popular way of utilizing it is to train neural language models (e.g. GPT-2 (Radford et al., 2019)) on commonsense knowledge bases such as ConceptNet (Speer and Havasi, 2013) and ATOMIC (Sap et al., 2019; Hwang et al., 2021) which contains detailed information regarding well-known facts or causal relationships. Thus the resulting language model, named COMET (Bosselut et al., 2019; Hwang et al., 2021), becomes capable of inferring new commonsense knowledge on novel phrases. Ammanabrolu et al. (2021b) proposes Causal, Commonsense Plot Ordering (C2PO) framework which takes advantage of COMET to infer predecessor and successor events and then bi-directionally search from pre-specified start event to end event, however, C2PO generates plots made up of highly constrained, templated text; Peng et al. (2021) leverages COMET to infer the character intentions and effects of actions so as to guide the generation process, but they did not consider controllability. There are also other approaches that directly incorporate commonsense knowledge graphs into the encoding process (Mihaylov and Frank, 2018; Guan et al., 2019). Their works paid more attention on improving coherence with the help of common-sense knowledge but did not take controllability into consideration.

3 Story Generation with Reader Models

In this section we introduce a framework—*Story generation with Reader Models* (StoRM)—for generating stories that models that the reader will believe about the fictional story world. We hypothesize that the incorporation of a *reader model* into the story generation process will increase story coherence. We define *story coherence* as the extent to which readers can identify connections between different events and entities in a story. In this work, the reader model is represented as a *knowledge graph*, a set of triples of the form $\langle subject, relation, object \rangle$ (see left side of Figure 2). By making the beliefs about what the reader likely knows explicit, we provide mechanisms for selecting which entities to include in the continuation of the story.

Because the StoRM framework maintains a knowledge graph that approximates the reader's beliefs about the story world, we are able to compare the reader model to a desired world state, also described as a knowledge graph. The StoRM framework is thus *controllable*—a user can provide a *story goal* in the form of a knowledge graph—*goal story world*—that describes the story world at the conclusion of the story. This allows the system to make informed decisions about which possible story continuations are likely to achieve the desired goal state by inferring how each possible story continuation changes the reader model to be closer to the desired goal story world

Our framework starts with a prompt and a description of the story outcome (See Figure 2). The prompt is transformed into a knowledge graph by extracting entities. The entities are expanded using two commonsense techniques: (1) ConceptNet (Speer and Havasi, 2013), a crowdsourced knowledge base of concepts and relations, and (2) COMET$^{20}_{20}$ (Hwang et al., 2021), a neural network that generates commonsense inferences.

The generation technique selects different entities and uses templates to generate possible story continuations. By targeting different entities and using template infilling, we reduce neural network hallucination of new entities and create a diverse set of story continuations. Each potential continuation is scored based on how it changes the knowledge graph relative to the goal, which is also transformed into a knowledge graph. The selected continuation starts the next iteration of the generation process.

3.1 Knowledge Graph Acquisition

With the automatic generation of the story, some important information could be forgotten. The knowledge graph is an explicit and *persistent* memory of entities mentioned or inferred from the story text generated so far. Knowledge Graphs represent information in the form of triples, consisting of a subject entity, relation and object entity. For example, "*Jenny lived in Florida*" is represented as $\langle jenny, live, florida \rangle$ The entities represent the nodes of the graph and their relations act as edges.

To acquire the knowledge graph, we firstly trained a Semantic Role Labeling (SRL) model (Gildea and Jurafsky, 2002) on VerbAtlas (Di Fabio et al., 2019), a hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from WordNet (Fellbaum, 1998) into semantically-coherent frames (see Appendix A.1). This SRL model provides the automatic identification and labeling of argument structures of stories. For example, it extracts `'verbatlas': 'EXIST_LIVE', 'args_words': {'Theme': 'Jenny', 'Attribute': 'Florida'}` from "*Jenny lived in Florida*". Verbs in the story will be

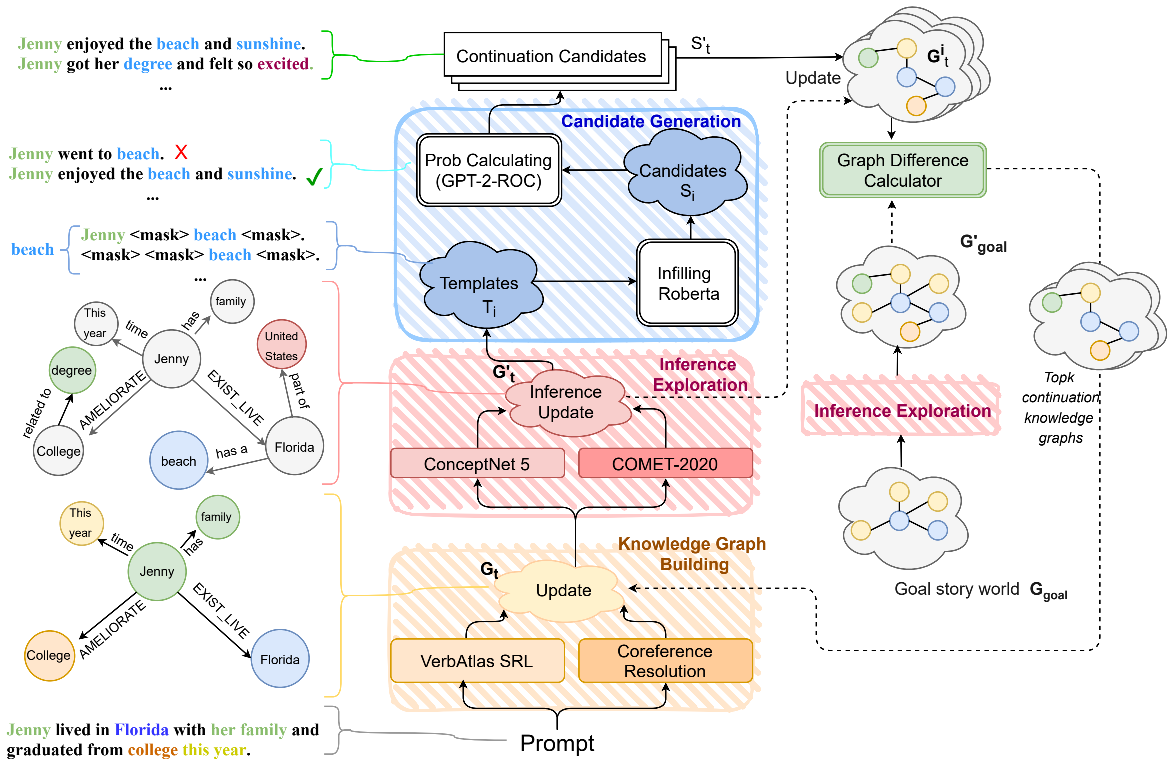


Figure 2: The overall procedure of Reader Model.

represented as the VerbAtlas frame. For example, "`live`" is represented as "`EXIST_LIVE`".

StoRM then converts the output of VerbAtlas SRL model into knowledge graph triples. Entities represent the theme and attribute and VerbAtlas frames act as edges. Multiple character names, objects names and pronouns make the knowledge graph representation hard to interpret. Hence, we adopt a end-to-end Coreference Resolution model (Lee et al., 2017) to find all expressions that refer to the same entity in a story to minimize the entity numbers.

StoRM starts with two knowledge graphs. The first, $\mathbf{G}_1$, is the converted prompt (first sentence). The second $\mathbf{G}_{\text{goal}}$ is the converted goal description. $\mathbf{G}_{\text{goal}}$ can also be generated from a target story, in which case each sentence is converted to nodes and relations and incrementally added to the graph. With the generation of the continuation candidates, we will update the knowledge graph $\mathbf{G}_t$ with new continuations to get new knowledge graph $\mathbf{G}_{t+1}$, where t is the index of the sentence in the story.

3.2 Graph Expansion

Human readers use commonsense knowledge to infer the presence of entities and concepts not explicitly mentioned in story text. For example, Florida has beaches and eating dinner implies dishes. In accordance, we use common-sense inferences of entities to expand the knowledge graph to provide entities for characters to interact with and thus drive the story forward. Because the presence of entities and concepts are inferred from prior events, the reader should be able to track the connections between entities and events, thus supporting perceived story coherence.

We use two inference techniques. We first consider ConceptNet5 (Speer and Havasi, 2013), a multilingual knowledge base, representing words and phrases that people use and the common-sense relations between them. We expand each entity except characters nodes with the inference extracted from ConceptNet5 to get inference entity set $E_{1,t}$. Second, we repeat the process with COMET$^{20}_{20}$ (Hwang et al., 2021), which is a transformer-based generative model trained on the ATOMIC$^{20}_{20}$ commonsense dataset (Hwang et al., 2021) to infer relations about sentences broadly divided into four categories: (1) Causes of a person's actions (preconditions of the event), (2) Attributes of a person, (3) Effects of actions on a person (postconditions of the event), and (4) Effects of actions on others (postconditions of the event). We expand the knowledge graph with these inference on the current story to get inference set $E_{2,t}$. Finally, we obtain the updated knowledge graph $\mathbf{G}'_t = \{e_{1,\mathbf{G}'_t}, ..., e_{l,\mathbf{G}'_t}\}$ and inference entities set $E_t = E_{1,t} \cup E_{2,t} = \{\hat{e}_{1,\mathbf{G}'_t}, ..., \hat{e}_{l,\mathbf{G}'_t}\} \subset \mathbf{G}'_t$, where $\hat{e}$ is an inferred entity and l is the number of the inference entities.

3.3 Continuation Candidate Generation

Given each entity in the knowledge graph including inference entities (§3.2) and current story history, we first generate a set of continuation candidates. We consider the conditional sentence generation as a infilling task (Taylor, 1953).

Templates. A set of templates T_i are generated on each entity $e_{i,\mathbf{G}'_t}$ (See Appendix A.2). For example, one of the templates generated on `beach` are `[subject] <mask> <mask> beach <mask>`. The `[subject]` of the sentence is (1) the same subject with the previous sentence, (2) no fixed (`<mask>`), or (3) any characters in previous story history. The number of `<mask>` before and after inference entity ranges from 1 to 10.

Text Infilling. We fine-tune RoBERTa (Liu et al., 2019) on ROCStories (Mostafazadeh et al., 2016)—a set of five-sentence stories involving commonsense scenarios—to infill the mask tokens. Details of training are shown in Appendix A.3. All the templates T_i are filled by fine-tuned RoBERTa and we obtain a number of continuation candidates $S_i = \{s_{i,1}, ...s_{i,m}\}$ for each entity $e_{i,\mathbf{G}'_t}$ where m is the number of templates of each entity.

Filtering. We fine-tune GPT-2 (Radford et al., 2019) on ROCStories (henceforth called GPT-2-ROC) and filter the continuation candidates by calculating their conditional probability $\mathbb{P}_s$ with it:

$$\mathbb{P}_s = \prod_{j=1}^{n} \mathbb{P}(\mathrm{X}_j | \mathrm{X}_1, ..., \mathrm{X}_{j-1}) \quad (1)$$

where n is the length of the sentence s and X_j is the jth token in sentence s. We only keep one sentence $s_i \in S_i$ with the highest probability for each entity $e_{i,\mathbf{G}'_t}$ and obtain the continuation candidates $S'_t = \{s_1, ..., s_l\}$.

3.4 Graph Difference

We seek to achieve controllability of the continuation candidates S' by calculating the *graph difference* between the candidate knowledge graph $\mathbf{G}_t^{s_i}$ and the goal knowledge graph $\mathbf{G}_{\text{goal}}$. The candidate knowledge graph $\mathbf{G}_t^{s_i}$ is obtained by updating knowledge graph $\mathbf{G}_t$ with continuation candidate s_i. We calculate the knowledge graph difference score:

$$R(s_i) = (1-\alpha) \times r_1(\mathbf{G}_t^{s_i}, \mathbf{G}_{\text{goal}}) + \alpha \times r_2(E_t^i, \mathbf{G}'_{\text{goal}}) \quad (2)$$

where r_1 is a story entity overlapping score and r_2 is a inference overlapping score between inference set E_t^i (§3.2) and updated goal knowledge graph $\mathbf{G}'_{\text{goal}}$ with inferences. α is a hyper-parameter to control the inference's contribution on calculating overlapping rate.

Story entity overlapping score (r_1) calculates the overlapping rate between the candidate knowledge graph $\mathbf{G}_t^{s_i}$ and the full knowledge graph $\mathbf{G}_{\text{goal}}$ without considering inference nodes. We define a match as same entities (nodes) and their corresponding edges (relations) between two knowledge graph. Then calculate the story entity overlapping rate by

$$r_1(\mathbf{G}_t^{s_i}, \mathbf{G}_{\text{goal}}) = \frac{\sum_j \sum_k \mathbb{I}(e_{j,\mathbf{G}_t^{s_i}} = e_{k,\mathbf{G}_{\text{goal}}})}{\text{size of } \mathbf{G}_{\text{goal}}} \quad (3)$$

where $\mathbb{I}(e_{j,\mathbf{G}_t^{s_i}} = e_{j,\mathbf{G}_{\text{goal}}}) = 1$ when there is a match between entity $e_{j,\mathbf{G}_t^{s_i}} \in \mathbf{G}_t^{s_i}$ and entity $e_{k,\mathbf{G}_{\text{goal}}} \in \mathbf{G}_{\text{goal}}$, otherwise 0.

After updating knowledge graph G_t with s_i, we expand updated knowledge graph $G_t^{s_i}$ with inference nodes $E_t^{s_i}$, which we repeat the process in Section 3.2 on $G_t^{s_i}$. We calculate the overlapping rate between $E_t^{s_i}$ and goal knowledge graph with inferences nodes, $\mathbf{G}'_{\text{goal}}$, as *inference overlapping score* r_2

$$r_2(E_t^{s_i}, \mathbf{G}'_{\text{goal}}) = \frac{\sum_j \sum_k \mathbb{I}(\hat{e}_{j,E_t^{s_i}} = e_{k,\mathbf{G}'_{\text{goal}}})}{\text{size of } \mathbf{G}'_{\text{goal}}} \quad (4)$$

where $\hat{e}_{j,E_t^{s_i}} \in E_t^{s_i}$ and $e_{k,\mathbf{G}'_{\text{goal}}} \in \mathbf{G}'_{\text{goal}}$.

We obtain the *topk* continuation knowledge graphs with the *topk* highest graph difference score. The *topk* continuation knowledge graphs will be used to produce continuations further. We always keep a total of k knowledge graphs (states) when generating stories for each index of the sentence in the story. Thus the full generation process is implemented as a form of beam search through reader model space.

4 Experiments

We evaluated our system with four experiments. The first experiment accesses whether the knowl-

Goal story world (without order):
`<Doug, meet, women>`, `<Doug, ask, girl>`
`<Doug, marry, women>`, `<girl, is, nice>`

StoRM: `Doug has always wanted to be married.` He decides to go on **a first date** with his girlfriend. Doug happily **spends time** with his girlfriend. Doug **loves** every minute of her. He is happy that he has finally found a perfect wife. He is happy that he **married a beautiful girl**. He is glad that he met his girlfriend after his first date.
Outline: Doug has always wanted to be married # too shy # ask out a nice girl # first date # marry
GPT-outline: `Doug has always wanted to be married.` Doug decided to **date** Amy more than anything else. He was *shy and* ***shy***. He never got to meet **a nice girl**. She decided he must be more serious. She went on a date with him and two of them hit it off. Later that night harry **proposed** to Amy and she said yes.
Yao et al. (2019): `Doug has always wanted to be married.` Doug was natural grown in the kid. He **met his girl** someday. Doug asked to watch her. They started **dating**. They both will honestly meet new friends together.

Table 1: Story examples generated by StoRM, GPT-outline, and Yao et al. (2019). StoRM successfully finds the correct order to follow a single topic (**bolded**)—"meet a girl and marry", and shows a good plot coherence. The story generated by GPT-outline succeeds to follow the outline(**bolded**) but suffers in repetition (in *italic*) and lacks coherence (underlined) during generation. Yao et al. (2019) suffers in plot coherence (underlined).

edge graph acquisition technique captures the information that natural language story conveys. The second experiment is an ablation study that assesses how each component contributes to the story generation process in Figure 2. The third and fourth experiments compare StoRM to two neural language model story generators on the the dimensions of coherence and controllability. Story examples can be found in Table 1.

Datasets. We conduct the experiments on the ROCStories corpus (Mostafazadeh et al., 2016). It contains $98,159$ five-sentence stories involving common-sense scenarios. We additionally extract outlines for ROCStories using the RAKE algorithm (Rose et al., 2010) for use controlling the baseline models.

Baselines. For fair comparison to baselines, we require story generation systems that operate on un-ordered outlines that abstractly indicate events that should appear in the story, and/or goal states as inputs.[1] We selected two strong baselines. The first is GPT-2-small (Radford et al., 2019) fine-tuned on ROCStories (Mostafazadeh et al., 2016) with outlines. GPT-2 is fed into outlines with the format of `{topic_1 # topic_2 #...#}`, and then minimizes the cross entropy loss between network output logits and golden truth story from which the topics were extracted. Training details can be found in Appendix A.4.

The second baseline is the system by Yao et al. (2019), which trained RNN based conditional generation models on ROCStories to generate story on outline. Their Plan-and-Write system is capable of generating its own outline of topics. However, for fair comparison, we provide the system with the outline of topics extracted from our dataset. This controls for goal-seeking behavior because the outlines are extracted from the same stories that are used to generate goal knowledge graphs for StoRM. Training details can be found in Appendix A.5.

4.1 Knowledge Graph Acquisition Evaluation

We assess whether knowledge graph can acquire the story world state accurately and comprehensively. We randomly select 125 sentences from ROCStories and convert them into knowledge graph triples. We recruited 30 participants on a crowdsourcing platform. Each participant read a randomly selected subset of knowledge graph triples (20 sentences per participant). They were asked to validate each graph triples given the sentence and then write down the missing information. For example, they need to check whether $\langle jenny, LIKE, beach\rangle$ given "*Jenny likes beach and sunshine*" is correct and write down the missing concept, "`sunshine`". At least 3 crowd workers validate each triple and we take the majority vote as the result. The detail of this study is shown in Appendix B.1 and B.2.

Table 2 shows the accuracy (precision) and sensitivity (recall) of the extracted knowledge graph

[1]Two potential baselines were considered but not pursued. The system by Tambwekar et al. (2019) is goal-driven but does not produce natural language without manual intervention. The system by Rashkin et al. (2020) accepts unordered outline terms but the results of the original paper could not be reproduced at the time of writing.

Precision %	**Recall %**	**# of triplets**
81.96‡	72.89‡	255

Table 2: Results of the human study, evaluating knowledge graph triplets. ‡ indicates κ > 0.4 or moderate agreement.

triples. We treat the majority vote from human participants as the ground-truth. *Precision* is the fraction of extracted triples that are correct rated by human participants. *Recall* is the fraction of the triples that are successfully extracted from stories. Precision, 81.96%, shows that the knowledge graph can represent the information in sentences accurately. Recall, 72.89%, proved that the knowledge graph can represent most of the information in sentences. Both of these two metrics have moderate agreement. This indicates that the knowledge graph extracted from sentences matches reader expectations and can be used as story world state upon which to base further story generation.

4.2 Ablation Study

We perform ablation studies to validate the contributions of different components in Figure 2. We randomly choose 50 stories from ROCStories (Mostafazadeh et al., 2016) and build a goal story world state (§3.1) to guide the story generation process. StoRM generates story continuations until knowledge graph difference score $R(s)$ reaches 0.8. We measure the following two metrics:

- *Average story length* (Avg. len): Calculate the average story length which is required to reach $R(s) = 0.8$ (§3.4). Smaller average story length stands for faster, and thus more direct, goal achievement. We stop generation when story length reaches 10.
- *Sentence transformer cosine similarity* (S-F) (Reimers and Gurevych, 2019): Evaluate the semantic similarity between generation and golden truth story by calculating embedding cosine similarity. Higher S-F score indicates higher similarity between generation and gold story.

Table 3 shows the result of the ablation study. Removing ConceptNet and COMET$^{20}_{20}$, which infer nodes in the reader model, significantly increases average story length and reduces similarity score. It indicates with the help of these two inference technique, StoRM is faster to achieve a better goal. Ablating the *candidate generation* module (Blue box in Figure 2) by replacing our technique with BART likewise diminishes similarity between gold story and generated story significantly.

Model		**Avg. len** ↓	**S-F %** ↑
StoRM Full	$\alpha = 0.5$	**7.96** ± 0.73	**77.54** ± 4.27
	$\alpha = 1.0$	8.96 ± 0.65^b	74.67 ± 3.25^a
	$\alpha = 0.25$	9.08 ± 0.65^a	70.12 ± 5.28^b
COMET$^{20}_{20}$ only		8.96 ± 0.74^a	70.10 ± 4.06^b
ConceptNet only		8.85 ± 0.69^b	73.71 ± 4.12^a
BART Candidates		8.08 ± 0.69	73.14 ± 3.90^a

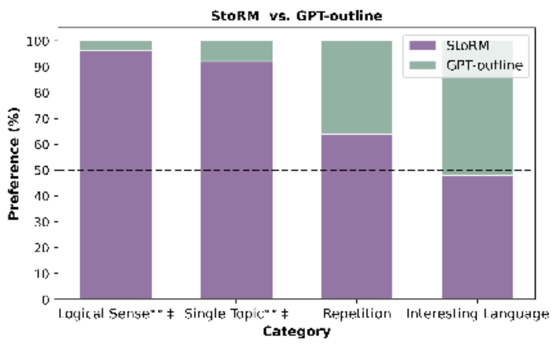
Table 3: Results of the ablation study. Smaller average length and higher S-F score indicate that the system is faster to achieve a goal with a more similar generated story with golden truth. `StoRM` is the model shown in Figure 2 and $\alpha = 0.5$. `ConceptNet only` and `COMET`$^{20}_{20}$ `only` indicate StoRM model only using ConceptNet or COMET$^{20}_{20}$ for inferring nodes. `BART` shows the result of replacing the whole candidate generation module with BART. α is tuning the inference contribution when calculating graph difference. ^{a} and ^{b} indicate StoRM($\alpha = 0.5$) results are significant better at $p < 0.05$ and $p < 0.01$ using the Mann-Whitney U test.

We experiment with three values of α in our StoRM framework. The best performing model has $\alpha = 0.5$, balancing between inference-node-guided and goal-node-guided story generation, where larger α indicates more inference-node-driven.

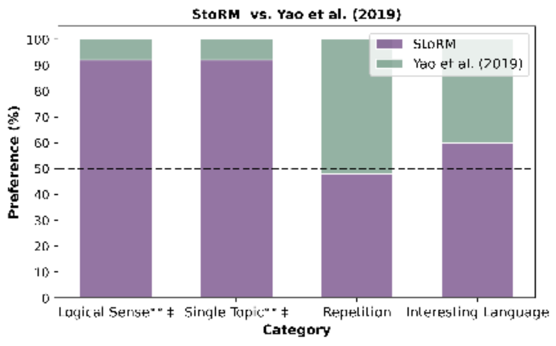
4.3 Story Coherence Evaluation

Having established that knowledge graph is able to represent the story, we seem to understand whether StoRM improves the coherence and quality of the generated story. In this paper, we evaluate coherence using human participant evaluation, asking a set of questions that includes dimensions such a logical coherence, loyalty to plot, and enjoyability. Variations of these questions have been used to evaluate other story generation systems (Purdy et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020, 2021a; Castricato et al., 2021; Peng et al., 2021). We focus on dimensions involving overall perceptions of narrative coherence:

- *Logical Sense*: Attempt to get at narrative coherence without using the term "coherence" which can sometimes also be confused with grammaticality.
- *Follows A Single Topic*: Query about coherence since incoherent plots can look like an intertwining of several unrelated topics.



(a) StoRM vs. GPT-outline



(b) StoRM vs. Yao et al.(2019)

Figure 3: Human evaluation results comparing StoRM with two baselines, $*$ indicates $p < 0.05$, $**$ indicates $p < 0.01$, † indicates $\kappa > 0.2$ or fair agreement. ‡ indicates $\kappa > 0.4$ or moderate agreement.

- *Repetition*: Measure which story has less repetitive words.
- *Interesting Language*: Focus on wording.

We recruited 40 participants and each participant reads a randomly selected subset of 10 story pairs, comprised of one story from StoRM and one from baselines—GPT-outline or Yao et al.. For the above four questions, participants answered which story best met the criteria. Our study was approved by our Institutional Review Board, and we payed participants the equivalent of $15/hr. Details can be found in Appendix B.3. To generate the stories, we randomly selected 25 stories from the ROCStories corpus. We build a goal knowledge graph story world to guide StoRM and use outlines extracted from the same story to seed baselines. The baselines have an advantage over StoRM; they use outlines that are ordered which should correlate with story coherence whereas our system must intuit the order of events that achieves the goal.

The results are shown in Figure 3, which indicate that StoRM performs significantly better than baselines on the the dimensions of "Logical Sense" and "Single Topic". These are the primary dimensions

Model	KG-o %	B-1	B-2	S-M	S-F %
StoRM	**53.21**	**.337**	**0.166**	**.076**	**74.54**
GPT-outline	47.24^a	.290	0.143	.074	74.04
Yao et al.	31.67^b	.248^b	0.051^b	.049^b	66.66^b

Table 4: Evaluation on goal-guided story generation. Columns show average score of each model's generated summaries according to various metrics. ^{a} and ^{b} indicate StoRM results are significant better at $p < 0.05$ and $p < 0.01$ using the Mann-Whitney U test.

that ask about story coherence. We conclude that our system improves the perception of narrative coherence of generated narratives and stays more on topic, while retaining comparably interesting language and avoidance of repetition (neither of which are statistically significantly different from the baseline). Since GPT-outline and Yao et al.(2019) are both guided by outlines, which are used as prompt to seed the neural language model, StoRM overcomes a disadvantage in that it must figure out how to reasonably order concept nodes in the goal story world and compose a coherent story.

4.4 Controllability Evaluation

We assess whether StoRM is able to achieve the given goal, as measured by the coverage of all the given concepts in goal story state. We evaluate controllability of StoRM and baselines with four metrics.

- *Knowledge graph overlapping rate* (KG-o): Generated stories are transformed into knowledge graph (§3.1). Calculate the overlapping rate of knowledge graph nodes between generated story and the golden truth story.
- *Sentence transformer cosine similarity* (S-F) (Reimers and Gurevych, 2019): Evaluate the semantic similarity between generation and golden truth story by calculating embedding cosine similarity.
- *Sentence mover's similarity* (Clark et al., 2019): Evaluate stories in a continuous space using word and sentence embeddings.
- *BLEU score* (B-1 and B-2) (Papineni et al., 2002): 1-gram and 2-gram BLEU scores are reported.

Table 4 shows the result of all the systems. We first convert all the generated stories of StoRM and baselines to knowledge graph and then calculate their knowledge graph overlapping rate (KG-o) with goal story world. Higher KG-o indicates better controllability. StoRM performs statistical significantly better than baselines in this dimension. Similarity between generated story and golden truth is also considered a way to evaluate controllability. StoRM outperforms Yao et al.(2019) in "sentence transformer similarity", "sentence mover's similarity" and "BLEU" but comparable to GPT-outline. Without seeding outlines to the language model like GPT-outline and Yao et al.(2019), StoRM is able to cover most of the concepts in the goal story state.

5 Conclusions

Neural language models are widely used to produce text, including stories. However, they struggle with maintaining *story coherence*—the logical progression of events—and goal-directedness. Our framework—*Story Generation with Reader Models* (StoRM)— augments neural language models with a reader model. This reader model—in this case an explicit knowledge graph—approximates the reader's beliefs about the story world. StoRM increases the story coherence by expanding the reader model with commonsense technique and producing continuations by selecting entities in this reader model. In order to achieve goal-directedness, StoRM allows the system to make informed decisions about which possible story continuations are likely to achieve the desired goal state by inferring how each possible story continuation changes the reader model to be closer to the desired goal story world.

A thorough experimental study shows that StoRM produces significantly more coherent and goal-directed stories than two strong baselines. The goal-directness results are significant because the StoRM framework takes a goal as a knowledge graph, which can be thought of as an unordered outline of concepts that should appear in the story; our system does well to find an appropriate sequencing of events.

References

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021a. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of AAAI*, volume 35.

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021b. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5859–5867.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of AAAI*, volume 34.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Louis Castricato, Spencer Frazier, Jonathan Balloch, and Mark O. Riedl. 2021. Tell me a story like i'm five: Story generation via question answering. In *Proceedings of the 3rd Workshop on Narrative Understanding*.

Marc Cavazza, Olivier Martin, Fred Charles, Steven J Mead, and Xavier Marichal. 2003. Interacting with virtual agents in mixed reality interactive storytelling. In *International Workshop on Intelligent Virtual Agents*, pages 231–235. Springer.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of NAACL-HTL*, pages 2250–2260.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. Verbatlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 627–637.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database cambridge. *MA: MIT Press*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Arthur C Graesser, Murray Singer, and Tom Trabasso. 1994. Constructing inferences during narrative text comprehension. *Psychological review*, 101(3):371.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6473–6480.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. 2017. Deeptingle. *arXiv preprint arXiv:1705.03557*.

Michael Lebowitz. 1987. Planning stories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 234–242.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lara Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of AAAI*, volume 32.

James Richard Meehan. 1976. *The Metanovel: Writing Stories by Computer.* Yale University.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Federico Peinado and Pablo Gervás. 2005. Creativity issues in plot generation. In *Workshop on Computational Creativity, Working Notes, 19th International Joint Conference on AI*, pages 45–52.

Xiangyu Peng, Siyan Li, Sarah Wiegreffe, and Mark Riedl. 2021. Inferring the reader: Guiding automated story generation with commonsense reasoning. *arXiv preprint arXiv:2105.01311*.

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

Julie Porteous and Marc Cavazza. 2009. Controlling narrative generation with planning trajectories: the role of constraints. In *Joint International Conference on Interactive Digital Storytelling*, pages 234–245. Springer.

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. 2018. Predicting generated story quality with quantitative measures. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *Proceedings of AAAI*, volume 30.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*, pages 161–176. Springer.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reinforcement learning. In *Proceedings of the 28th IJCAI*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Scott R Turner. 2014. *The creative process: A computer model of storytelling and creativity*. Psychology Press.

Stephen Ware and Cory Siler. 2021. Sabre: A narrative planner supporting intention and deep theory of mind. In *Proceedings of the 17th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Stephen G Ware and R Michael Young. 2010. Modeling narrative conflict to generate interesting stories. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Rolf A Zwaan and Gabriel A Radvansky. 1998. Situation models in language comprehension and memory. *Psychological bulletin*, 123(2):162.

A Implementation Details

A.1 Semantic Role Labeling Using VerbAtlas

For English semantic role labeling (SRL), we use a fine-tuned transformer model proposed by (Shi and Lin, 2019) which is the current state-of-the-art for English SRL. It is a BERT (Devlin et al., 2019) model with a linear classification layer trained on the Ontonotes 5.0 dataset to predict PropBank SRL. We use an open-source implementation [2], which is based on the official AllenNLP BERT-SRL model [3]. Trained with the following hyperparameters:

- Batch size: 32
- Dropout for the input embeddings: 0.1
- Learning rate: $5e^{-5}$
- Optimizer: Adam
- Total Epochs: 15

Then, we use the mappings from Propbank frames to VerbAtlas (Di Fabio et al., 2019) classes to return the correct corresponding VerbAtlas classes instead of Propbank's (Palmer et al., 2005). The direct mapping is possible because, for every VerbAtlas class, there is only one ProbBank frame, which allows us to utilize the rich content provided by VerbAtlas while using the same model initially trained to predict ProbBank.

A.2 Continuation Candidate Generation Details

For each entity $e_{i,\mathbf{G}'_t}$, we generate $5 \times 10 \times (c+1)$ templates, where c is the number of subjects in the story history. Rules of making the templates are as follows,

- The first token in the template has two choices: (1) Previous subject, i.e. *"Jenny"*; (2) $\langle mask \rangle$.
- Between the first token and the entity $e_{i,\mathbf{G}'_t}$, we put $0 \sim 4$ $\langle mask \rangle$.
- After the entity $e_{i,\mathbf{G}'_t}$, we put $0 \sim 10$ $\langle mask \rangle$ tokens

Examples are as follows when $e_{i,\mathbf{G}'_t}$ =`"beach"`,

- `Jenny <mask> beach <mask>`.
- `Jenny <mask> <mask> beach`.
- `Jenny <mask> <mask> beach <mask> <mask> <mask>`.

A.3 RoBERTa Fine-tuning

We fine-tune RoBERTa (Liu et al., 2019) on ROCStories (Mostafazadeh et al., 2016) to infill the mask tokens in the given text template. We preprocess the ROCStories by masking 15% of all the tokens randomly, concatenating all texts together, and splitting them into chunks of the same length (equal to 128). Each chunk is then used as one training sample.

During fine-tuning, we use the AdamW optimizer (Loshchilov and Hutter, 2017) to train the RoBERTa for 3 epochs with batch size = 8. Other optimizer-related hyperparameters are attached as follows.

- learning rate: $\gamma = 2 \times 10^{-5}$
- betas: $\beta_1 = 0.9$, $\beta_2 = 0.999$
- epsilon: $\epsilon = 10^{-8}$
- weight decay: $\lambda = 0.01$

A.4 Baselines—GPT-outline

We use the small version of GPT-2 (Radford et al., 2019) with 124M parameters as the base for all fine-tuned models. We converted the data as the following format: `topic_1 # topic_2 # ...# topic_n # Stories`. For example, `Florida # beach # Jenny lived in Florida. She loves beach.` When fine-tuning GPT-2 on ROCStories and the commonsense knowledge resources (done separately), we train with a batch-size of 16, a learning rate of 0.00005, and using the Adam optimizer with gradient clipping at a max norm of 1. GPT-2 is fed into outlines with the format of `{topic_1 # topic_2 #...#}`, and then minimizes the cross entropy loss between network output logits and gold truth story from which the topics were extracted. All models were trained on single GeForce RTX 2080 GPUs in Pytorch using the Huggingface Transformers library.[4]

A.5 Baselines—Yao et al.(2019)

We replicate the Plan and Write model using the code published on the paper's public repository [5] to train our baseline (Yao et al., 2019). We first filtered the training dataset, removing our test story outlines to prevent data leakage. Then we train the model with the hyperparameters specified in their documentation. Below are those hyperparameter values:

- Dropout for input embedding: 0.4

[2]https://github.com/Riccorl/transformer-srl
[3]https://demo.allennlp.org/semantic-role-labeling

[4]https://huggingface.co/transformers/
[5]https://bitbucket.org/VioletPeng/language-model/

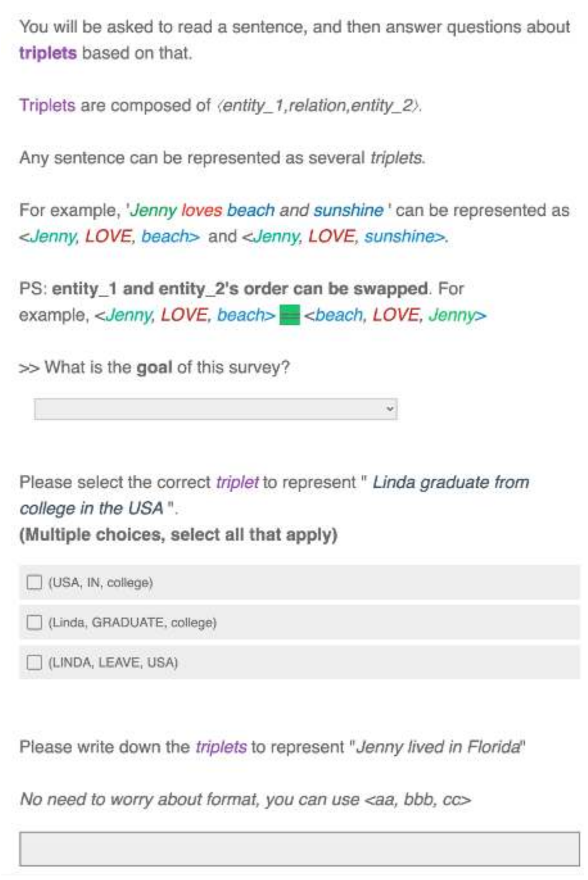
- Dropout for the RNN layers: 0.25
- Random Seed: 141
- Total Epochs: 500
- Word Embedding size: 1000
- Hidden units size per RNN layer: 1000
- Batch size: 80
- Learning rate: 30
- Optimizer: SGD
- Sequence Length: 70

However, the best checkpoint was around 100 epochs as the validation loss stopped decreasing at 2.85.

B Human Evaluation Details

B.1 Task Instruction

We ask participants a set of questions to make sure they understand our task. The details can be found in Figure 4.


You will be asked to read a sentence, and then answer questions about **triplets** based on that.

Triplets are composed of ⟨*entity_1,relation,entity_2*⟩.

Any sentence can be represented as several *triplets*.

For example, '*Jenny loves beach and sunshine*' can be represented as <*Jenny, LOVE, beach*> and <*Jenny, LOVE, sunshine*>.

PS: **entity_1 and entity_2's order can be swapped**. For example, <*Jenny, LOVE, beach*> 🟩 <*beach, LOVE, Jenny*>

>> What is the **goal** of this survey?

Please select the correct *triplet* to represent " *Linda graduate from college in the USA* ".
(Multiple choices, select all that apply)

- ☐ (USA, IN, college)
- ☐ (Linda, GRADUATE, college)
- ☐ (LINDA, LEAVE, USA)

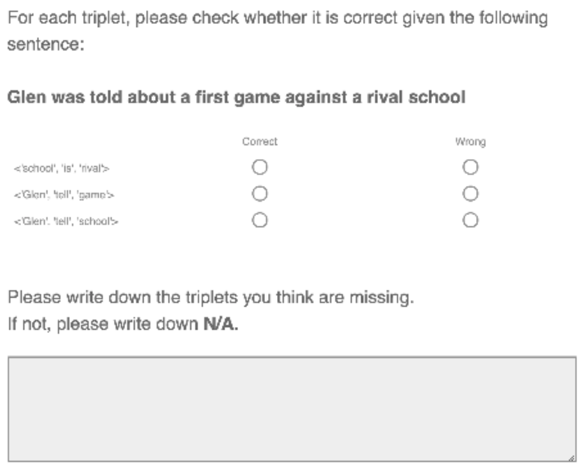
Please write down the *triplets* to represent "*Jenny lived in Florida*"

No need to worry about format, you can use <aa, bbb, cc>

Figure 4: Screenshot of the human study instruction.

B.2 Knowledge Graph Acquisition Evaluation Set-up

We assess whether knowledge graph can acquire the story world state accurately and comprehensively. We randomly select 125 sentences from ROCStories and convert them into knowledge graph triplets. We recruited 30 participants on a crowdsourcing platform. Each participant read a randomly selected subset of knowledge graph triplets (20 sentences per participant). They were asked to validate each graph triplets given the sentence and then write down the missing information. An example is shown in Figure 5.


For each triplet, please check whether it is correct given the following sentence:

Glen was told about a first game against a rival school

	Correct	Wrong
<'school', 'is', 'rival'>	○	○
<'Glen', 'tell', 'game'>	○	○
<'Glen', 'tell', 'school'>	○	○

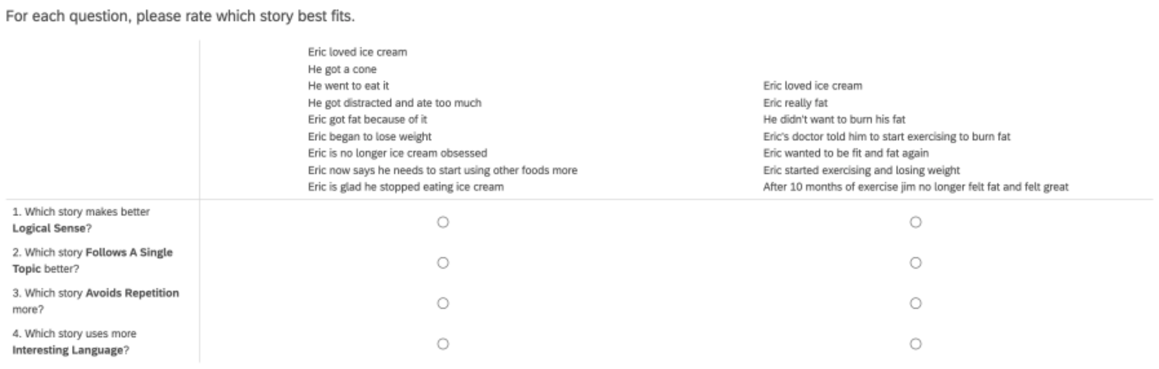
Please write down the triplets you think are missing.
If not, please write down **N/A.**

Figure 5: Screenshot of Knowledge Graph Acquisition evaluation.

B.3 Story Coherence Evaluation Set-up

We evaluate coherence using human participant evaluation, asking a set of questions that includes dimensions such a logical coherence, loyalty to plot, and enjoyability. Example of human study is shown in Figure 6. We ask the following four questions:

- Which story makes better logical sense?
- Which story follows a single topic?
- Which story avoids repetition?
- Which story uses more interesting language?



For each question, please rate which story best fits.

	Eric loved ice cream He got a cone He went to eat it He got distracted and ate too much Eric got fat because of it Eric began to lose weight Eric is no longer ice cream obsessed Eric now says he needs to start using other foods more Eric is glad he stopped eating ice cream	Eric loved ice cream Eric really fat He didn't want to burn his fat Eric's doctor told him to start exercising to burn fat Eric wanted to be fit and fat again Eric started exercising and losing weight After 10 months of exercise jim no longer felt fat and felt great
1. Which story makes better **Logical Sense**?	○	○
2. Which story **Follows A Single Topic** better?	○	○
3. Which story **Avoids Repetition** more?	○	○
4. Which story uses more **Interesting Language**?	○	○

Figure 6: Screenshot of the human study on evaluating coherence.