Creative Storytelling with Language Models and Knowledge Graphs

Xinran Yang[a], Ilaria Tiddi[a]

[a]*Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands*

Abstract

Automated story generation is a popular and well-recognized task in the field of natural language processing. The emergence of pre-trained language models based on large Transformer architectures shows the great capability of text generation. However, language models are limited when the generation requires explicit clues within the context. In this research, we study how to combine knowledge graphs with language models, and build a creative story generation system named DICE. DICE uses external knowledge graphs to provide context clues and implicit knowledge to generate coherent and creative stories. The evaluation shows that our approach can effectively inject the knowledge from knowledge graphs into the stories automatically generated by the language model.

Keywords

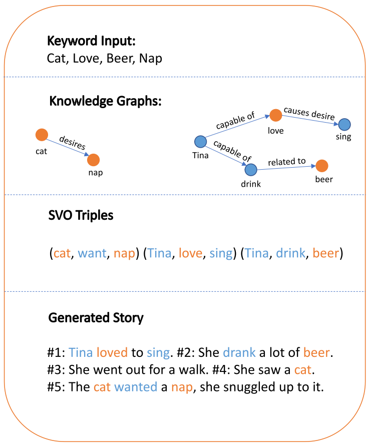
knowledge graph, language model, story generation, natural language generation

1. Introduction

Story generation is a challenging task that requires reasonable and relevant content in the generated sentences as well as dealing with logic and implicit information (Guan et al. 2019). After large-scale pre-trained language modes like OpenAI GPT-2 (Radford et al. 2019) and BERT (Devlin et al. 2018) have been released in recent years, machines have shown the ability to generate a paragraph of understandable text according to a given topic. These language models are able to generate mostly-grammatical sentences with nearly perfect syntax and punctuation (Koncel-Kedziorsk et al. 2019). However, the text generated by these language models often lacks commonsense knowledge (Logan et al. 2019) and it is hard to control the content of the automatically generated text. To solve the problem, one solution is to take advantage of structured inputs, such as tabular inputs and knowledge graphs (Koncel-Kedziorski et al. 2019). Meanwhile, one of the most popular methods to combine language models and knowledge graphs, is using knowledge graph embeddings. However, creating embeddings for knowledge graphs is a complex and time-consuming process; moreover, knowledge graphs tend to be often updated, and new embeddings have to be created (Wu et al. 2019). This research introduces a new method to combine knowledge graphs with language models without embedding approaches.

We aim to answer the following research questions: **Q1.** How to combine the language model with knowledge graphs for the story generation without knowledge graph embeddings? **Q2.** What are the advantages and disadvantages of using knowledge graphs to automatically generate a story?

We propose a two-layer system called DICE, which contains a knowledge enrichment layer and a text generation layer, applying the knowledge graph and the language model respectively, to generate coherent and



Keyword Input:
Cat, Love, Beer, Nap

Knowledge Graphs:

SVO Triples

(cat, want, nap) (Tina, love, sing) (Tina, drink, beer)

Generated Story

#1: Tina loved to sing. #2: She drank a lot of beer.
#3: She went out for a walk. #4: She saw a cat.
#5: The cat wanted a nap, she snuggled up to it.

Figure 1: An example of the story generation. The orange words are the keywords provided by the user, and the blue words are the extended entities and relations from the DICE knowledge graph. These words are connected as knowledge graphs (SVO triples). "#i" indicates the sentence is the i-th sentence of the story.

Proceedings of the CIKM 2020 Workshops, October 19-20, 2020, Galway, Ireland

EMAIL: x6.yang@student.vu.nl (X. Yang); i.tiddi@vu.nl (I. Tiddi)
URL: https://kmitd.github.io/ilaria/ (I. Tiddi)
ORCID: 0000-0001-7116-9338 (I. Tiddi)

creative stories[1]. Figure 1 presents an example of the story generation process. In the example, the system takes 4 keywords as an input, then enriches the keywords with the knowledge graph and constructs subject-verb-object (SVO) triples, the latter will be used as a prompt for the language model to generate stories.

The current work explores the possibility of connecting the knowledge graph and the language model with an interface. The advantage of using an interface is that the language model can rapidly adapt to the changes from an updated knowledge graph. For the knowledge enrichment layer, we implemented two versions of DICE knowledge graph. For version1, we retrieve the knowledge from ConceptNet (Speer et al. 2017) and WordNet (Miller 1995) and construct an integrated knowledge graph of commonsense knowledge for the story generation. For version2, we enriched the knowledge graph of version1 by using DBpedia facts. For the text generation layer, we choose ROCStories[2] (Mostafazadeh et al. 2016) as our story corpus to fine-tune the language model, GPT-2. More details are discussed in Section 3.

The contributions are as follows:

- We propose a new way of combining knowledge graphs and language models for text generation without using knowledge graph embeddings. The results show that we can effectively inject the knowledge from knowledge graphs into the automatically generated stories as a background or a plot and therefore control the content of these stories to some extent.
- We introduce a fine-tuned model which accepts SVO triples as a prompt instead of sentences used by original GPT-2 models, to generate reasonable and creative stories with the context provided by the SVO triples.

2. Related Work

2.1. Text Generation using Language Models

Story generation is a knowledge-intensive process (Li et al. 2013). In particular, open story generation requires artificial intelligence systems to create narratives about any topic without a pre-defined domain model (Li et al. 2013). Meanwhile, a creative story should be both novel and appropriate (Sternberg 1999). Existing natural language generation systems are often limited when the tasks require higher levels of creativity and originality (Jain et al. 2017). Pre-trained language models based on large Transformer architectures (Vaswani et al. 2017), such as GPT-2 and BERT, can be a potential solution for this problem. Recently, the OpenAI team has announced the upgraded GPT-3 (Brown et al. 2020) with 175 billion parameters which is 100 times larger than the previous version, GPT-2. These language models show impressive text generation capabilities that can achieve state-of-the-art results without extra training (Keskar et al. 2019). However, these language models perform poorly when capturing the long tail of rare entities such as numbers and dates (Logan et al. 2019). Moreover, these models are unable to build context clues and use implicit knowledge to generate a reasonable story ending (Guan et al. 2019).

2.2. Text Generation with Knowledge Graph Embeddings

The problem mentioned above can be improved by combining language models with knowledge graphs, where the former can facilitate the knowledge extracted from knowledge graphs. For example, Logan et al. (2019) built the knowledge graph language model (KGLM) that could select and copy related facts from a knowledge graph. Ostendorff et al. (2019) enriched BERT with knowledge graph embeddings for document classification and got better results than the standard BERT approach. Meanwhile, Koncel-Kedziorski et al. (2019) introduced a new attention model for graph encoding and used it for the graph-to-text generation. The main shortcoming of these models is their high cost of computational resources which leads to a long training and task execution time (Yao et al. 2019). Koncel-Kedziorski et al. (2019) also showed that their proposed model failed to mention 40% of entities in the knowledge graphs in the generated text.

2.3. Knowledge Enrichment with Knowledge Graphs

Hsu et al. (2019) proposed the distill-enrich-generate framework that using knowledge graphs to enrich the words distilled from the input images and then generating stories. Liu et al. (2019) used external knowledge graphs to enrich the input sentence as a sentence tree for solving NLP tasks such as classification and sequence labeling. Guo et al. (2019) built a poetry knowledge graph for keyword mapping, extension, and selection to generate Chinese classical poems with high

[1]Data and code available at github.com/ranyxr/dice_story
[2]https://cs.rochester.edu/nlp/rocstories/

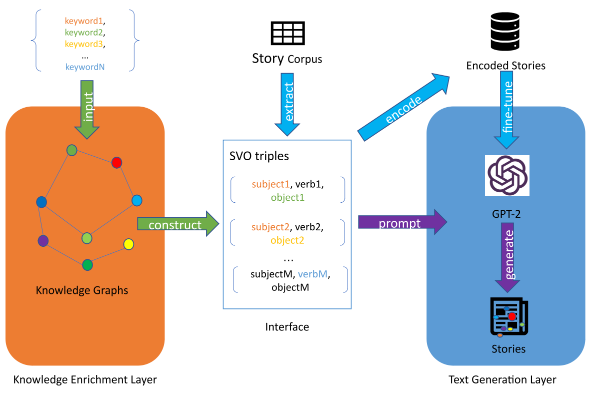


Figure 2: Two-layer architecture of DICE system. The green arrows indicate the workflow of the knowledge enrichment process. The purple arrows indicate the workflow of the text generation process. The blue arrows indicate the workflow of the language fine-tuning process.

quality and relevance. Similarly, Zhou et al. (2020) resort to a knowledge graph that consists of a collection of head-relation-tail triples to retrieve related topics in their intelligent dialogue system.
Different from some researches above, instead of delivering a graph-to-text task which emphasizes the explicit translation from graph to text without creative writing, this study puts more focus on using information from knowledge graphs to provide a background or a plot for the language model as guidance or inspiration.

3. Method

3.1. Overview

The task here is to generate 5-sentence stories from a set of SVO triples that are extracted and regrouped in a knowledge graph. The expected input of the system is a set of keywords provided by users. Figure 2 shows the two-layer architecture of the DICE system. We use SVO triples as an interface to connect the knowledge enrichment layer and the text generation layer. The SVO triples can be constructed from knowledge graphs or extracted from story corpus; meanwhile, they serve as a prompt for the language model to generate stories. The system firstly checks the relationships between these keywords and adds additional information using the knowledge graph, then generates a set of SVO triples to feed the language model to generate stories.

Two processes are involved to complete this task: language model fine-tuning and story generation. Language model fine-tuning is the pre-processing step for story generation, which includes two stages: SVO triple extraction and fine-tuning. Story generation also has two stages, i.e., knowledge enrichment and text generation. In the next section, we will discuss each stage in detail.

3.2. Language Model Fine-tuning

The OpenAI team has released GPT-3, the GPT-2's successor, but it was not available when we conducted the research. As a result, we choose GPT-2 as the natural language generator. OpenAI has released 4 versions of GPT-2[3]: the small version with 124M parameters, the medium version with 355M parameters, the large version with 774M parameters, and the XL version with 1.5B parameters. Considering the large amount of training data (the encoded story corpus is 19M), we choose the medium version of GPT-2 to strike the balance of speed, size, and creativity. An open-source Python package, gpt-2-simple[4], is used to support the fine-tuning and text generation process. Meanwhile, we choose the ROCStories as our story corpus, which contains nearly 10 thousand short stories, each story includes a title and five-sentence content.

3.2.1. SVO Triple Extraction

After acquiring the story corpus, we need to encode the dataset into a format that allows GPT-2 to generate text according to the specified SVO triples. We extract SVO triples from each story, then add the triples as a prefix for each story respectively. This way, the language model can learn from a hint that each story is generated conditionally on the SVO triples.

We use spaCy[5] to extract SVO triples from each story as "entities and relations". However, sometimes the process may encounter the coreference problem, i.e., a pronoun is used as a subject. For example, the sentence is "My sister has a dog. She loves him.", the triple directly extracted by spaCy is *(My sister, has, dog)* and *(She, loves, him)*, which are not the expected result because we want a more specific reference as a subject, i.e., *(My sister, has, dog)* and *(My sister, loves, dog)*. The resolution is using neuralcoref[6] that applies the neural net scoring model to find coreferences in the text (Clark & Manning 2016). Meanwhile, to simplify the triple, we convert the verb into its lemma and only extract the main text of the subject and the object. For

[3]https://openai.com/blog/gpt-2-1-5b-release/
[4]https://github.com/minimaxir/gpt-2-simple
[5]https://spacy.io/
[6]https://spacy.io/universe/project/neuralcoref

the example above, we extract "sister" instead of "my sister", "love" instead of "loves".
As a result, one example from the encoded dataset is the following:

> (Joseph, sign, deal), (Joseph, be, musician), (Joseph, be, songwriter), (Joseph, hope, write), (Joseph, lose, wallet), (woman, contact, Joseph), (Joseph, have, idea) The Best Single Joseph has just recently signed a deal with a new record label. He is a musician and a songwriter who hopes to write a best new hit. On his way to a local coffee shop to brainstorm, he lost his wallet. Joseph was frustrated until a woman contacted him and returned it. Suddenly, he realized he had an idea for his new song about kindness.

Words in red are the SVO triples; words in orange are the story title; words in blue are the story content.

3.2.2. Fine-tuning

The last step of this process is to fine-tune the model based on the encoded dataset, which includes both SVO triples and the original ROCStories. However, language models like GPT-2 are built for longform content, generating short text like 5-sentence stories is not the typical generation scenario. To workaround this issue, we use GPT-2-simple, which allows us to add flags to indicate where is the start and the end of each short text (5-sentence story in this case), then the language model will automatically extract the shortform texts during the fine-tuning process.

The final fine-tuned model is called the DICE model, which can be found and downloaded on Google Drive[7].

3.3. Story Generation

We use the SVO triples as a prompt for GPT-2. The triples are constructed based on the keywords by using knowledge graphs . Each triple includes a subject and an object as its entities and a verb as its relation that connects the entities. The SVO triples can not only give the language model topics (entities) to talk about but also define part of the plots (relations) of the story. For example, *(Jane, be, singer)* defines the background of the story, where there is a person whose name is Jane who is a singer. The story generation includes two stages: knowledge enrichment and text generation.

[7]https://drive.google.com/drive/folders/1T68rWkOde5ZwcuodQ9iWuYJAcqAmb0Jo

3.3.1. Knowledge Enrichment

The system includes a new knowledge graph dataset named *DICE KG*. We implemented two versions of DICE KG. Version1 (CW, i.e., ConceptNet and WordNet) combines two large open-source knowledge graphs: ConceptNet 5.6.0 and WordNet. ConceptNet is a knowledge graph that connects words and terms (phrases of natural language) with assertions (labeled, weighted edges) (Speer et al. 2017). Unlike ConceptNet, WordNet is a large lexical database of English with cognitive synonyms (synsets), which are connected by means of conceptual-semantic and lexical relations (Miller 1995). The DICE KG converts these two datasets into an integrated model, and as a result, the dataset contains more than 1.6 million nodes and over 3 million relationships with 54 types. The DICE KG is large enough for finding relations between the keywords given by users and constructing a set of SVO triples using the entities and relations in the knowledge graph. Moreover, each relationship between the words has an annotation named "weight", which can help the system to find a more reasonable path in the next step, i.e., the SVO triple construction.

We also introduce another version (DBCW, i.e., DBpedia, ConceptNet, and WordNet) of DICE KG that enriches version1 with DBpedia's mappings[8]. The DBCW version includes over 8.5 million nodes with 6 labels and over 23 million relationships with 694 types. In this version, we enrich the common concepts from ConceptNet and WordNet with factual instances and properties from DBpedia. In Section 4, we compare the performance of the two versions of DICE KG.

To construct SVO triples from the given keywords, there are 3 steps: internal matching, external enrichment, and converting paths to triples. The internal matching concerns finding meaningful relations between the keywords, so that we can later put the keywords at the corresponding position in an SVO triple. If there is a keyword that has no relation with other keywords, we use external enrichment to assign other related words in the knowledge graph to construct an SVO triple for the keyword. The first two steps are both semi-automatic, i.e., we use Cypher to query the graph database and get the matching candidates while manually filtering the matching results, which are still needed to ensure the quality of the SVO triples.

Figure 3 shows an example of the SVO triple construction. We assume that the keywords are: *{love, cat, beer, nap}*. Firstly, we try to lookup the one-hop relationship (only specific relations are considered, such as

[8]https://databus.dbpedia.org/dbpedia/mappings/mappingbased-objects/2020.07.01

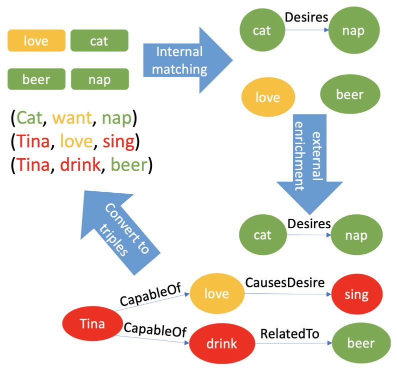


Figure 3: An example of SVO triple construction. Words in yellow are verbs. Words in green are nouns. Words in red are the enriched words from knowledge graphs.

CapableOf and Desires) between the keywords in the knowledge graph. In this case, we find one direct relation: *(cat, desires, nap)*. Next, we assign additional information to the keywords without a direct relation. In this case, for the verb “love”, we randomly choose the word “sing” as the verb’s object, which is connected to “love” through a relation called “CausesDesire”. Meanwhile, we choose “Tina”, which belongs to the person class, as the verb’s subject. For “beer” which is a noun, we assign it a verb “drink”, which is related to “beer”, and we also choose “Tina” as its subject to keep the story simple. Finally, we also need to map the directly one-hop relation into a more common word, for example, *(cat, desires, nap)* becomes *(cat, want, nap)*. As a result, the final SVO triples are *(cat, want, nap), (Tina, love, sing), and (Tina, drink, beer).*

3.3.2. Text Generation

The final stage is the text generation. After we get the SVO triples, we can use these as a prefix to generate stories from the trained model. In this process, we use GPT-2 as the story generator. Meanwhile, we use gpt-2-simple which allows for prefixes to force the generated text to start with the prefix and generate stories from these triples. Finally, we truncate the prefix and flags in the generated stories, to return text only with titles and contents. Table 1 shows one example generated by DICE using the triples mentioned above. These stories are handpicked from 75 automatedly generated stories. We can see the stories can exactly reflect the entities and relations from the SVO triples in generated stories, although the triples may not be presented in the stories 100% of the time.

Table 1
A story generated by DICE. Words in red are the subjects from the SVO triples; words in orange are the verbs (relations) from the SVO triples; words in blue are the objects from the SVO triples.

Title	Content
Lazy Cat	Tina loved to sing and drink beer with her friends. One day she was drunk and didn’t know what to do. She decided to go to the bar and see what she could do. She drank some beer and then went home. She went to sleep and woke up to her cat’s snoring.

4. Experiments

4.1. Baselines

DICE (CW) vs. Human. A given keyword set will be provided to both a human and the DICE system (with CW version of knowledge graph) to create stories, then we compare the results of human-written stories and machine-written stories.

DICE (CW) vs. GPT-2. For the original GPT-2 model, we construct one or two sentences containing all the entities in the keyword set, and we use these sentences as input for the GPT-2 model which is directly fine-tuned on ROCStories to generate a story. We then use the same keyword set to generate stories using the DICE model and compare the results.

DICE (CW) vs. GPT-2-keyword-generation. GPT-2-keyword-generation[9] is open-source software that using GPT-2 to generate text pertaining to the specified keywords. We compare the stories directly generated from a set of keywords with the stories generated by the DICE system.

DICE-CW vs. DICE-DBCW. We also compare the performance of the DICE system when using different versions of DICE KG to evaluate whether factual knowledge graphs can contribute to the story generation.

4.2. Evaluation

4.2.1. Evaluation Metrics

The evaluation focuses on two aspects of the generated output: story-independent metrics and story-dependent metrics (Roemmele et al. 2017). Story-independent metrics, including grammatical correctness, clarity, and engagement, will be used to analyze the quality of the generated output without considering its context; whereas

[9]github.com/minimaxir/gpt-2-keyword-generation

Table 2
Explanations and approaches for each metric. Metrics in orange are story-independent metrics. Metrics in blue are story-dependent metrics.

Metrics	Explanation	Evaluation approach
grammatical correctness	The correctness of spelling, grammar and punctuation	Automatic
clarity	Whether the text is easy to understand.	Automatic
engagement	Whether the writing style is interesting and effective.	Automatic
creativity	Whether the stories are creative or not.	Manual
coherence	Semantically coherent of the output.	Manual
Keyword coverage	To what extent do the keywords are presented in the generated text.	Automatic

story-dependent metrics, including coherence, keyword coverage, and creativity, will be used to evaluate the generated stories with reference to the context (Roemmele et al. 2017). On the other hand, the evaluation combines both automatic evaluation and manual evaluation. Explanation of each metric and the evaluation approaches are shown in Table 2.

Automatic Evaluation. For story-independent metrics, we used the automated analysis tool, Grammarly, to evaluate the overall grammaticality performance of the generated text. For keyword coverage, we used a script to monitor to what extent do the keywords were presented in the generated stories.

Manual Evaluation. Stories should be reasonable and coherent with the context (Guan et al. 2019), which is hard to access by automatic tools. As a result, a manual evaluation was also performed to more accurately evaluate the quality of each story. We invited 3 individuals to score the stories from each model, including stories from the original ROCStories. We applied 5-point Likert scales to rate each story on its creativity and coherence. Then we calculated the overall average score for each model.

Furthermore, we used a questionnaire[10] to investigate whether readers could tell the difference between the automatically generated stories and the human-written ones. We handpicked two stories generated by the DICE system where the stories were generated based on a given keyword set. Then we invited a person (non-native English speaker but with professional working proficiency) to write two stories with the same keywords. For human-written stories, each story should only contain 5 sentences and every keyword in the keyword set must be mentioned in the story content. Finally, we invited people to estimate whether the story is written by a human or a machine and score each story on its creativity and coherence.

5. Results and Discussion

5.1. Experiment Results

We picked 100 random samples for each model to evaluate their performance. We gathered the automatic evaluation results and manual evaluation results and separated them by story-independent metrics and story-independent metrics, which were shown in Table 3 and Table 4 respectively. The result shows there is no much difference according to the story-independent metrics among the stories written by the language models and human-written stories. The overall grammaticality performance of each model is satisfactory. The Grammarly overall score of the fine-tuned GPT-2 model is even higher than the score of human-written stories. For samples from ROCStories, most of the grammatical errors are the punctuation misuse. While for the stories generated by language models, the biggest writing issue is the determiner (a/an/the/this, etc.) misuse, followed by punctuation misuse and wordy sentences.

For the two story-dependent metrics of creativity and coherence, all the models perform poorly compared with human writers. In general, the generated stories are not always logical and making sense, even with a properly trained model. The OpenAI team shows that it takes a few tries to get a good and reasonable result, and meanwhile, the number of tries is highly dependent on the topics presented in the training data. Particularly, in this case, the given keywords can influence the performance of the result significantly. For example, if the given keywords are barely related to each other, then the model can perform poorly. This is because unrelated keywords make it more difficult to generate related SVO triples, and unrelated SVO triples lead to unconnected sentences in the generated stories. However, the keyword coverage of the DICE system (96% for DICE-CW and 97% for DICE-DBCW) is significantly higher than other baselines (73% for GPT-2, 88% for GPT-keyword-generation). However, for the DICE-DBCW, the coverage of the enriched words (80%) from DBpedia is lower compared with the keyword coverage. This is because some of the enriched

[10]https://forms.gle/jEu1LohH5zkADiNt6

words are proper nouns, like brand names, which are hardly shown in the training text.

Table 3
Results of the story-independent metrics.

Model	Correctness	Clarity	Engage	Score
GPT-2	21 alerts/ 4276 words	Very clear	Engaging	82/100
GPT-keyword-generation	26 alerts/ 3512 words	Mostly clear	Bland	78/100
DICE-CW	18 alerts/ 3931 words	Mostly clear	Bland	75/100
DICE-DBCW	31 alerts/ 5279 words	very clear	A bit bland	80/100
Human	54 alerts/ 4591 words	Very clear	A bit bland	80/100

Table 4
Results of story-dependent metrics.

Model	Creativity	Coherence	Keyword coverage
GPT-2	2.3/5	2.4/5	0.7275
GPT-keyword-generation	2.4/5	2.7/5	0.88
DICE-CW	2.2/5	2.5/5	0.9625
DICE-DBCW	2.3/5	2.7/5	0.9725
Human	3.7/5	4.9/5	N/A

5.1.1. Questionnaire Results

The questionnaire has received 54 responses. Most of the respondents are native English speakers (4/5 of the respondents), and some of them are non-native speakers (1/5 of the respondents) but with effective English proficiency. The result is shown in Table 5. In general, there is a great chance (37.5% on average) for people to make a mistake when judging whether the story is written by a human or a machine. In particular, stories with short sentences and wrong word choices are more likely to be regarded as a machine-written story. On the other hand, for stories that are interesting and creative but without coherence between the sentences, people are more likely to make a mistake and think the stories are written by a human.

Table 5
Result of the questionnaire.

Story No.	Written by	Average Score rate by human	Vote for Machine	Vote for Human
Story1	Human	2.70	70.4%	29.6%
Story2	Machine	2.80	70.4%	29.6%
Story3	Human	3.37	31.5%	68.5%
Story4	Machine	2.74	81.5%	18.5%

5.2. Injecting Relations into Stories

As mentioned in the last section, the keyword coverage (96%) and the relation coverage (100%) of the DICE system are very high during the test. This means the SVO triples can effectively affect the plots of the generated stories. During the experiment, we find that we can use SVO triples to inject entities and the relations between the entities into the stories as backgrounds or plots. As a result, the quality of the SVO triples and the order of these triples can significantly affect the quality of the automatically generated stories. Since these triples are generated from the knowledge graphs, the logic and relationships behind these knowledge graphs are also important to a better story generation.

5.3. Quality of Generated Stories

As shown in Table 4, there is little difference in the creativity score and the coherence score from the baselines to the DICE model. Although with the DICE model, we are able to inject relations into the stories, the relation can only affect the logic within each sentence while it cannot influence the logic that runs through the story. This is because the SVO triples extracted during the language fine-tuning process, are extracted from each sentence separately in the stories which are loosely connected, so they cannot reflect relations like causation throughout the text. As a result, the coherence of the generated stories from the DICE model is not satisfying in general.

5.4. Commonsense vs. Factual KG

We introduce two knowledge graphs in this research. The knowledge graph used in version1 (CW) is a semantic knowledge graph where common concepts and words have many connections with each other, which is the foundation to relate keywords and construct SVO triples. While for the fact-based knowledge graphs like DBpedia, they can hardly provide connections between the common concepts, and as a result, they can hardly contribute to the triple construction process. However, with a combination of semantic knowledge graphs and factual knowledge graphs, i.e., DICE KG version2 (DBCW), we can make use of the knowledge about the instances of the concepts and the properties of the instances from factual knowledge graphs, and we can use it to enrich the entities in the triples.

6. Conclusions

In this paper we showed how to use subject-verb-object triples as a context clues input to the generative model, to connect language models and knowledge graphs for story generation. Evaluation results showed that we can effectively inject entities and relations from knowledge graphs into the generated stories. Future work will focus on improving the coherence of the generated stories and making them have smooth transitions between sentences. For example, in order to improve the performance of the internal matching process, we can classify popular words into specific classes and use ontology techniques, such as SCHACL (Knublauch & Kontokostas 2017) and OWL restrictions (McGuinness & Van Harmelen 2004), to make sure these classes can interact with each other based on specific rules.

References

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[2] Chen, J., Chen, J., & Yu, Z. (2019, July). Incorporating structured commonsense knowledge in story completion. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 6244-6251).

[3] Chen, Z., Eavani, H., Liu, Y., & Wang, W. Y. (2019). Few-shot NLG with Pre-trained Language Model. arXiv preprint arXiv:1904.09521.

[4] Clark, K., & Manning, C. D. (2016). Deep reinforcement learning for mention-ranking coreference models. arXiv preprint arXiv:1609.08667.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[6] Guan, J., Wang, Y., & Huang, M. (2019, July). Story ending generation with incremental encoding and commonsense knowledge. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 6473-6480).

[7] Guo, Z., Yi, X., Sun, M., Li, W., Yang, C., Liang, J., ... & Li, R. (2019, July). Jiuge: A Human-Machine Collaborative Chinese Classical Poetry Generation System. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 25-30).

[8] Hsu, C. C., Chen, Z. Y., Hsu, C. Y., Li, C. C., Lin, T. Y., Huang, T. H. K., & Ku, L. W. (2019). Knowledge-Enriched Visual Storytelling. arXiv preprint arXiv:1912.01496.

[9] Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A., & Sankaranarayanan, K. (2017). Story generation from sequence of independent short descriptions. arXiv preprint arXiv:1707.05501.

[10] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858.

[11] Knublauch, H., & Kontokostas, D. (2017). Shapes constraint language (SHACL). W3C Candidate Recommendation, 11(8).

[12] Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., & Hajishirzi, H. (2019). Text Generation from Knowledge Graphs with Graph Transformers. arXiv preprint arXiv:1904.02342.

[13] Li, B., Lee-Urban, S., Johnston, G., & Riedl, M. (2013, June). Story generation with crowdsourced plot graphs. In Twenty-Seventh AAAI Conference on Artificial Intelligence.

[14] Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2019). K-bert: Enabling language representation with knowledge graph. arXiv preprint arXiv:1909.07606.

[15] Logan, R., Liu, N. F., Peters, M. E., Gardner, M., & Singh, S. (2019, July). Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (pp. 5962-5971).

[16] McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, 10(10), 2004.

[17] Miller, G. A. (1995). WordNet: a lexical database for English. Communications of the ACM, 38(11), 39-41.

[18] Mostafazadeh, N., Vanderwende, L., Yih, W. T., Kohli, P., & Allen, J. (2016, August). Story cloze evaluator: Vector space representation evaluation by predicting what happens next. In Proceedings of the 1st Workshop on Evaluating Vector-Space Rep-

resentations for NLP (pp. 24-29).

[19] Ostendorff, M., Bourgonje, P., Berger, M., Moreno-Schneider, J., Rehm, G., & Gipp, B. (2019). Enriching BERT with Knowledge Graph Embeddings for Document Classification. arXiv preprint arXiv:1909.08402.

[20] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8).

[21] Roemmele, M., Gordon, A. S., & Swanson, R. (2017, August). Evaluating story generation systems using automated linguistic analyses. In SIGKDD 2017 Workshop on Machine Learning for Creativity (pp. 13-17).

[22] Speer, R., Chin, J., & Havasi, C. (2017, February). Conceptnet 5.5: An open multilingual graph of general knowledge. In Thirty-First AAAI Conference on Artificial Intelligence.

[23] Sternberg, R. J. (Ed.). (1999). Handbook of creativity. Cambridge University Press.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[25] Wu, T., Khan, A., Gao, H., & Li, C. (2019). Efficiently embedding dynamic knowledge graphs. arXiv preprint arXiv:1910.06708.

[26] Yao, L., Mao, C., & Luo, Y. (2019). KG-BERT: BERT for knowledge graph completion. arXiv preprint arXiv:1909.03193.

[27] Zhou, L., Gao, J., Li, D., & Shum, H. Y. (2020). The design and implementation of xiaoice, an empathetic social chatbot. Computational Linguistics, 46(1), 53-93.