

Guided Story Generation

Lara J. Martin (she/they)

<https://laramartin.net/neurosymbolic-text-gen/>

Learning Objectives

Appraise different ways people have extracted “plots” from stories

Appraise different ways people have used script/plot-like structures to guide text generation

Consider how a guided system would work with transformers

Compare and contrast old guided story systems

Define the Story Cloze Test and determine its place in guided story generation

Review: What are procedures?

- A procedure is “a series of **actions** conducted in a certain **order** or manner,” as defined by Oxford
- A more refined definition: “a series of **steps** happening to achieve some **goal**^[1]”
 - Why?
- Examples of procedures: instructions (recipes, manuals, navigation info, how-to guide), algorithm, scientific processes, etc.
 - We focus on **instructions**, which is human-centered and task-oriented
- Examples of non-procedures: news articles, novels, descriptions, etc.
 - Those are often narrative: events do not have a specific goal
- The umbrella term is **script**^[2]

Review:

Testing LLM Knowledge of Procedures

- Task #1 Goal Inference: Given a **goal**, choose the most likely **step** out of 4 candidates.
 - **Input:** “How to prevent coronavirus”
 - **Choices:** **Wash your hands?** Wash your cat? Clap your hands? Eat your protein?
- Task #2 Step Inference: Given a **step**, choose the most likely **goal** out of 4 candidates.
 - **Input:** “Blink repeatedly.”
 - **Choices:** **Handle Pepper Spray in Your Eyes?** Relax Your Eyes? Draw Eyes? Diagnose Pink Eye?
- Task #3 Step Ordering: Given a **goal** and two unordered **steps**, determine which comes first.
 - **Input:**
Goal: How to Act After Getting Arrested.
Step (a): Get a lawyer. **Step (b):** Request bond from the judge

Review: Combining Procedures

- Models can infer goals, steps, and ordering in **existing procedures**. Can we go one step further?
- **Creating new procedures:**

If you know

how to make an **apple** pie

and

how to make a **banana** **cake**

can you infer

how to make a **banana** pie ?

- This is commonsense knowledge to humans; do language models have it?

Review: Intent Detection

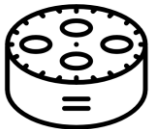
- Task-oriented dialog systems needs to match an **utterance** to an **intent**, before making informed responses
- Sentence classification task
 - Given an utterance, and some candidate intents
 - Choose the correct intent
 - Evaluated by accuracy



What's the cheapest business class flight tomorrow to Shenzhen?

Intent: **Check Flight Price**

It is \$2800 with XX airlines at 14:30.



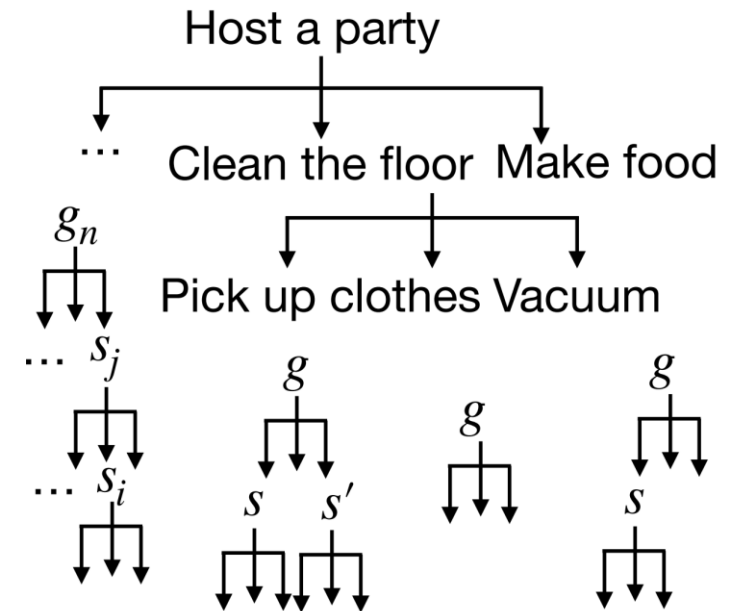
Example from Snips (Coucke et al., 2018)

Utterance: "Find the schedule at Star Theatres."

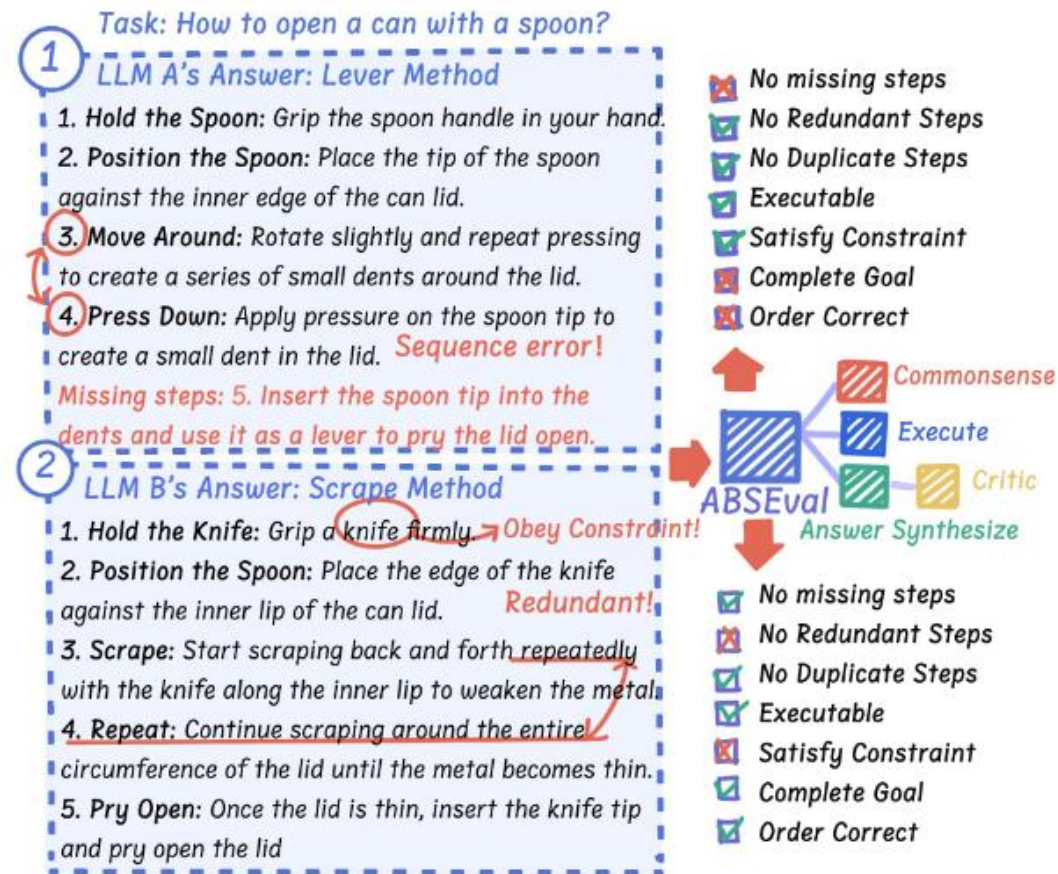
Candidate intents: Add to Playlist, Rate Book, Book Restaurant, Get Weather, Play Music, Search Creative Work, **Search Screening Event**

Review: Procedures are Hierarchical

- An event can simultaneously be a **goal** of one procedure, and a **step** in another
- A procedural hierarchy... So what?
 - Can “explain in more details” by expansion
 - Can shed light on event **granularity** (why?)
- How do you build such hierarchy?
 - To “host a party”, I need to “clean the floor”; to “clean the floor”, I need to do what?



Review: LLMs as Evaluators (LLM-as-Judge)



Think-Pair-Share

- What are some ways you might use procedures for text generation?
- How does this differ from scripts?

Scripts, Procedures, and Plots...oh my!

There are multiple ways of tying
together *events*

Review: Events

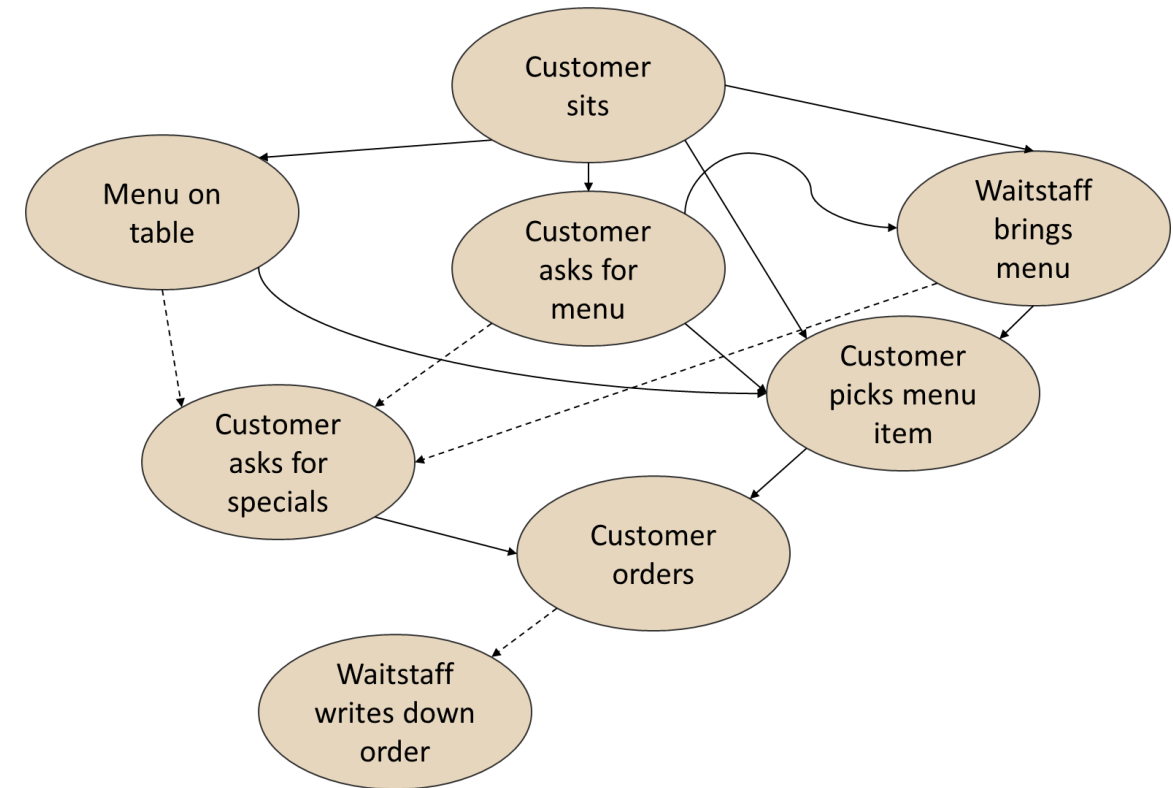
An occurrence that

- has a distinct beginning & end
- may contain multiple actions
 - E.g., “paying the bill” might involve taking out your credit card, handing it over to the staff, etc.
- can be at a variety of levels of granularity
 - E.g. “opens wallet” vs “pays the bill” vs “ate at the restaurant”

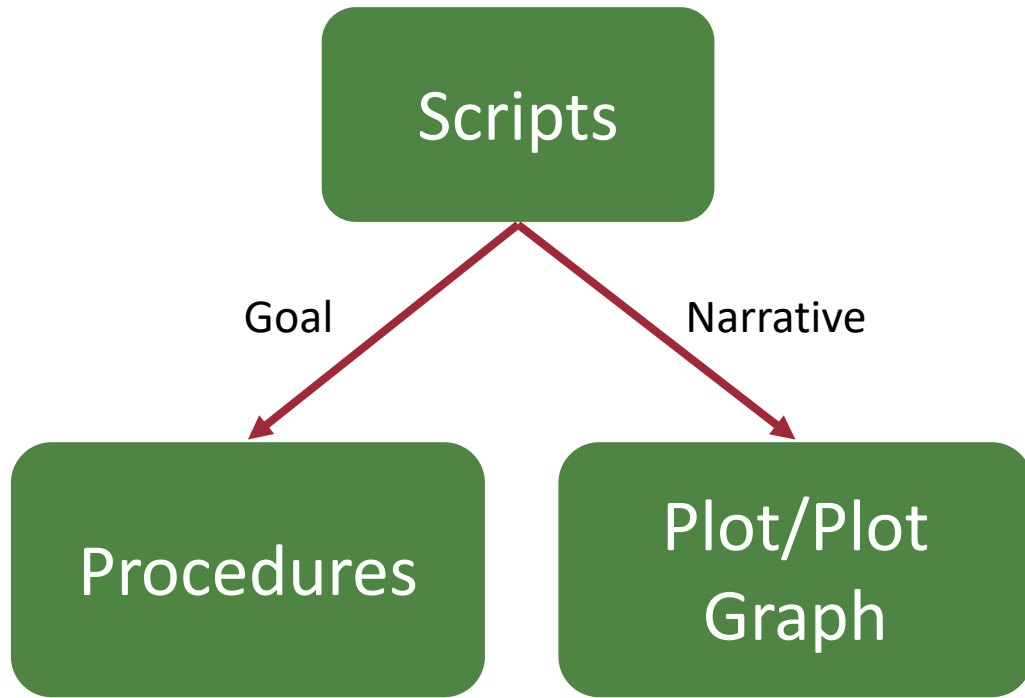
In storytelling, we might consider this as something that moves the narrative along
(As opposed to descriptions of people/places: “It was a dark and stormy night...”)

Scripts, Procedures, and Plots...oh my!

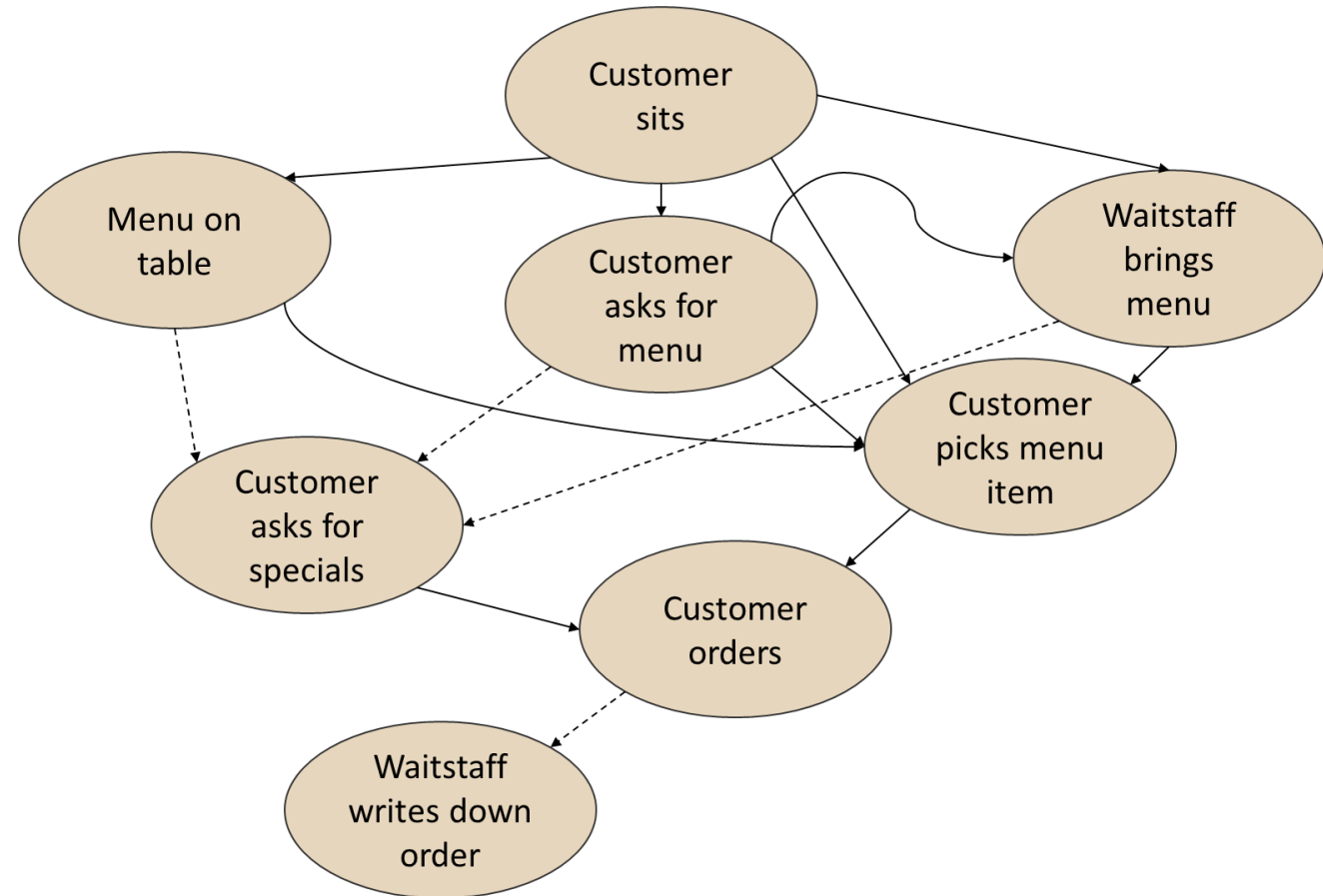
Schank & Abelson believe that everyone has **scripts** in their heads built from common experiences



Scripts, Procedures, and Plots...oh my!



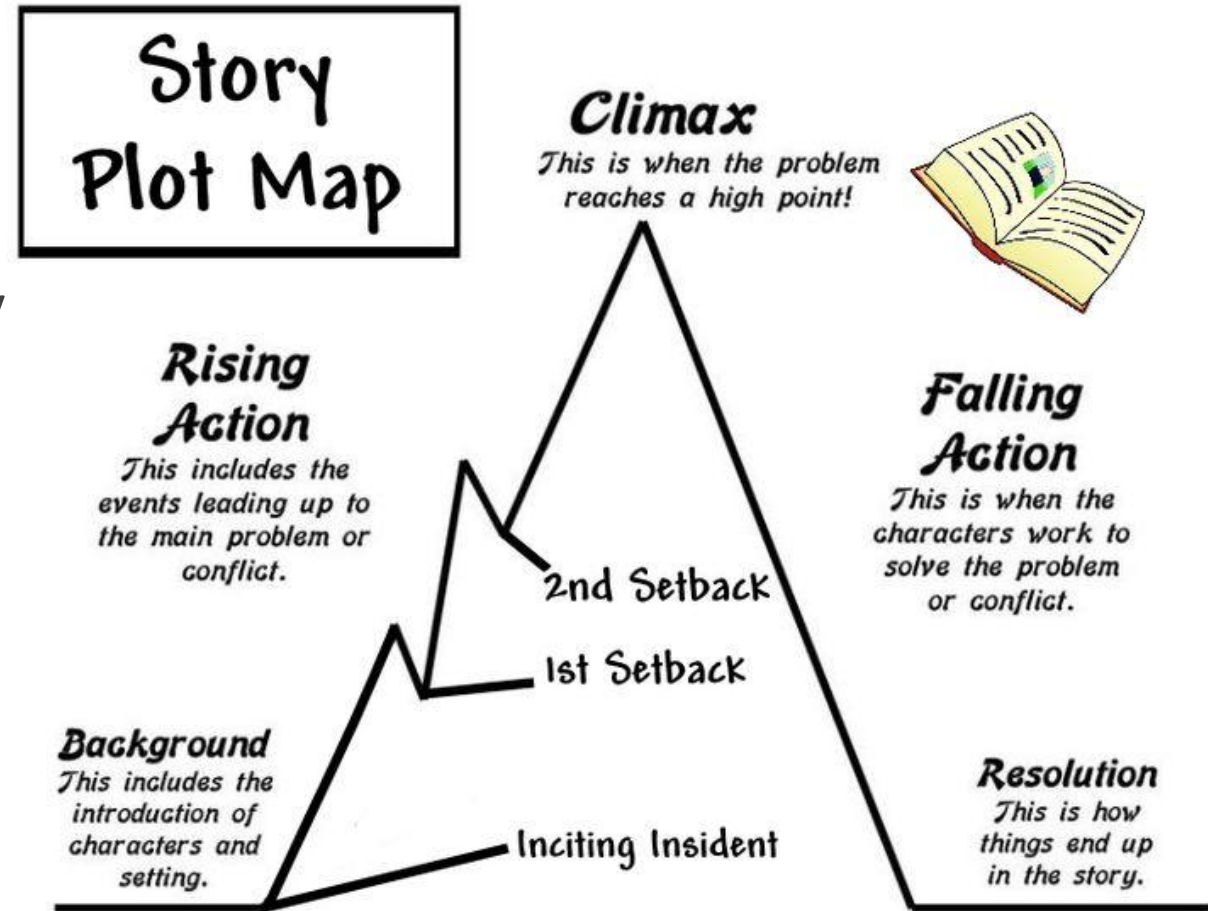
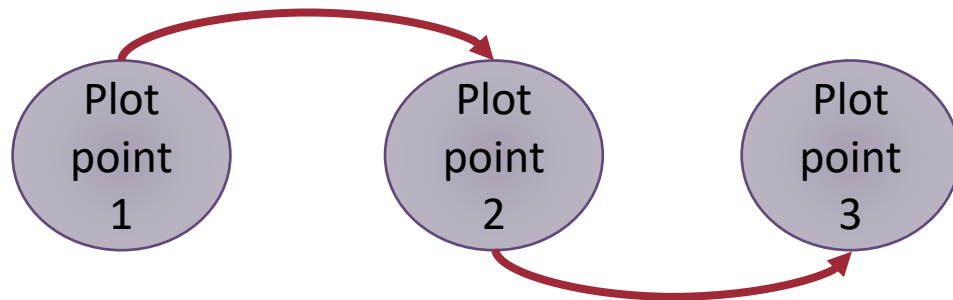
(Rough Dichotomy)



Scripts, Procedures, and Plots...oh my!

Schank & Abelson believe that everyone has scripts in their heads built from common experiences

Authors often plan out **plots** before they write stories



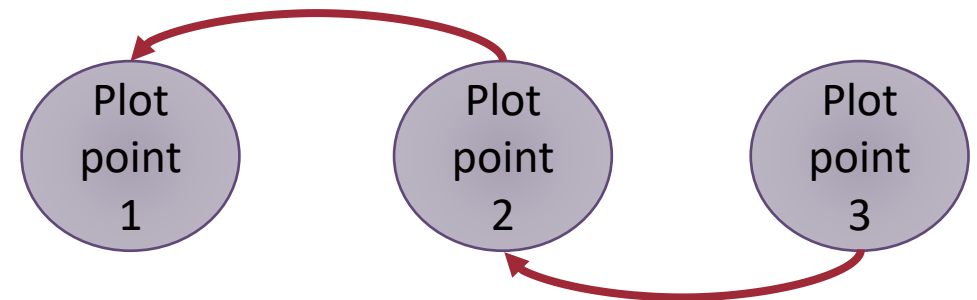
<https://i.pinimg.com/736x/57/f7/03/57f703afc709080bddc2c3cfed8dd061.jpg>

Scripts, Procedures, and Plots...oh my!

Schank & Abelson believe that everyone has scripts in their heads built from common experiences

Authors often plan out plots before they write stories

Stories that aren't planned out either have to “**reincorporate**”[1] ideas or the stories feel unfinished



[1] The idea of *reincorporation* is explored in the book [Impro by Keith Johnstone](#)

(1)

Plan-and-Write: Towards Better Automatic Storytelling

Lili Yao,^{1,3*} Nanyun Peng,^{2*} Ralph Weischedel,² Kevin Knight,² Dongyan Zhao,¹ Rui Yan^{1†}

liliyao@tencent.com, {npeng,weisched,knight}@isi.edu

{zhaodongyan,ruiyan}@pku.edu.cn

¹Institute of Computer Science and Technology, Peking University

²Information Sciences Institute, University of Southern California, ³Tencent AI Lab

Abstract

Automatic storytelling is challenging since it requires generating long, coherent natural language to describes a sensible sequence of events. Despite considerable efforts on automatic story generation in the past, prior work either is restricted in plot planning, or can only generate stories in a narrow domain. In this paper, we explore open-domain story generation that writes stories given a title (topic) as input. We propose a *plan-and-write* hierarchical generation framework that first plans a storyline, and then generates a story based on the storyline. We compare two planning strategies. The *dynamic* schema interweaves story planning and its surface realization in text, while the *static* schema plans out the entire storyline before generating stories. Experiments show that with explicit storyline planning, the generated stories are more diverse, coherent, and on topic than those generated without creating a full plan, according to both automatic and human evaluations.

Introduction

A narrative or story is anything which is told in the form of a causally/logically linked set of events involving some

Title (Given)	The Bike Accident
Storyline (Extracted)	Carrie → bike → sneak → nervous → leg
Story (Human Written)	<u>Carrie</u> had just learned how to ride a bike. She didn't have a <u>bike</u> of her own. Carrie would <u>sneak</u> rides on her sister's bike. She got <u>nervous</u> on a hill and crashed into a wall. The bike frame bent and Carrie got a deep gash on her <u>leg</u> .

Table 1: An example of title, storyline and story in our system. A storyline is represented by an ordered list of words.

and Young 2010), we propose to decompose story generation into two steps: 1) story planning which generates plots, and 2) surface realization which composes natural language text based on the plots. We propose a *plan-and-write* hierarchical generation framework that combines plot planning and surface realization to generate stories from titles.

Extracting Plots

Carrie had just learned how to ride a bike. She didn't have a bike of her own. Carrie would sneak rides on her sister's bike. She got nervous on a hill and crashed into a wall. The bike frame bent and Carrie got a deep gash on her leg.

Carrie→bike→sneak→nervous→leg

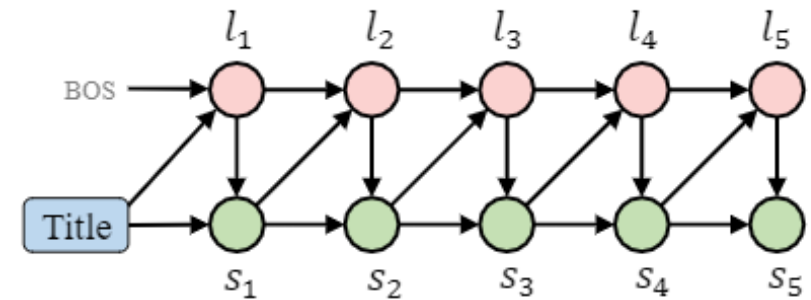
Plan-and-Write Overview

Extracted most important word from each sentence using RAKE algorithm (keyword extraction) to create a storyline (aka plot)

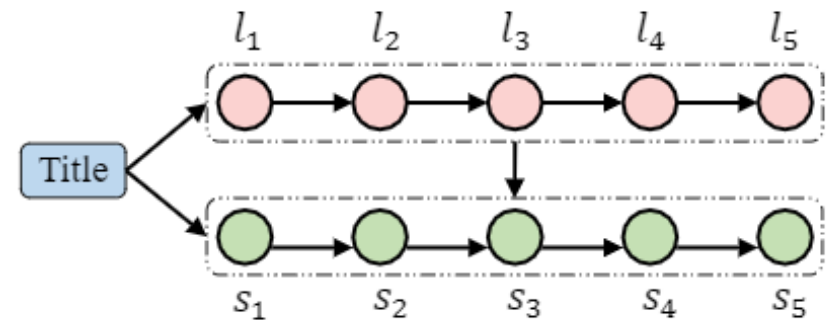
Used storyline as input to plan out stories

Dynamic generation → using storyline and sentences to inform each other

Static generation → plan ahead and then generate



(a) Dynamic schema work-flow.



(b) Static schema work-flow.

Plan-and-Write System

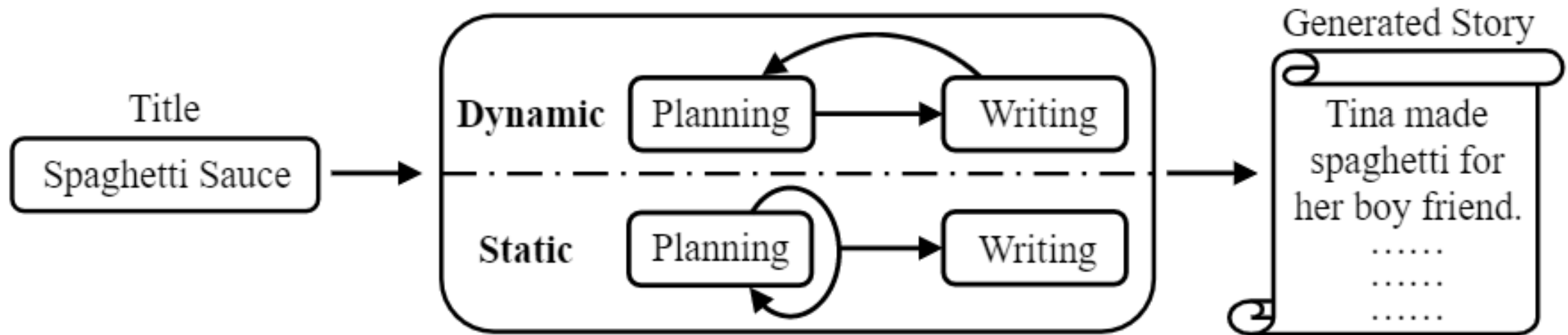


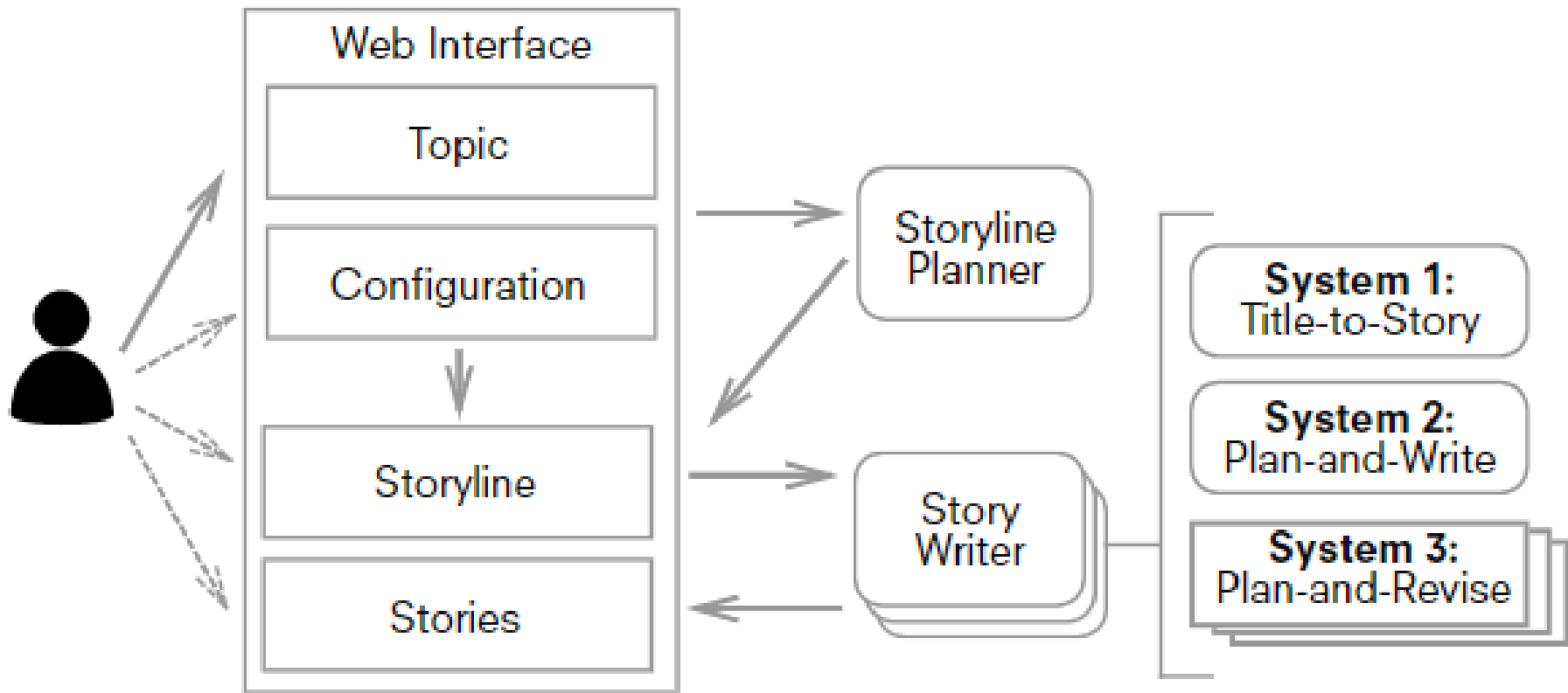
Figure 1: An overview of our system.

Examples

Which story would you consider better?

Title: Computer		
Baselines	Inc-S2S	Tom's computer broke down. He needed to buy a new computer. He decided to buy a new computer. Tom bought a new computer. Tom was able to buy a new computer.
	Cond-LM	The man bought a new computer. He went to the store. He bought a new computer. He bought the computer. He installed the computer.
Dynamic	Storyline	needed → money → computer → bought → happy
	Story	John <u>needed</u> a computer for his birthday. He worked hard to earn <u>money</u> . John was able to buy his <u>computer</u> . He went to the store and <u>bought</u> a computer. John was <u>happy</u> with his new computer.
Static	Storyline	computer → slow → work → day → buy
	Story	I have an old <u>computer</u> . It was very <u>slow</u> . I tried to <u>work</u> on it but it wouldn't work. One <u>day</u> , I decided to buy a new one. I <u>bought</u> a new computer .

Plan, Write, and Revise



Plan, Write, and Revise

Stories v1.0 Auto Interactive Advanced ▾ 6.55 seconds

weight lifting Generate

Ready

Storyline

weights -> saw -> decided impress -> struggled -> learned

Title to Story

i wanted to lose some weight .
i decided to go on a diet .
alas , i lost my weight .
i realized i needed to lose weight .
i decided to lose weight .

Plan and Write

tim was trying to lift weights.
he saw an ad for a gym.
he decided to impress them.
he struggled to lift them.
tim learned how to lift weights.

Plan and Revise

sam was trying to lift weights.
he saw an ad for a gym.
he decided to impress them.
he struggled to do so.
he learned a lot about himself.

(a) cross-model interaction, comparing three models with advanced options to alter the storyline and story diversities.

Stories v1.0 Auto Interactive Advanced ▾ 0.54 seconds Ready

Title

culture shock

Storyline

vacation country

wanted

tried food asked

confusing ↺

hilarious

You may edit the storyline phrases at any time.

Story

tom ^{was} went on vacation in ^{the} a new country.

he wanted to try something new.

he tried a new kind of food. ^{that he liked}

it was confusing. ↺

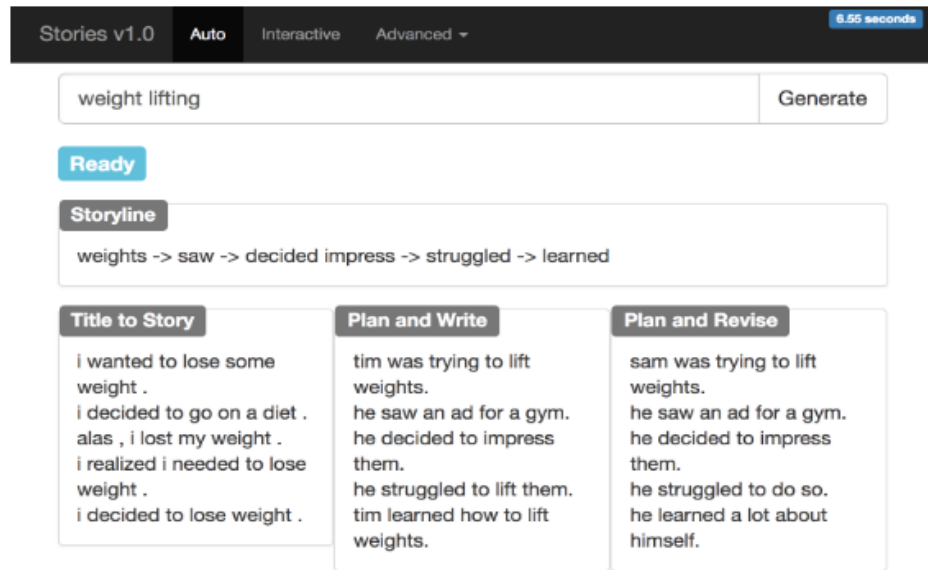
tom did n't know how to be polite. ↺

You may edit the story sentences at any time.

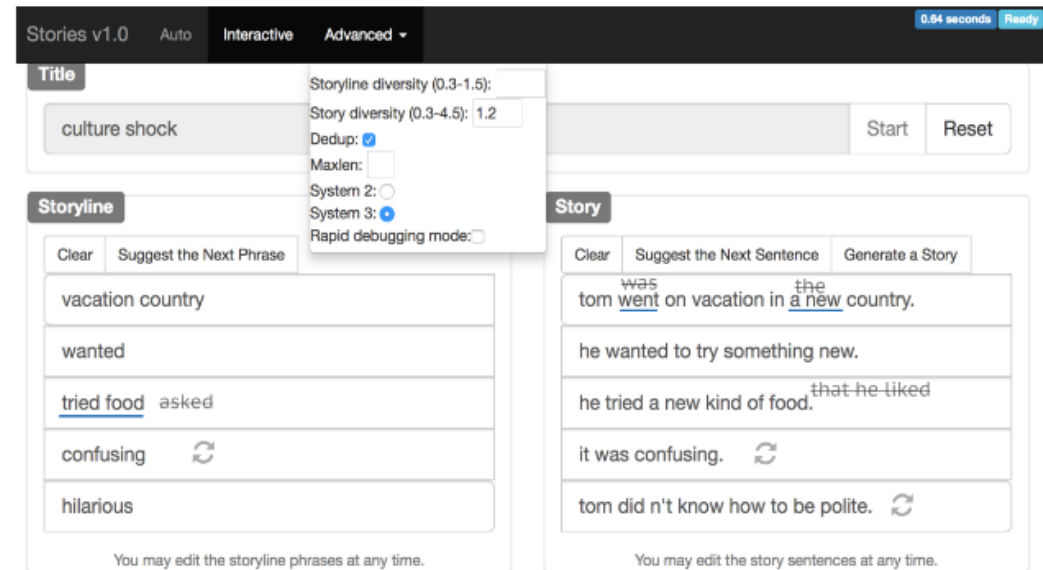
(b) intra-model interaction, showing advanced options and annotated with user interactions from an example study.

Updating Plan, Write, & Revise?

Plan, Write, and Revise was written in 2019. How might you “update” this work?
Does it need to be updated?



(a) cross-model interaction, comparing three models with advanced options to alter the storyline and story diversities.



(b) intra-model interaction, showing advanced options and annotated with user interactions from an example study.

Re³

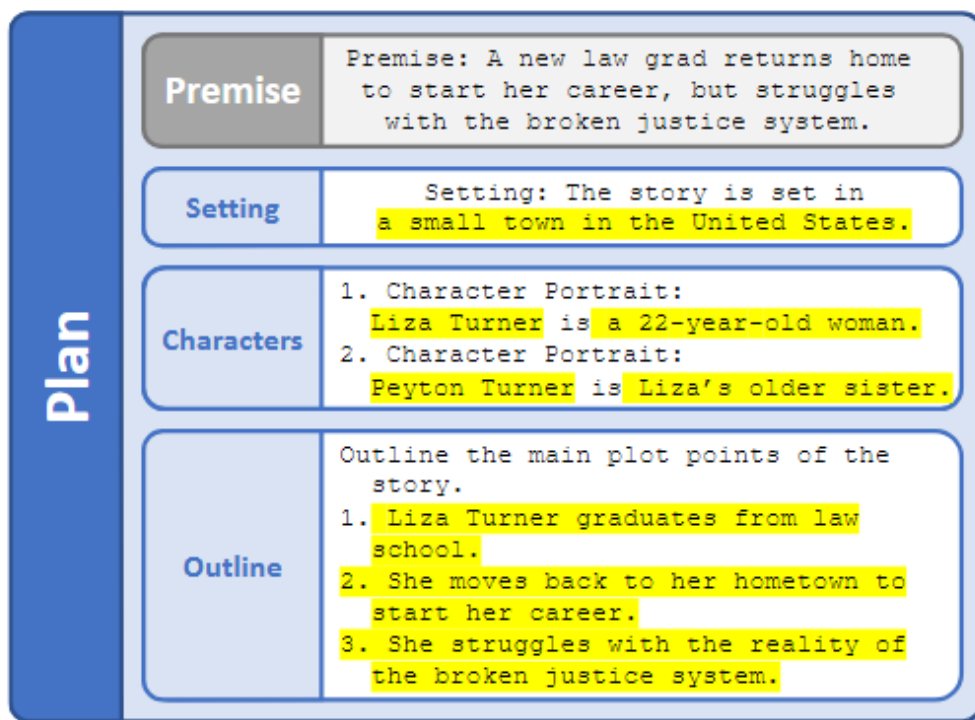


Figure 2: Illustration of Re³'s Plan module, which prompts a language model to generate a setting, characters, and outline based on the premise. Highlighting indicates generated text.

Re³: Generating Longer Stories With Recursive Reprompting and Revision

Kevin Yang¹ Yuandong Tian² Nanyun Peng³ Dan Klein¹

¹UC Berkeley, ²Meta AI, ³UCLA

{yangk,klein}@berkeley.edu, yuandong@meta.com, violetpeng@cs.ucla.edu

Abstract

We consider the problem of automatically generating longer stories of over two thousand words. Compared to prior work on shorter stories, long-range plot coherence and relevance are more central challenges here. We propose the Recursive Reprompting and Revision framework (Re³) to address these challenges by (a) prompting a general-purpose language model to construct a structured overarching plan, and (b) generating story passages by repeatedly injecting contextual information from both the plan and current story state into a language model prompt. We then revise by (c) reranking different continuations for plot coherence and premise relevance, and finally (d) editing the best continuation for factual consistency. Compared to similar-length stories generated directly from the same base model, human evaluators judged substantially more of Re³'s stories as having a coherent overarching plot (by 14% absolute increase), and relevant to the given initial premise (by 20%).

1 Introduction

Generating long-term coherent stories is a long-standing challenge for artificial intelligence, requir-

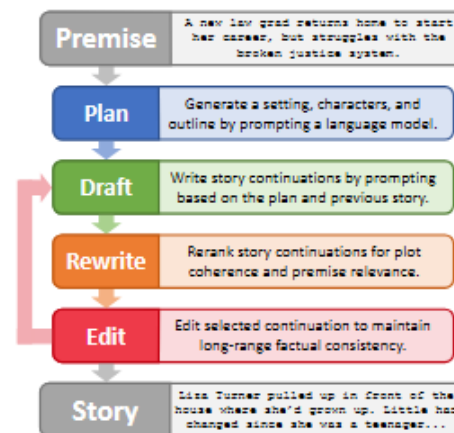
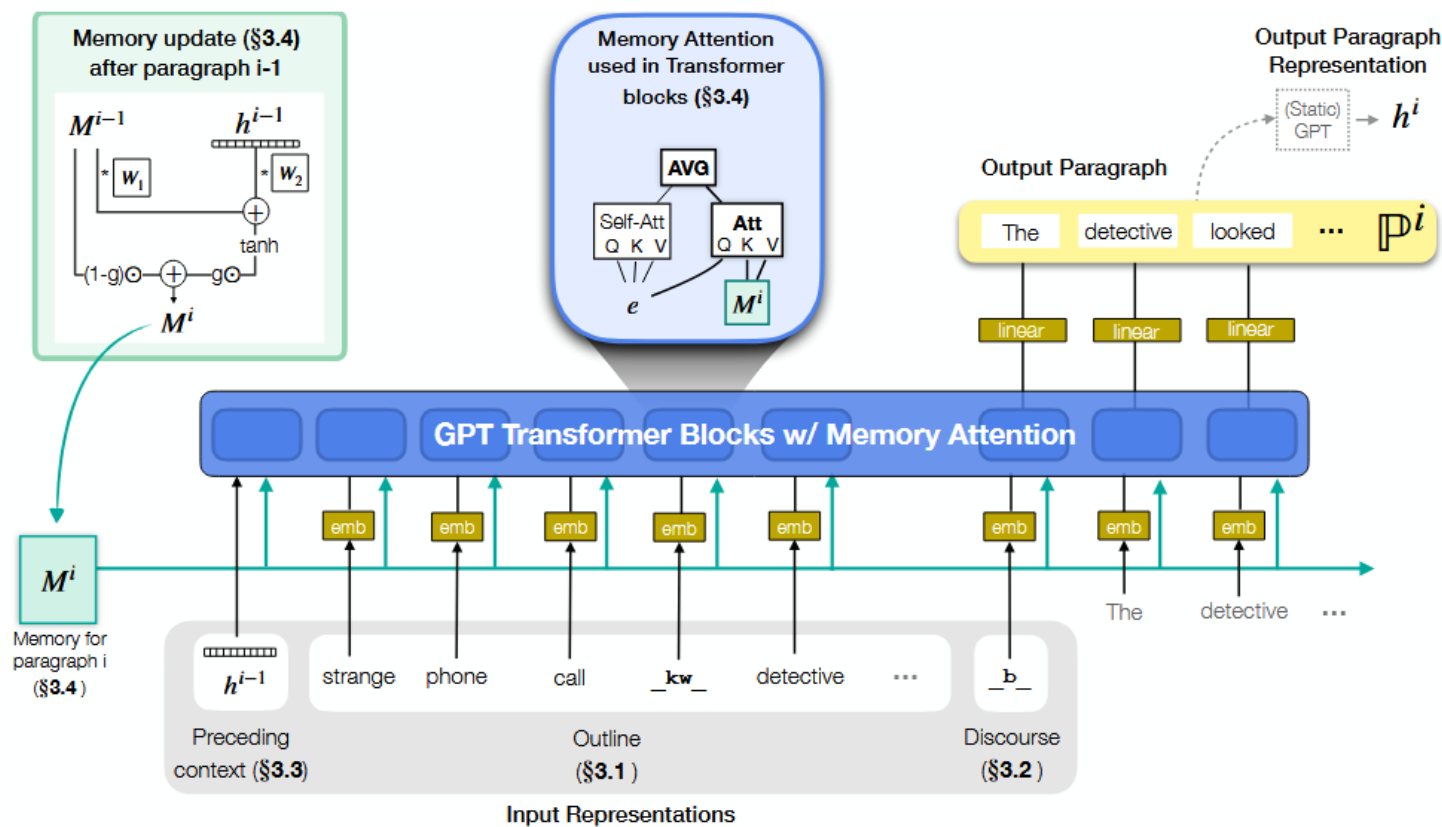


Figure 1: High-level overview of Re³.

length increases limited primarily by evaluation rather than technical issues.¹ Generating stories of such length faces qualitatively new challenges compared to prior work on shorter stories. First, the system must maintain a coherent overarching plot over thousands of words. Given an initial premise, it should maintain relevance to this premise over thousands of words as well. Additional challenges include preservation of narration style and avoiding

PlotMachines



PLOTMACHINES: Outline-Conditioned Generation with Dynamic Plot State Tracking

Hannah Rashkin¹, Asli Celikyilmaz², Yejin Choi^{1,3}, Jianfeng Gao²

¹ Paul G. Allen School of Computer Science & Engineering, University of Washington

² Microsoft Research, Redmond, WA, USA

³ Allen Institute for Artificial Intelligence, Seattle, WA, USA

{hrashkin,yejin}@cs.washington.edu, {aslice1,jfgao}@microsoft.com

Abstract

We propose the task of *outline-conditioned story generation*: given an outline as a set of phrases that describe key characters and events to appear in a story, the task is to generate a coherent narrative that is consistent with the provided outline. This task is challenging as the input only provides a rough sketch of the plot, and thus, models need to generate a story by interweaving the key points provided in the outline. This requires the model to keep track of the dynamic states of the latent plot, conditioning on the input outline while generating the full story. We present PLOTMACHINES, a neural narrative model that learns to transform an outline into a coherent story by tracking the dynamic plot states. In addition, we enrich PLOTMACHINES with high-level discourse structure so that the model can learn different writing styles corresponding to different parts of the narrative. Comprehensive experiments over three fiction and non-fiction datasets demonstrate that large-scale language

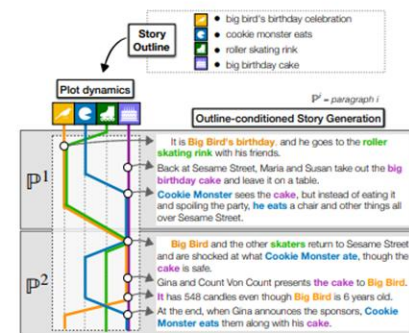


Figure 1: An outline (input) paired with a story (output) from the Wikiplots training set. Plot elements from the outline can appear and reappear non-linearly throughout the plot, as shown in plot dynamics graph. Composing stories from an outline requires keeping track of how outline phrases have been used while writing.

(2)

Story Realization: Expanding Plot Events into Sentences

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung,
Zhaochen Luo, William Ma, Lara J. Martin, Mark O. Riedl

School of Interactive Computing

Georgia Institute of Technology

{raj.amanabrolu, etien, wcheung8, zluo, wma61, ljmartin, riedl}@gatech.edu

Abstract

Neural network based approaches to automated story plot generation attempt to learn how to generate novel plots from a corpus of natural language plot summaries. Prior work has shown that a semantic abstraction of sentences called *events* improves neural plot generation and allows one to decompose the problem into: (1) the generation of a sequence of events (event-to-event) and (2) the transformation of these events into natural language sentences (event-to-sentence). However, typical neural language generation approaches to event-to-sentence can ignore the event details and produce grammatically-correct but semantically-unrelated sentences. We present an ensemble-based model that generates natural language guided by events. We provide results—including a human subjects study—for a full end-to-end automated story generation system showing that our method generates more coherent and plausible stories than baseline approaches¹.

1 Introduction

Automated story plot generation is the problem of creating a sequence of main plot points for a story in a given domain. Generated plots must remain consistent across the entire story, preserve long-term dependencies, and make

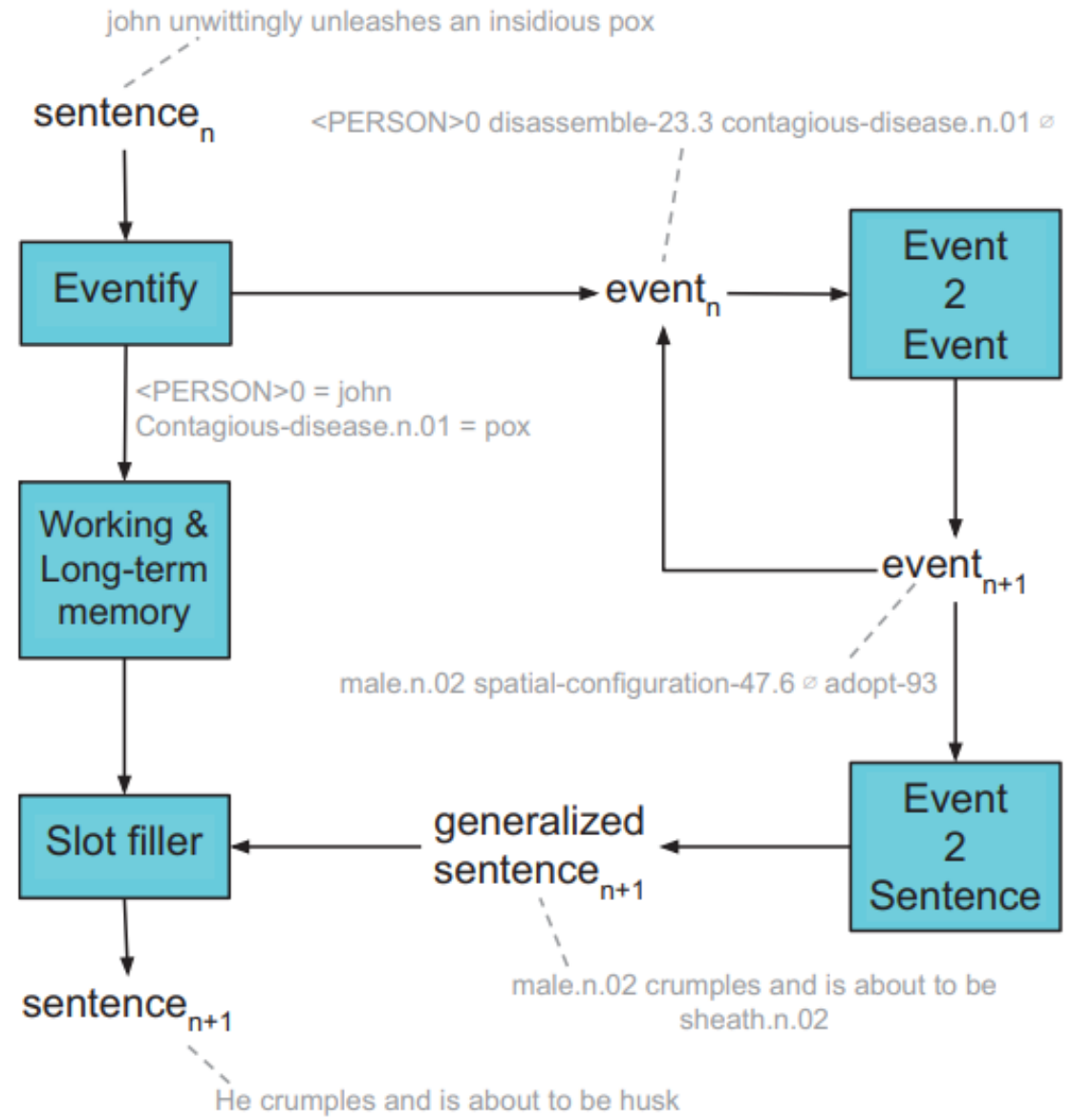
for explicit domain modeling beyond providing a corpus of example stories. The primary pitfall of neural language model approaches for story generation is that the space of stories that can be generated is huge, which in turn, implies that, in a textual story corpora, any given sentence will likely only be seen once.

Martin et al. (2018) propose the use of a semantic abstraction called an *event*, reducing the sparsity in a dataset that comes from an abundance of unique sentences. They define an event to be a unit of a story that creates a change in the story world’s state. Technically, an event is a tuple containing a subject, verb, direct object, and some additional disambiguation token(s).

The event representation enables the decomposition of the plot generation task into two sub-problems: *event-to-event* and *event-to-sentence*. Event-to-event is broadly the problem of generating the sequence of events that together comprise a plot. Models used to address this problem are also responsible for maintaining plot coherence and consistency. Once new events are generated, however, they are still not human-readable. Thus the second sub-problem, event-to-sentence, focuses on transforming these events into natural language sentences.

Story Realization

Extract events from stories



Review:

Probabilistic Event Representation

From sentence, extract event representation:
(subject, verb, direct object, modifier, preposition)

Original sentence: yoda uses the force to take apart the platform

Events:

- yoda use force ∅ ∅
- yoda take_apart platform ∅ ∅

Generalized Events:

<PERSON>0 fit-54.3 power.n.01 ∅ ∅

<PERSON>0 destroy-44 surface.n.01 ∅ ∅

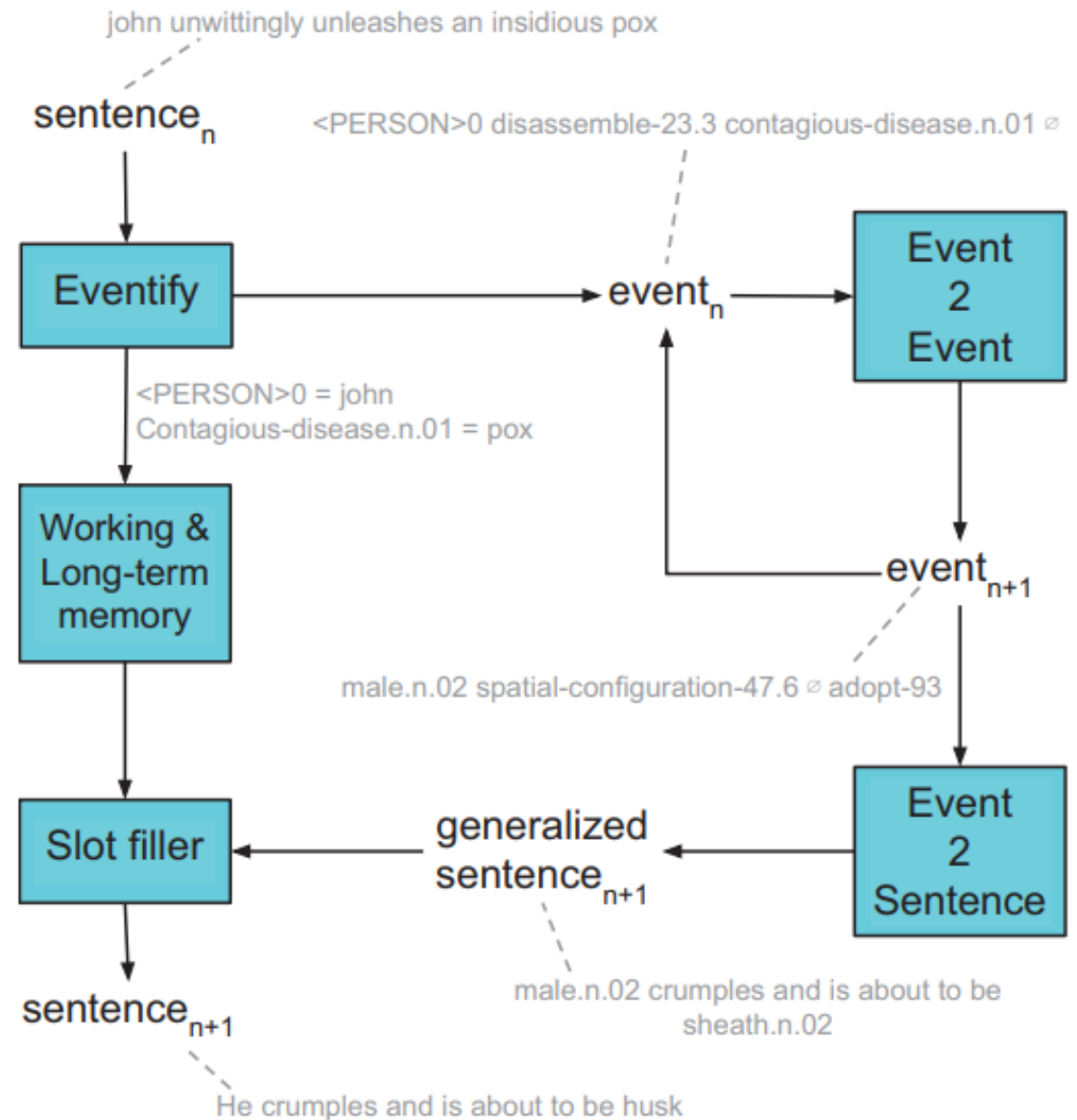
Story Realization

Extract events from stories

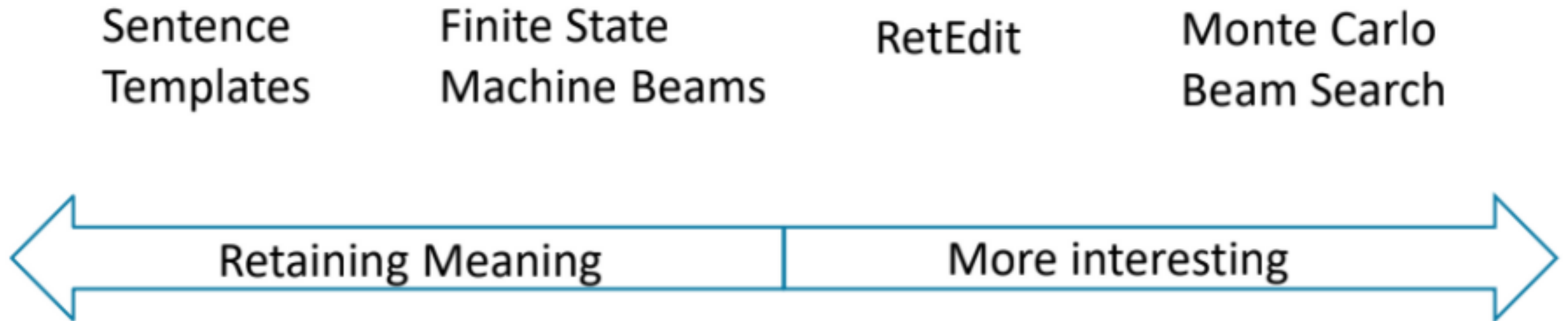
Generate the plot using a seq2seq network

Use an ensemble of methods to find the best sentence given an event

Get a confidence score from each model, and accept the sentence if it's above a threshold



Balance of Meaning and Interestingness



Story Realization: Cascading Ensemble

RetEdit

Sentence Templates

Monte Carlo Beam Search

Finite State Machine Constrained Beams

Seq2Seq (Greedy Decoding)

RetEdit

Retrieve: Map event to its closest sentence and retrieve that

Edit:

- Seq2seq with attention and copying (Gu et al., 2016)
- Takes the retrieved sentence and the original input event, then edits

Confidence score: proportional to $1 - \text{retrieval distance}$

Input Event	RetEdit
$\langle \langle \text{PRP} \rangle, \text{act-114-1-1, to, } \emptyset, \text{event.n.01} \rangle$	$\langle \text{PRP} \rangle$ and $\langle \text{PERSON} \rangle 0$ move to the event.n.01 of the natural_object.n.01.
$\langle \langle \text{PERSON} \rangle 2, \text{send-11.1, through, } \langle \text{PERSON} \rangle 6, \langle \text{LOCATION} \rangle 1 \rangle$	$\langle \text{PERSON} \rangle 2$ sends $\langle \text{PERSON} \rangle 6$ through the $\langle \text{LOCATION} \rangle 1$.

Sentence Templates

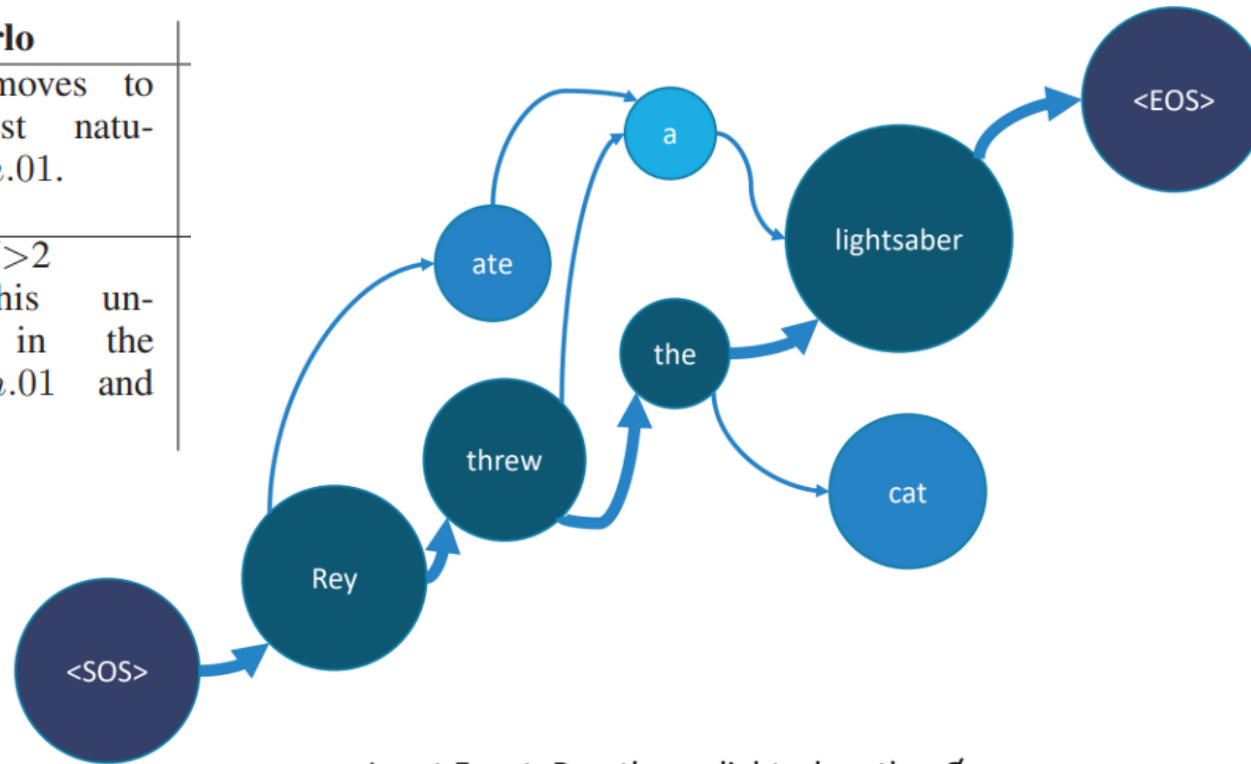
Input Event	Templates
$\langle \langle \text{PRP} \rangle, \text{act-114-1-1}, \text{to}, \emptyset, \text{event.n.01} \rangle$	$\langle \text{PRP} \rangle$ act-114-1-1 to event.n.01.
$\langle \langle \text{PERSON} \rangle 2, \text{send-11.1}, \text{through}, \langle \text{PERSON} \rangle 6, \langle \text{LOCATION} \rangle 1 \rangle$	The $\langle \text{PERSON} \rangle 2$ send-11.1 the $\langle \text{PERSON} \rangle 6$ through $\langle \text{LOCATION} \rangle 1$.

$$\begin{aligned}
 S &\rightarrow NP \ v \ (NP) \ (PP) \\
 NP &\rightarrow d \ n \\
 PP &\rightarrow p \ NP
 \end{aligned}
 \quad [_ _ s] \{ v \ [_ _ o] \ [p \ _ _ m] \}$$

$$\text{Confidence score: } 1 - \frac{\sum \text{loss}}{\text{sentence length}}$$

Monte Carlo Beam Search

Input Event	Monte Carlo
$\langle \langle \text{PRP} \rangle, \text{act-114-1-1, to, } \emptyset, \text{event.n.01} \rangle$	$\langle \text{PRP} \rangle$ moves to the nearest natural_object.n.01.
$\langle \langle \text{PERSON} \rangle 2, \text{send-11.1, through, } \langle \text{PERSON} \rangle 6, \langle \text{LOCATION} \rangle 1 \rangle$	$\langle \text{PERSON} \rangle 2$ passes this undercover in the body_part.n.01 and collapses.



Beams weighted to favor tokens in input

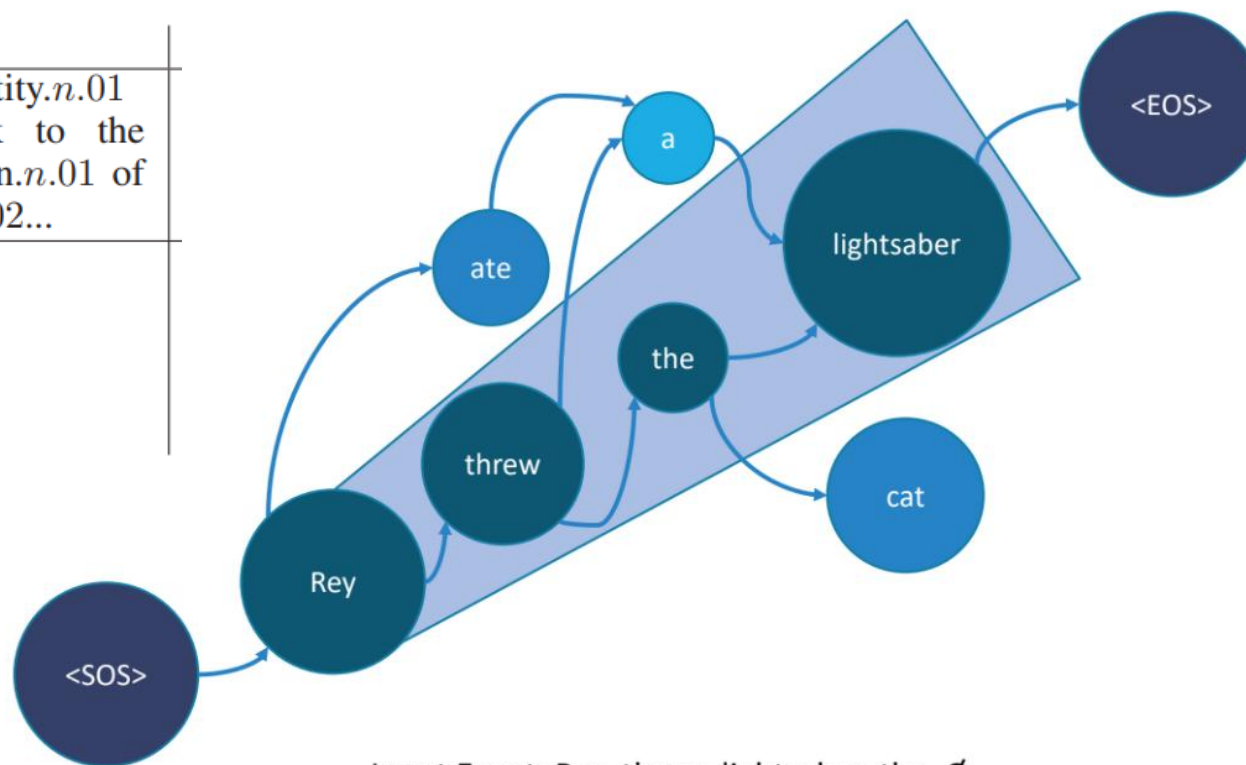
Weights learned at validation

Input Event: Rey, threw, lightsaber, the, $\emptyset$

Confidence Score: $s_t = \alpha * s_{t-1} + (1 - \alpha) * \text{AVG}(\text{playout}_t)$

Finite State Machine Constrained Beams

Input Event	FSM
$\langle \langle \text{PRP} \rangle, \text{act-114-1-1, to, } \emptyset, \text{event.n.01} \rangle$	physical_entity.n.01 move back to the phenomenon.n.01 of the craft.n.02...
$\langle \langle \text{PERSON} \rangle 2, \text{send-11.1, through, } \langle \text{PERSON} \rangle 6, \langle \text{LOCATION} \rangle 1 \rangle$	$\emptyset$



Input Event: Rey, threw, lightsaber, the, $\emptyset$

Confidence Score: Match at least 3 tokens in input

Examples

Table 1: Event-to-sentence examples for each model. $\emptyset$ represents an empty parameter; $\langle \text{PRP} \rangle$ is a pronoun.

Input Event	RetEdit	Templates	Monte Carlo	FSM	Gold Standard
$\langle \langle \text{PRP} \rangle, \text{act-114-1-1}, \text{to}, \emptyset, \text{event.n.01} \rangle$	$\langle \text{PRP} \rangle$ and $\langle \text{PERSON} \rangle 0$ move to the event.n.01 of the natural_object.n.01.	$\langle \text{PRP} \rangle$ act-114-1-1 to event.n.01.	$\langle \text{PRP} \rangle$ moves to the nearest natural_object.n.01.	physical_entity.n.01 move back to the phenomenon.n.01 of the craft.n.02...	$\langle \text{PRP} \rangle$ move to the event.n.01.
$\langle \langle \text{PERSON} \rangle 2, \text{send-11.1}, \text{through}, \langle \text{PERSON} \rangle 6, \langle \text{LOCATION} \rangle 1 \rangle$	$\langle \text{PERSON} \rangle 2$ sends $\langle \text{PERSON} \rangle 6$ through the $\langle \text{LOCATION} \rangle 1$.	The $\langle \text{PERSON} \rangle 2$ send-11.1 the $\langle \text{PERSON} \rangle 6$ through $\langle \text{LOCATION} \rangle 1$.	$\langle \text{PERSON} \rangle 2$ passes this undercover in the body_part.n.01 and collapses.	$\emptyset$	In activity.n.01 to avoid $\langle \text{PRP} \rangle$ out.n.01 $\langle \text{PERSON} \rangle 2$ would transport $\langle \text{PERSON} \rangle 6$ through the $\langle \text{LOCATION} \rangle 1$.

End-to-End Examples

Table 2: End-to-end pipeline examples on previously-unseen input data. The Event-to-Sentence model used is the full ensemble. Sentences are generated using both the extracted and generated events.

Input Sent.	Extracted event	Generated Events (Event-to-Event)	Generated Sentences (Event-to-Sentence)	Slot-filled Sentences
On Tatooine, Jabba the Hutt inspects the drone barge recently delivered to him.	<<ORG>0, assessment-34.1, ∅, vessel.n.02, ∅>	<<PERSON>1, settle-36.1.2, ∅, indicator.n.03, indicator.n.03> ; <music.n.01, escape-51.1-1, from, ∅, ∅>; <<PRP>, discover-84, to, run-51.3.2, progenitor.n.01>	The <ORG>0 can not scan the vessel.n.02 of the <VESSEL>0. <PERSON>1 decides to be a little person.n.01 at the structure.n.01. the music.n.01 arrives. <PRP> finds a lonely person.n.01 on the upper one of the craft.n.02 which is not a personal_letter.n.01 but does not respond to hails .	The Jabba the Hutt can not scan the bareboat of the Uss Lakota. O Yani decides to be a little mailer at the air-dock. The Music arrives. She finds a lonely mailer on the upper one of the bareboat which is not a love letter but does not respond to hails.
Boba Fett has just chased down another bounty, a Rodian art dealer who sold fake works to Gebbu the Hutt.	<<PERSON>0, chase-51.6, ∅, bounty.n.04, ∅>	<<PERSON>0, chase-51.6, to, magnitude.n.01, ∅>; <magnitude.n.01, comprehend-87.2, off, craft.n.02, magnitude.n.01>; <<PERSON>2, amuse-31.1, off, ∅, ∅>; <<PERSON>2, discover-84, off, change_of_integrity.n.01, ∅>	<PERSON>0 enters the bounty.n.04 and tells <PRP>. <PERSON>0 attaches the explosive.a.01 to the person.n.01 who is trying to fix the device.n.01. the magnitude.n.01 doesn't know the craft.n.02 off the craft.n.02. <PERSON>2 is surprised when <PRP> learns that the person.n.01 is actually <PERSON>7. <PERSON>2 sees the change_of_integrity.n.01 and tells <PRP>.	Boba Fett enters the bounty and tells it. Boba Fett attaches the explosive to the peer who is trying to fix the toy. The multiplicity doesn't know the bounty off the bounty. Dark Jedi Lomi Plo is surprised when it learns that the peer is actually Mrs Connors. Dark Jedi Lomi Plo sees the combination off the Orbs and tells them.

Guided Open Story Generation Using Probabilistic Graphical Models

Sagar Gandhi*
University of Kentucky
Lexington, Kentucky
sga267@uky.edu

Brent Harrison
University of Kentucky
Lexington, Kentucky
harrison@cs.uky.edu

ABSTRACT

In this work, we present an approach for performing computational storytelling in open domain based on Author Goals. Author Goals are constraints placed on a story event directed by the author of the system. There are two challenges present in this type of story generation: (1) automatically acquiring a model of story progression, and (2) guiding the progress of story progression in light of different goals. We propose a novel approach to story generation based on probabilistic graphical models and Loopy Belief Propagation (LBP) that addresses both of these problems. We show the applicability of our technique through a case study on the Visual Storytelling (VIST) 2017 dataset. We use image descriptions as author goals. This empirical analysis suggests that our approach is able to utilize goals information to better automatically generate stories.

CCS CONCEPTS

• **Computing methodologies** → **Probabilistic reasoning**; *Discourse, dialogue and pragmatics; Natural language generation; Learning in probabilistic graphical models.*

KEYWORDS

Belief Propagation, Computational Storytelling, Natural Language Generation, Probabilistic Graphical Models

ACM Reference Format:

Sagar Gandhi and Brent Harrison. 2019. Guided Open Story Generation Using Probabilistic Graphical Models. In *The Fourteenth International Conference on the Foundations of Digital Games (FDG '19)*, August 26–30, 2019, San Luis Obispo, CA, USA, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 7 pages. <https://doi.org/10.1145/3337722.3341871>

problem *domain model* [3, 5, 25]. This type of story generation is sometimes called *Closed Story Generation*. While stories generated in this way are often coherent, the space of possible stories that can be generated is tightly coupled with the domain model. In order to generate different types of stories, a new domain model must be authored, which is often a time consuming task.

Open Story Generation seeks to address this limitation by enabling stories to be told in any conceivable domain through the use of machine learning. The goal is to use these systems to tell a variety of stories without the need for re-learning specific domain models.

One limitation of open story generation systems is that, due to the complexity of the models used for training, it can be difficult to encode specific author goals into the story generation process. To address this limitation of open story generation systems, we propose a new framework that can reason about the overall structure of a story as well as how to incorporate author goals into the story generation process. This system enables many different types of stories to be generated based on these author goals without needing to retrain or re-learn a domain model.

In this paper, we use a novel approach for open story generation based on probabilistic graphical models. This gives us an enhanced ability to reason about story structure when compared to neural-based approaches. There are also several ways to perform inference over these structures including MCMC sampling and reinforcement learning, but we choose to model story structure and author goals using bipartite graphs and perform inference over them using Loopy Belief Propagation (LBP). We use LBP rather than MCMC or reinforcement learning because the latter approaches often require a large amount of data to perform inference which may not be readily available in practice.

1) Extract Semantic Relationships

Use discourse representation structure (DRS) parser to get semantic relationships

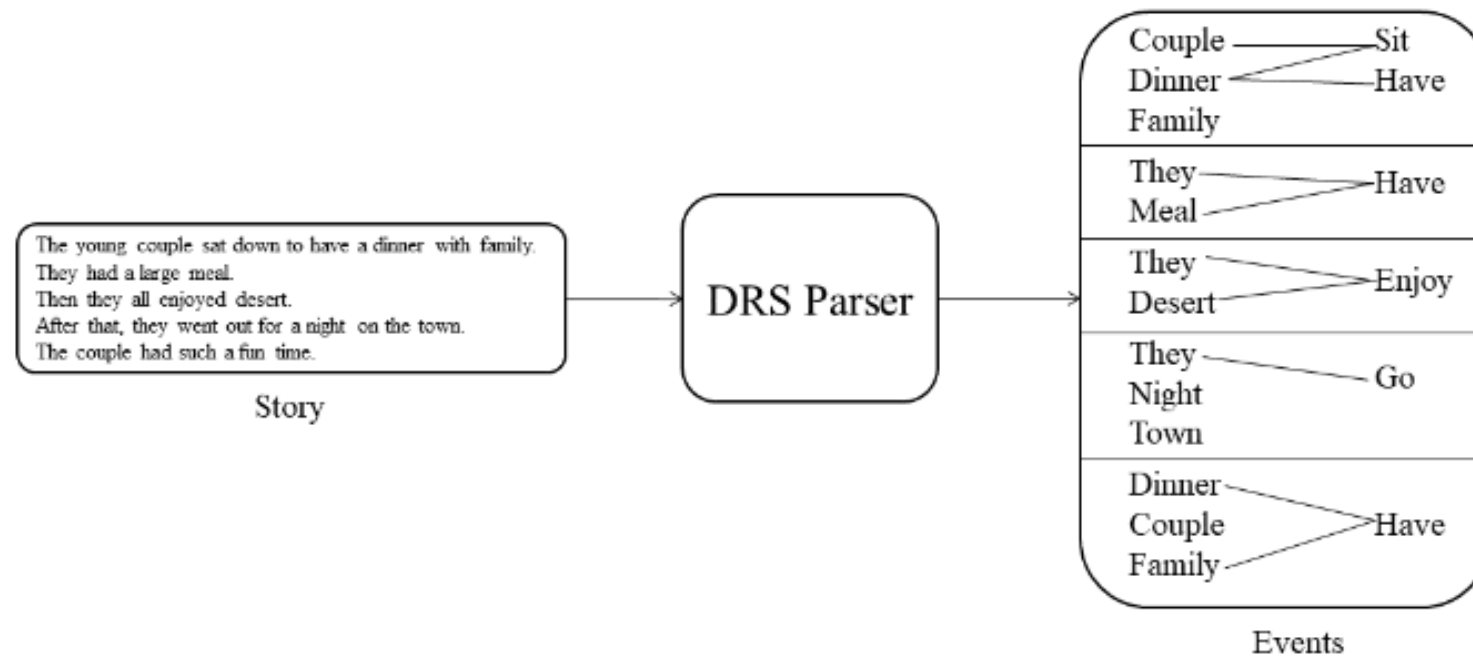
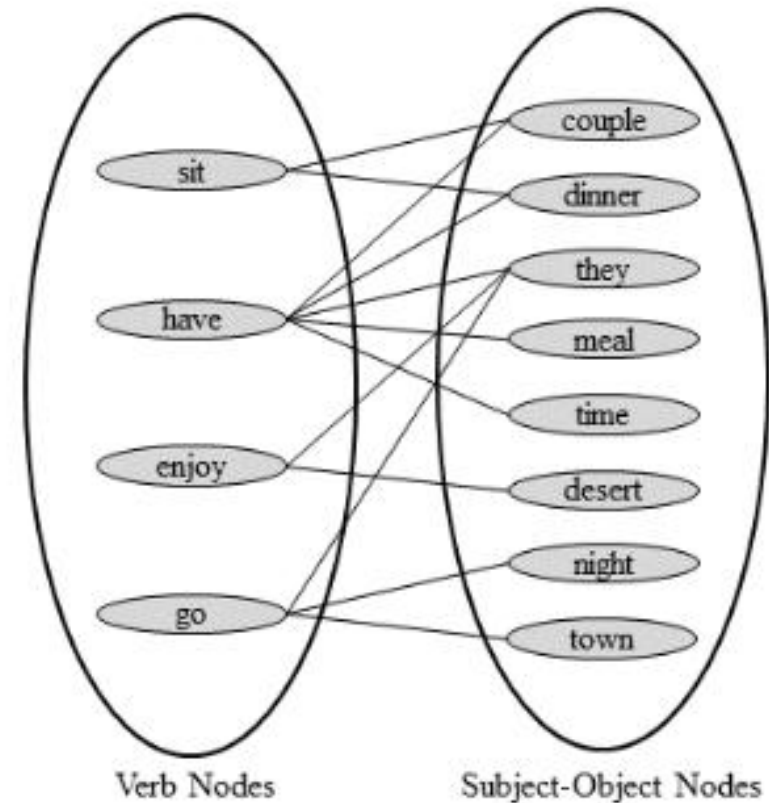


Figure 1: Event Representation using DRS Parser. Note that the words without edges are removed while forming the graphs.

2) Create Story Graphs from Semantic Relationships

Story Graph: “edges [...] between story verb nodes and story subject-object nodes.”

Gandhi, S., & Harrison, B. (2019). Guided open story generation using probabilistic graphical models. *International Conference on the Foundations of Digital Games (FDG)*, 1–7. <https://doi.org/10.1145/3337722.3341871>



Instance of a Story Graph

Figure 2: Story graph example. By adding the events generated in Figure 1, we get this instance of a graph.

3) Create Goal Graph

Goal Graph:
“relationship between author goals and story events”

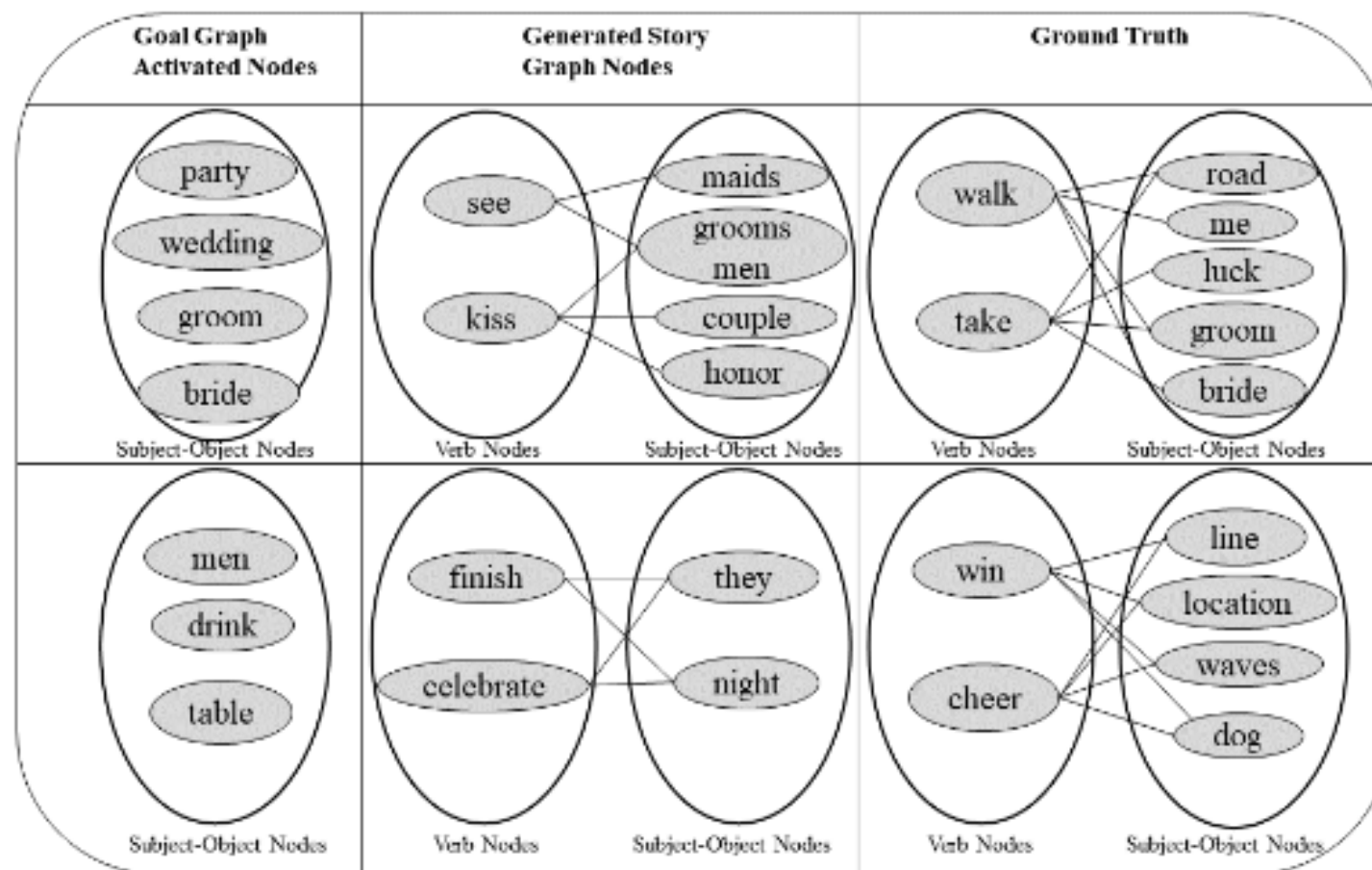


Figure 4: Evaluation Intricacies. Shown are two examples of generated action verbs and subject-action words from the given author goal (Description).

Gandhi, S., & Harrison, B. (2019). Guided open story generation using probabilistic graphical models. *International Conference on the Foundations of Digital Games (FDG)*, 1–7. <https://doi.org/10.1145/3337722.3341871>

Using VIST for Stories

Example Generated Story

1



The dog was ready to go.

2



He had a great time on the hike.

3



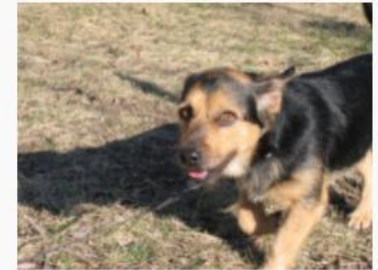
And was very happy to be in the field.

4



His mom was so proud of him.

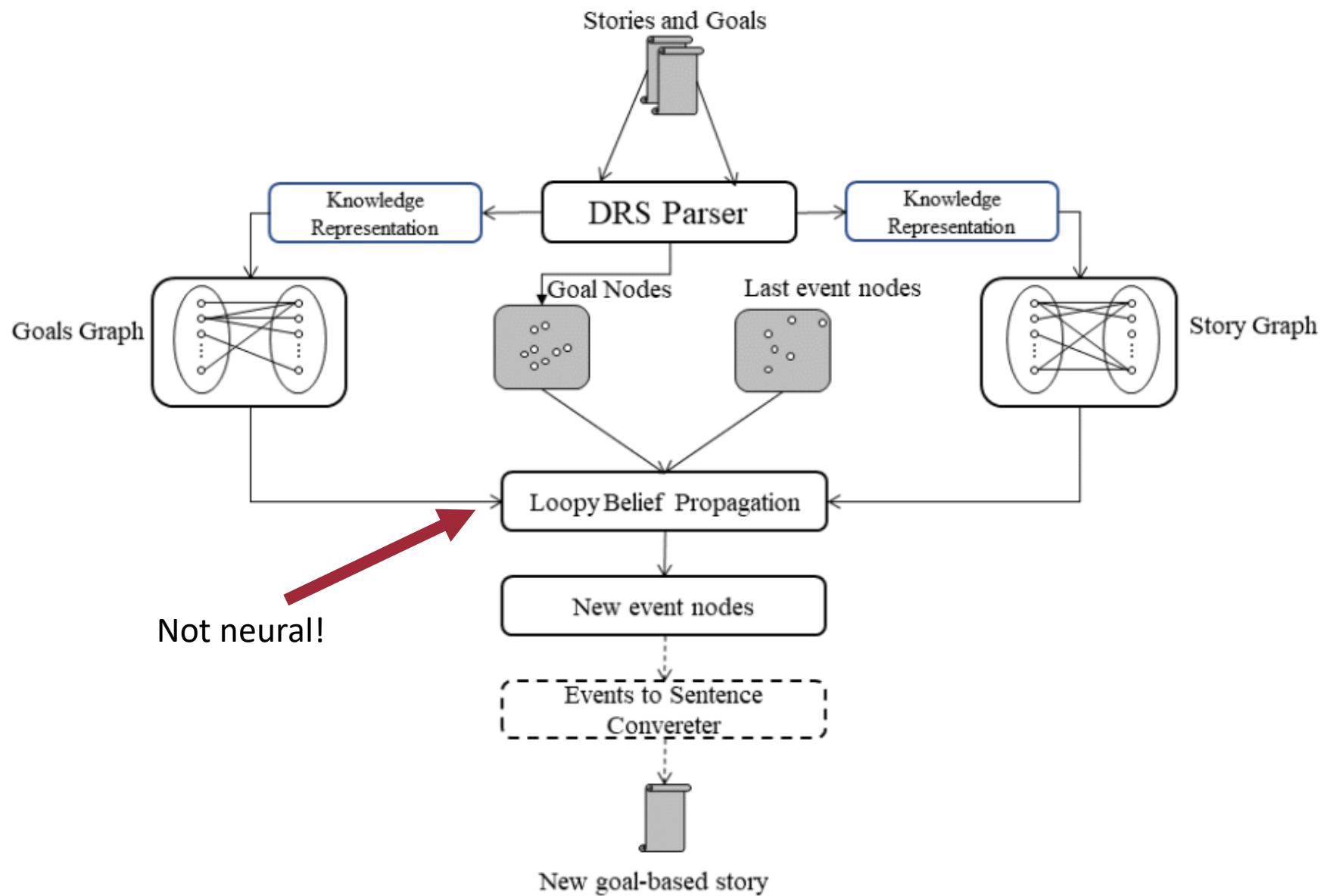
5



It was a beautiful day for him.

Photos by [kameraschwein](#) / CC BY-NC-ND 2.0

From [VIST: Visual Storytelling Dataset \(visionandlanguage.net\)](#)



Story Generation

ALGORITHM 1: Story generation algorithm

Data: *Story Graph, Goal Graph*

Result: Story *S*

for $event_i \leftarrow 1 : n$ **do**

if $CG \neq ParseCurrentGoal()$ **then**

$CG = ParseCurrentGoal()$

end

$Initial\ SVN = LBPI nfer(StoryGraph, SSON_{i-1})$

$SVN_i = LBPI nfer(GoalGraph, Initial\ SVN, CG)$

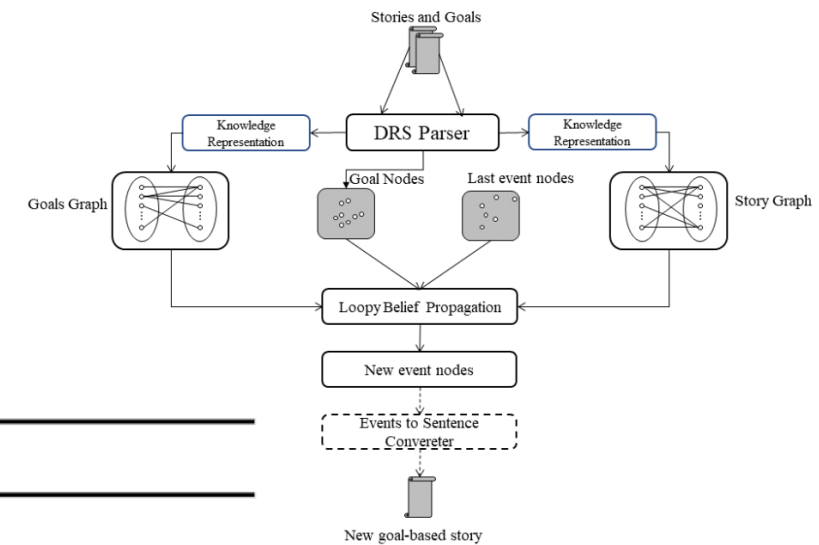
$SSON_i = LBPI nfer(StoryGraph, SVN_i)$

$S+ = GetEvent(SVN_i, SSON_i)$

end

Subject-Verb

Subject-Object



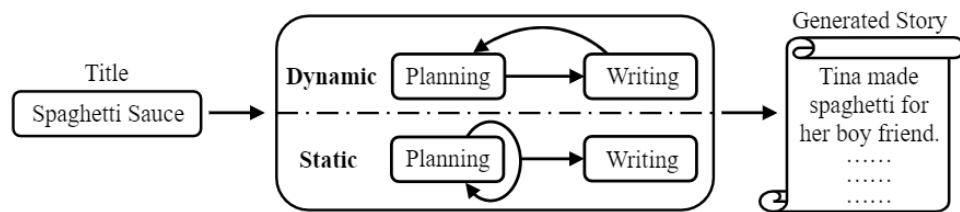
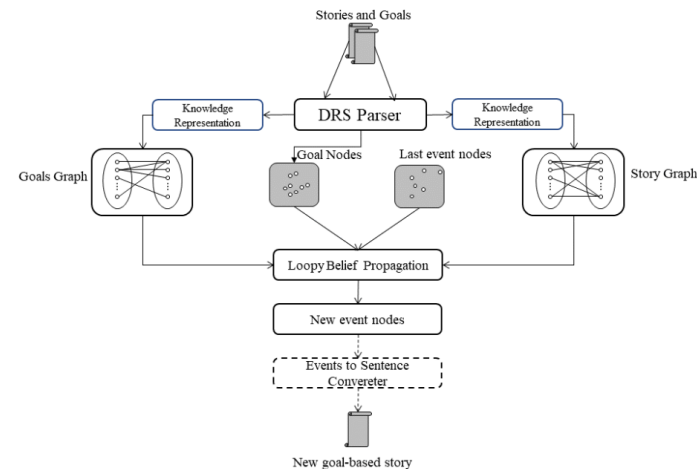
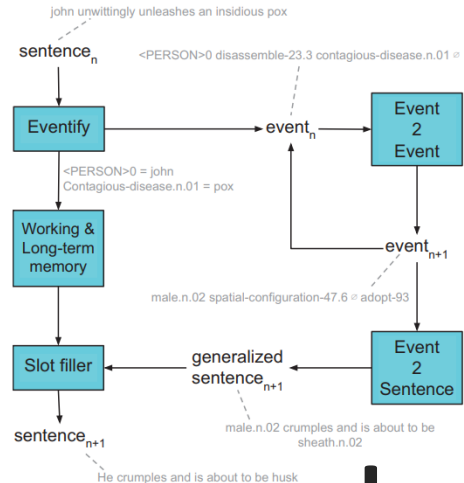


Figure 1: An overview of our system.



How do these three systems compare?

Similarities between the systems

Generate stories

Iterative architecture for generation

- Plan and write (or plan then write)

They all use events

Differences between the systems

Differences in when/how to plan

Different event architectures

Different inputs & outputs

Ways of Extracting Plot Points

Most salient keywords

Event representations

Verb-Noun Sets

Generating with Plot Points

Co-generated vs conditioned (prompted) with plot

Generate event & then translate to natural language

Graph algorithms (Loopy Belief Propagation)

Expanding the Methods

We saw how Plan & Write has evolved.
How might some of these other techniques differ when using transformers (if at all)?

Guided Open Story Generation Using Probabilistic Graphical Models

Sagar Gandhi*
University of Kentucky
Lexington, Kentucky
sga267@uky.edu

Brent Harrison
University of Kentucky
Lexington, Kentucky
harrison@cs.uky.edu

ABSTRACT

In this work, we present an approach for performing computational storytelling in open domain based on Author Goals. Author Goals are constraints placed on a story event directed by the author of the system. There are two challenges present in this type of story generation: (1) automatically acquiring a model of story progression, and (2) guiding the progress of story progression in light of different goals. We propose a novel approach to story generation based on probabilistic graphical models and Loopy Belief Propagation (LBP) that addresses both of these problems. We show the applicability of our technique through a case study on the Visual Storytelling (VIST) 2017 dataset. We use image descriptions as author goals. This empirical analysis suggests that our approach is able to utilize goals information to better automatically generate stories.

CCS CONCEPTS

• Computing methodologies → Probabilistic reasoning; Discourse, dialogue and pragmatics; Natural language generation; Learning

problem *domain model* [3, 5, 25]. This type of story generation is sometimes called *Closed Story Generation*. While stories generated in this way are often coherent, the space of possible stories that can be generated is tightly coupled with the domain model. In order to generate different types of stories, a new domain model must be authored, which is often a time consuming task.

Open Story Generation seeks to address this limitation by enabling stories to be told in any conceivable domain through the use of machine learning. The goal is to use these systems to tell a variety of stories without the need for re-learning specific domain models.

One limitation of open story generation systems is that, due to the complexity of the models used for training, it can be difficult to encode specific author goals into the story generation process. To address this limitation of open story generation systems, we propose a new framework that can reason about the overall structure of a story as well as how to incorporate author goals into the story generation process. This system enables many different types of stories to be generated based on these author goals without needing to retrain or re-learn a domain model.

The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Story Realization: Expanding Plot Events into Sentences

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung,
Zhaochen Luo, William Ma, Lara J. Martin, Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
{raj.ammanabrolu, etien, wcheung8, zluo, wma61, ljmartin, riedl}@gatech.edu

Abstract

Neural network based approaches to automated story plot generation attempt to learn how to generate novel plots from a corpus of natural language plot summaries. Prior work has shown that a semantic abstraction of sentences called *events* improves neural plot generation and allows one to decompose the problem into: (1) the generation of a sequence of events (event-to-event) and (2) the transformation of these events into natural language sentences (event-to-sentence). However, typical neural language generation approaches to event-to-sentence can ignore the event details and produce grammatically-correct but semantically-unrelated sentences. We present an ensemble-based model that generates natural language guided by events. We provide results—including a human subjects study—for a full end-to-end automated story generation system showing that our method generates more coherent and plausible stories than baseline approaches¹.

1 Introduction

Automated story plot generation is the problem of creating a sequence of main plot points for a story in a given domain. Generated plots must remain consistent across the entire story, preserve long-term dependencies, and make

for explicit domain modeling beyond providing a corpus of example stories. The primary pitfall of neural language model approaches for story generation is that the space of stories that can be generated is huge, which in turn, implies that, in a textual story corpora, any given sentence will likely only be seen once.

Martin et al. (2018) propose the use of a semantic abstraction called an *event*, reducing the sparsity in a dataset that comes from an abundance of unique sentences. They define an event to be a unit of a story that creates a change in the story world's state. Technically, an event is a tuple containing a subject, verb, direct object, and some additional disambiguation token(s).

The event representation enables the decomposition of the plot generation task into two sub-problems: *event-to-event* and *event-to-sentence*. Event-to-event is broadly the problem of generating the sequence of events that together comprise a plot. Models used to address this problem are also responsible for maintaining plot coherence and consistency. Once new events are generated, however, they are still not human-readable. Thus the second sub-problem, event-to-sentence, focuses on transforming these events into natural language sentences.