NLP Project : event and character embedding to construct interactive story generator

Théotim Barbier

Abstract

This project takes place in the lecture of Natural Language Processing in University Degli Studi di Milano conducted by proffessor F. Alfio. The programming part of the project can be find here. This project consist in the creation of a interactive story generator, which aims to construct a consistent story in which the user is frequently asked to interact with the story creation. The approach of event abstraction and character embedding (**[2]** and **[3]**) will be explored in order to create a consistent story. You can find the github repository here : `https://github.com/theot-student/NLP-Project`

1 Introduction

Automated story generation is a Natural Language Processing (NLP) problem which consist of creating a convincing story for human. This requires to generate a coherent sequence of successives events and actions and to ensures the consistency of the story along the generation. The consistency is the maintenance of logical event along the story. For exemple, during a story, a dead character should not reappear for no reason. Nowadays, consistency is still a big problem for story generation.

For now, differents models have been studied in order to provide a good story. The first models where relieving on the use of Recurrent Neural Network (RNN) in encoder and decoder like in **make links**. Nowaday, the transformers like GPT and BERT are used to make pretty good story generation.

In this report, we will discuss to use the definition of a story as a sequence of event conducted by characters. Therefore as in **[2]** and **[3]** we will try use event caracterization and character embedding to generate a coherent and consistent story. An event can be define as a significant change in a character's life or in the world in the story. Then generate a good sequence of events is necessary to ensure a good coherent in the story. Then the character embedding is a way to caracterize the

character at a moment in the story. Then provide a good evolution of the characters caracterization ensures the story to be consistent .

To do that we will explore the use of two differents neural network. First we use a RNN for event generation mixing with the character embedding. We call this new definition of event a CEvent (character event). Then we use a BERT transformer to generate sentences from event. The flexibility of the transformers make easy to extend their use to translate event to sentences with a good linguistic

2 Related Work

The use of events to generate has already been explored in the work of **[2]** and **[4]** . They use several definition of event and two RNN for Event2Event and Event2Sentence to generate story.

In **[3]** they use the character embedding with the LSTM to provide consistents story.

3 Dataset

In this project, we use the Writting Prompt dataset from kaggle (`https://www.kaggle.com/datasets/ratthachat/writing-prompts`). This dataset contains prompts written a reddit user. Then some other reddit users have created stories related to the prompts. We use these prompts and story as examples of story to train our models.

4 Theory

4.1 Event representation

Use definition of event from Event Paper *Event Representations for Automated Story Generation with Deep Neural Nets*

We define the generation of story as choosing a sequence of event. To do so, considering we have the probability of the next event considering the previous event, we can take the maximum probability over the next event. Therefore, it consist of maximize the probability distribution over a paramater θ (which represent the parameter to train in the RNN):

$$\max_{\theta} \mathbb{P}_{\theta}(e_{t+1}/e_0, ..., e_t)$$

where e_{t+1} is the event to predict and $e_t ... e_0$ are the past event from a given story. This representation is a alternative to a classical language model. In a lack of simplicity, in this project, we will use an easier language model :

$$\mathbb{P}(e_{t+1}/e_t)$$

where we consider only the previous event to predict the next event. Then we use a RNN encoder decoder to approximate the probability distibution.

The advantage of using event instead of complete sentence make a more abstract definition of the story which enable more accuracy of the model because there are much less possible event than possible sentences.

Here we define an event as in **[2]** . An event is :

$$e =< s, v, o, m >$$

where s is the subject of the event, v is the verb, o is the object and m is the modifier. The object is the linguistic object or complement of the verb and the modifier is and indirect object which can modify the event also known sometime as the "wildcard". The object or the modifer can also be empty and then are put to the token "NoObject". To create event in a sentence, we use the spacy tokenizer which provide a lot of linguistic characterization and allows to recover the events word. Then we get the lemma of each word for more clarity. For exemple the sentence, "Deborah is an old witch" would output an event like <Deborah, be, witch, old>.

We can create several event from one sentence when there are several subject. For exemple "Antoine and Baptiste are clowns" create the two events : <Antoine, be, clown, NoObject> and <Baptiste, be, clown, NoObject>.

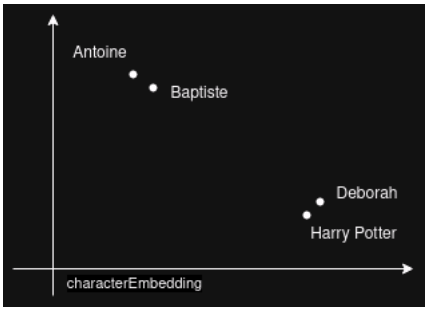
Some other improvements can be done for the event representation like in **[2]** but we will stuck to the classical one for simplicity.

4.2 Character embedding

use definition of character embedding from A Character-Centric Neural Model for Automated Story Generation

We now define our character embedding which will theoriticaly enhance the consistency of the generation. We consider that the characters evolve along the story. Then the character embedding follows the story line. Another advantage of character embedding is that different characters that have a similar representation will be encouraged to make the same acion (*figure 1*).

Figure 1: Exemple of character embedding. Baptiste and Antoine are clown and then are similar in the space of character embedding. Similarely for Deborah and Harry Potter who are witch and wizards.



For the character embedding we use as in **[5]** and **[3]** the extraction of some linguistic features related to the characters such as action verbs and adjectives. Then we get the embedding as a list of word representing the character. In the classic approach, they take this character embedding :

$$C = \frac{1}{N} \sum_{i=1}^{N} embedding(w_i)$$

with C the character embedding, w_i the word to represent the character and N the number of word that characterizes the character.

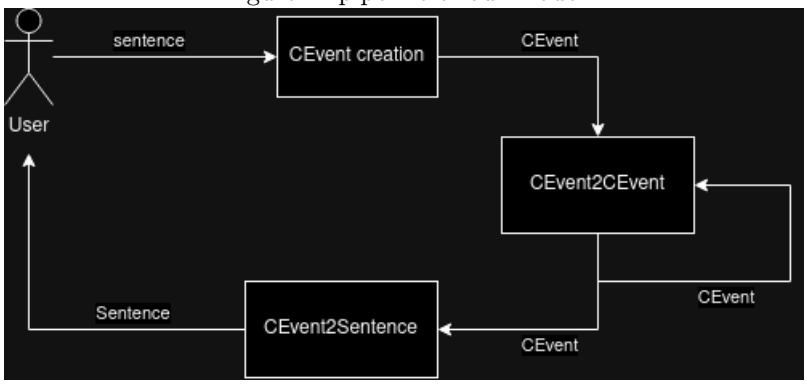
Experimentaly, we use spacy to extract the linguistic features and to recognize when a token is a character.

4.3 CEvent (mixed event and character)

In this project, we use a mix of the event abstraction and the character embedding. This would theoritically provides the avantages of logical sequence of events and the inproveness of consistency with character embedding. To do that, the first way we can think about is that, when we get a character in the event creation, we replace the lemmatized word embedding by the character embedding. Doing that, we don't have to change the classical RNN encoder decoder model, because the word embedding and the character embedding would be the same size.

Unfortunately, to use the classical RNN encoder, we need to provide one hot encoding of word and character. But this one hot encoding has the size of the vocabulary and then the RNN parameters are really huge (a matrix of the size of vocabulary $\times$ size of hidden layer) and are too difficult to train (at least on my compute).

Figure 2: pipeline of our model



Then we could think of implementing a Embedding model in order to reduce the RNN parameters size, but it would just transfer the problem in the embedding neural network.

Then there are two solutions remaining :

1. we could use an already existant word2vec algorithm to provide the embedding of the words.
2. we could use another characterization of the event with character in which we add the word representing the character in the event.

For simplicity, we choose the second solution, and then we have a CEvent like :

$$e =< s, v, o, m, c_1...c_k >$$

where $c_1...c_k$ are the word representing the character. We assume that there is at most one character participating to the event and it can only be the subject or the object of the event.

Then we get our new CEvent representation and can now train our models

train better model with this reprentation.

4.4 CEvent2Cevent and CEvent2Sequence

We recall that we have two step in our process *(figure 2)* :

1. We start with a start event and generate the next event in the story (CEvent2CEvent)
2. We then generate a sentence from the event we have generate (Cevent2Sentence)

and we redo this two step until a stopping criterion of max sentences.

To do the two steps we use differents models.

CEvent2Cevent

As said before we use a RNN encoder decoder for the CEvent2CEvent task. This would an efficient model in the case of classical CEvent embedding because the classical CEvent have always the same size of 4 embedded word. But in our case we have CEvent that can have more than 4 tokens. Then the RNN is less accurate but still works.

CEvent2Sequence

For this task we use a derivate of BERT transformer named BART. BART is a model from huggingFace (`https://huggingface.co/docs/transformers/model_doc/bart`) that has well performed in the task of text generations, especially for translation task. It is a pre-trained model, so we just need to fine tune this model to make it understand the CEvent.

5 Experiments

All the experiments can be find on the github here : `https://github.com/theot-student/NLP-Project`. They are all execute on a google collab jupyter notebook.

Since the training of the models are costly, we take a few sample of the prompts and stories from the dataset. In fact for the CEvent2CEvent task we use about of 6500 examples and for the CEvent2Sentence we use 200 examples.

CEvent2CEvent

For the CEvent2CEvent RNN we use a maxCeventToken variable of 25 which limits the number of token in a CEvent. We also use the ADAM optimiser with a learning rate of 0.001 for training.

CEvent2Sentence

Foir this task we use BART model with the BART specific BART tokenizer to process our example of CEvent and sentences. Once again, we use a maxCeventToken variable of 25 and limit the generation of a sentence to 50 tokens. We use the already implemented training method of BART model on 3 epochs.

6 Results

First of all, since the number of examples on which we conduct our experiments is too low, the results will not be so accurate. It would be better to conduct the experiment on really larger bag of example but this is not feasible on my machine. However, we will still evaluate our results and try to find a meaning to them.

To have a baseline, we also have implemented a classic event representation and models for Event2Event and Event2Sentence tasks as in **[2]**.

6.1 Evalutation metrics

First we need to define evaluation metrics to evaluate our models.

Perplexity

Perplexity is a measure of how "surprise" we are to find a n-gram in a text. Effectively, if the perplexity is large, it means that the model is generating new words or uncommon words. It is define with :

$$perplexity = 2^{-\sum_w p(w) \log_2 p(w)}$$

where w is a token in the text and p is the term frquency defined as :

$$p(w) = \frac{count(w)}{\sum_{v \in V} count(v)}$$

where V is the vocabulary of the text.

BLEU

The BLEU **[1]** metric is classic metric for translation problem. This is based of the comparaison of the output with some references that we know can be used. They are compared using a modified n-gram precision. Here, we just use the translate module of nltk library which provides an already implemented method to compute BLEU.

Human advice

I will humbly try to judge the quality of the sentence generate by the models. But this is probably not a really accurate metric to use.

Table 1: evaluation of CEvent2CEvent and Event2Event models

	BLEU Score	Perplexity
Event2Event	0.0302	1.1311
CEvent2CEvent	0.0425	1.0230

Table 2: evaluation of CEvent2Sentence and Event2Sentence models

	BLEU Score	Perplexity
Event2Sentence	0.111	2.131
CEvent2Sentence	0.064	1.3703

6.2 CEvent2CEvent result

We compare our CEvent model to the baseline of the Event model with the two metrics we have in *table 1*. Notice that for event generation task, the BLEU score is not so relevant because, the events have always the same size.

The scores show that our CEvent2CEvent model performs slightly better than the baseline model. But the gap between the two methods is not so relevant because of the number of examples used to train.

6.3 CEvent2Sentence result

We now compare our sentence generation models in *table 2*.

Once again, the scores show that our CEvent2Sentence model performs slightly better in perplexity. However the gap is still not so relevant. In this case the human advice can be use to measure coherence of the generated sentence. We see that the generated sentence are usually not linguisticaly logic and sometimes are don't even a real sentence but a sequence of symbols. This is probably due to the fact that we did not train enough the model to generate good sentences.

6.4 Adding the 2 models for story generation

We also tried to mix the two models to generate a story from a prompt. Unfortunately the generation conducts to a incoherent story which can not be considered as a good result. You can find some generate story in the code here : `https://github.com/theot-student/NLP-Project`.

7 Future research ideas

One way to improve the results is evidently to train the models with more examples. This will conduct the results to be more accurate in term of the neural network optimal generation (not in theoritical results). Another way of improvement is to use other models such as LSTM instead of RNN or GPT-4 instead of BART. This could increase the story generation but these models are costly in terms of storage and training.

We could also explore other way to define the CEvent as said before. For exemple, we could use the classic way of character embedding to substitute the character in the events.

Finally, as a story has a start and an end, we could define a start event and final event to generate consistent story that stops itself in a coherent way. This idea could also be used for our first goal, which is interactive story generator. In fact it could be interesting to ask user to provide a start event so that the generator provides a small sequence of event that stop by itself. Moreover, the character embedding could also be important to store the caracteristics of the user character along the story, in order to construct a consistent story.

A final idea would be to train directly a huge neural network model combining directly the embedding, the CEvent2Cevent and CEvent2Sentence tasks. This would probably be better to produce coherent story but such a model would be surely two costly to train.

8 Conclusion

We provided an new implementation of event named CEvent which mix the event representation of the story and the embedding of the character of the story. We then learned model to generate CEvent from CEvent and sentence from CEvent. Our results show the CEvent model is slightly better than the basic Event representation.

9 Bibliography

1. 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 on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, USA, 311–318. `https://doi.org/10.3115/1073083.1073135`

2. Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. *Event representations for automated story generation with deep neural nets.* `https://arxiv.org/abs/1706.01331`

3. Liu, D., Li, J., Yu, M.-H., Huang, Z., Liu, G., Zhao, D., & Yan, R. (2020). *A Character-Centric Neural Model for Automated Story Generation.* Proceedings of the AAAI Conference on Artificial Intelligence, 34(02), 1725-1732. https://doi.org/10.1609/aaai.v34i02.5536

4. Pichotta, K., and Mooney, R. J. 2016. *Learning Statisti- cal Scripts with LSTM Recurrent Neural Networks.* In Pro- ceedings of the Thirtieth AAAI Conference on Artificial In- telligence

5. Bamman et al., ACL 2013 *Learning Latent Personas of Film Characters* (https://aclanthology.org/P13-1035)

6. the 2024 lectures of Natural Language Processing conducted in University Degli Studi di Milano conducted by proffessor F. Alfio.

7. NLP From Scratch: Translation with a Sequence to Sequence Network and Attention from pytorch websitehttps://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html