ANLP Report

Team 39:

Bhavani Chalasani (2022101014)

Nishita Kannan (2022101112)

Archit Narwadkar (2022111032)

Introduction of the Problem

Story generation tasks must generate stories that have consistent plots, maintain consistency among characters, and maintain coherency of text units. However, current implementations of story generation struggle to capture the high-level interactions between the plot points and maintain consistent plots throughout the story. Other methods show very bad discourse coherency, so there is a need to create a model that accounts for these caveats. The proposed model consists of a two stage framework - the first stage organizes and generates a story outline to depict the story plots and events, and the second stage expands this generated outline into a complete story. This implementation also includes components to enhance discourse coherency and coreference consistency.

Selection of Baselines

As per the following papers:

- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI Blog, 1(8).
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 889–898.

we have considered the following baselines (These are also what has been stated in the main paper):

- Model - GPT2-117M
- Embedding Dimension - 768
- Number of Attention Heads - 12
- Learning Rate - 0.0005
- Dropout Rate - 0.3
- λ_1 - 0.1 (λ_1 and λ_2 are used to calculate the loss of the second transformer)
- λ_2 - 0.3
- Vocabulary - GPT2's BPE vocabulary with size of 50,527

Dataset Characteristics

Writing Prompts Dataset

- This dataset is collected from Reddit's WRITINGPROMPTS forum. There are 300k stories and the dataset is split into TRAIN, VAL and TEST (90%/5%/5%). WRITINGPROMPTS is a community where online users inspire each other to write by submitting story prompts. Each prompt can have multiple story responses. The prompts have a large diversity of topic, length, and detail.
- This dataset is the primary dataset that has been used to train and evaluate both the transformers.

- The source datasets have difference prompts, while the target datasets have the corresponding generated stories.
- The links to these datasets are here: https://paperswithcode.com/dataset/writingprompts

Book 8 Dataset

- This is a discourse marker dataset which contains 3.6M sentence pairs and each pair is labeled with one connective of 8 connectives as discourse label.
- This is used to fine tune the BERT model that we are referring as to 'Golden BERT'.
- The datasets are of the form 'sentence1, sentence2, discourse marker'. They are all tsv files.
- The link to the datasets are here:

Train - https://dissent.s3-us-west-2.amazonaws.com/data/discourse_EN_EIGHT_and_but_because_if_when_before_so_though_2017dec18_test.tsv

Validation - https://dissent.s3-us-west-2.amazonaws.com/data/discourse_EN_EIGHT_and_but_because_if_when_before_so_though_2017dec18_valid.tsv

Test - https://dissent.s3-us-west-2.amazonaws.com/data/discourse_EN_EIGHT_and_but_because_if_when_before_so_though_2017dec18_train.tsv

Approach

As per the paper, in order to account for the drawbacks of current story generation models, we approach the model to implement with a a two stage framework - the first stage organizes and generates a story outline to depict the story plots and events, and the second stage expands this generated outline into a complete story.

We include an element to enhance discourse coherency, which is the degree of connection between sentences or parts of a text that allows it to be understood as a whole. This ensures that the generated text follows a logical progression and that the meaning remains consistent throughout the discourse. We also include an element to enhance coreference consistency, which is a property of the generated text, where expressions that refer to the same entity are consistently and accurately identified and maintained throughout the text. It ensures that different terms that refer to the same thing or person are clearly and logically linked, preventing ambiguity in the generated story.

The first transformer generates an outline from a given prompt. It compares the generated prompt with a prompt created by concatenating the keywords of a story that are generated by the RAKE algorithm and an abstract generated by the TextRank algorithm. The RAKE (Rapid Automatic Keyword Abstraction) algorithm is used to extract keywords from a block of text, or a document. It extracts the top-ranking keywords/phrases as the output. The TextRank algorithm is used to extract an abstract from a block of text, or a document. The words with the highest scores are considered the most important keywords within the document, and these words are used to compose the abstract.

The second transformer takes a story prompt and an outline, and generates a coherent, comprehensive story. The outline that it takes as input is the outline that has been generated by the first transformer. The loss that is used to train the second transformer is a combination of the transformer loss, loss for discourse coherency, CoreNLP loss for coreference consistency.

For the discourse coherency model, we fine tune a pre-trained BERT model (we refer to this as golden BERT) to predict the connectors (discourse markers) between two passed sentences. We then train a model (encoder with two layer MLP) from scratch to take the generated story from the transformer, predict the connector between the sentence pairs, and generate the loss based on the marker predicted by the golden BERT model and the discourse coherency model. This loss tells us how connected and coherent the sentences in the generated story are.

For the coreference consistency loss, we generate coreference chains (a series of mentions of the same entity throughout a text) and coreference clusters (a group of mentions that are related to the same entity and can be clustered together) in order to generate a coreference loss to maximize attention weights of tokens in the same cluster.

Implementation

Link to the Repository: https://github.com/112Nisha/ANLP_Project/tree/final_submission

First Transformer - Uses prompts to generate outlines

- We create a dataloader of context-target pairs where the context is the prompt that we have loaded from the source datasets, and the target is the output of our generate_outline() function that takes the prompt from the source dataset and generates an outline by concatenating the abstract, generated by the generate_abstract() function and the keywords, generated by the extract_keywords(). The extract_keywords() function uses the Rake object from the rake_nltk library to generate the keywords, and we have considered the top 10 keywords, as mentioned in the paper. The generate_abstract() uses the summarizer from the summa library to generate abstracts. We are considering 30% sentences of each story as the abstract, as mentioned in the paper.
- We then encode the context and target pairs using the GPT2Tokenizer and pass this into our transformer, which makes use of nn.TransformerDecoder, along with positional encoding and masking functions [get_embedding_with_positional_encoding() and causal_masking() respectively]. The transformer returns both the predicted output and a list of attention weights. These attention weights are used for the coreference consistency element. The predicted output and the expected output (target) are then passed through a loss function (CrossEntropyLoss), and a loss is generated. This loss is then used to update the model, and the decoded predicted output is sent to the second transformer.

Second Transformer - Uses prompts and outlines to generate stories

- We create a dataloader of context-target pairs where the context is the concatenation of the prompt that we have loaded from the source datasets, the outline (from the first transformer), and the story (only from training, not testing), and the target is the story from the target datasets.
- We then encode the context and target pairs using the GPT2Tokenizer and pass this into our transformer, which makes use of nn.TransformerDecoder, along with positional encoding and masking functions [get_embedding_with_positional_encoding() and causal_masking() respectively]. The transformer returns both the predicted output and a list of attention weights. These attention weights are used for the coreference consistency element. The predicted output and the expected output (target) are then passed through a loss function (CrossEntropyLoss), and a loss is generated. This loss is then combined with the loss from the model for discourse coherency and the coreference loss of the generated story. The final loss is given by:

$$\mathcal{L} = \mathcal{L}_{\text{lm}} + \lambda_1 \mathcal{L}_{\text{dis}} + \lambda_2 \mathcal{L}_{\text{coref}}$$

Discourse Aware Model

- We fine tuned a pre-trained BERT model using the Book 8 dataset, in order to allow it to generate discourse markers given a sentence pair. We call this Golden BERT.
- The model we implemented is an encoder with two layer MLP and is called DiscourseAwareStoryGenerator, the training function of which generates a probability distribution for a predicted discourse marker for a given sentence pair, which it then compares with the generated output of Golden BERT for the same sentence pair. This loss is then used to update the model, and it is returned to the second transformer to help update the transformer as well. The predict_discourse_marker() function is used to call Golden BERT to generate the actual discourse marker.
- Inside the forward function of the DiscourseAwareStoryGenerator, the sentences are tokenized and encoded using the BertTokenizer.from_pretrained('bert-base-uncased') and BertModel.from_pretrained('bert-base-

uncased') and the final hidden state (from the encoder) is averaged across all tokens in the sentence, resulting in a single vector representation for the sentence. This entire tokenizing and encoding process is done in the encode_sentence function of the DiscourseAwareStoryGenerator. Once the sentence embeddings are computed, the embeddings of the first two sentences in the sequence are concatenated. The concatenated sentence representations are then passed through a linear layer, followed by a non-linear activation (tanh). The final output is produced by another linear transformation, and a softmax function is applied to generate the probability distribution over the difference discourse markers.

- The DiscourseAwareStoryGenerator is trained along with the second transformer, and it's loss is generated by using the golden BERT model as a standard to compare predicted markers to.

Coreference Consistency

- We calculate the coreference loss but first generating coreference clusters for a given text. We do this using the get_coref_clusters() function which takes the generated story and an nlp pipeline (from the stanza library), identifies the coreference chains within the story and creates a cluster for each chain, using the mentions in the chain, along with a cluster id and start and end indices for each mention and the text of each mention. It then returns a list of all the coreference clusters.
- The coreference_loss() function takes the generated story, an nlp pipeline, and the attention weight matrix as input, uses the get_coref_clusters() function to get a list of coreference function and then applies the following formula over the clusters in order to get the coreference loss, which is then returned to the second transformer:

$$\mathcal{L}_{\text{coref}} = -\frac{1}{pq} \sum_{i=1}^{pq} \frac{1}{N_i} \sum_{k=1}^{i-1} \mathbb{1}(c_k = c_i) \log \alpha_{ik}$$

where p is the number of coreference clusters, q is the number of entity mentions for each cluster. Ni is the number of entity mentions in the same cluster ci, and alpha is the attention weights matrix.

NOTE: We have implemented the transformers from both scratch and by using the gpt2 config. For the outputs, we have used a smaller model, trained on a smaller dataset for more epochs.

Analysis

We have considered four different evaluation metrics for our implementation, as per the paper:

- **Perplexity**: Perplexity is a metric to evaluate the performance of a language model. It measures how well the model predicts a sequence of words.
- **Distinct-1 and Distinct-2 scores**: Distinct-1 and Distinct-2 are evaluation metrics commonly used to evaluate the diversity of generated text. They measure how repetitive or varied the output is by calculating the ratio of unique unigrams (for distinct-1) and bigrams (for distinct-2) to the total number of unigrams and bigrams, respectively.
- **Averaged Coreference Chains**: A coreference chain consists of all the mentions in a text that refer to the same entity (the coherence of the text). Averaging the coreference chains evaluates how well a system identifies and links mentions in text that refer to the same entity.
- **Percentage of Unknown Labels**: This is the percentage of sentence pairs with unknown labels in generated stories. The less sentence pairs with unknown labels the model generates, the better the coherency of a generated story is.

The values of the metrics were taken with respect to generating the outline as the abstract and the outline as the keywords of the story.

First Transformer

Method	Perplexity	Distinct-1 (%)	Distinct-2 (%)	Unknown (%)	Coref Chains
From Keyword	293.083	0.1083	1.5276	-	-
From Abstract	209.794	0.1025	1.7136	-	-

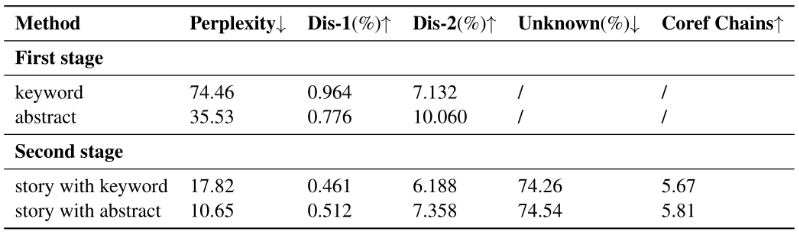
Second Transformer

Method	Perplexity	Distinct-1 (%)	Distinct-2 (%)	Coref Chains	Unknown (%)
From Keyword	720.436	0.333	0.335	0.23	50
From Abstract	651.756	0.312	0.238	0.37	50

We have additionally tested the entire pipeline, in which the outline generated by the first transformer becomes a part of the input to the second transformer. The values of the evaluation metrics are as follows:

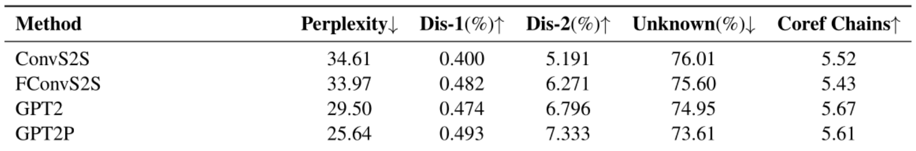
Perplexity	Distinct-1 (%)	Distinct-2 (%)	Unknown (%)	Coref Chains
826.628	0.308	0.202	50	0.51

These values are to be compared with the following values from the paper:



Method	Perplexity↓	Dis-1(%)↑	Dis-2(%)↑	Unknown(%)↓	Coref Chains↑
First stage					
keyword	74.46	0.964	7.132	/	/
abstract	35.53	0.776	10.060	/	/
Second stage					
story with keyword	17.82	0.461	6.188	74.26	5.67
story with abstract	10.65	0.512	7.358	74.54	5.81

These are the values of other models:



Method	Perplexity↓	Dis-1(%)↑	Dis-2(%)↑	Unknown(%)↓	Coref Chains↑
ConvS2S	34.61	0.400	5.191	76.01	5.52
FConvS2S	33.97	0.482	6.271	75.60	5.43
GPT2	29.50	0.474	6.796	74.95	5.67
GPT2P	25.64	0.493	7.333	73.61	5.61

The values that our model is getting are not nearly as good as what has been shown in the paper. The reasons for the same are as follows:

- **Training:** Due to computational restrictions, our models were only trained over 10% of the original dataset, for 10 epochs, while the model in the paper has been trained using the entire dataset.
- **Golden BERT:** The model in the paper uses an implementation of a discourse coherency element that uses a golden BERT standard to account for incoherence in the generated stories. While we have implemented the same, we have not trained it as much as we had hoped to, due to computational restrictions, leading to a poorer loss of the component, which contributes to a poorer loss of the second transformer.
- **Propagation:** The higher perplexity indicates subpar generated texts, and this, along with the discourse coherency element, which seems to have been predicting either of two markers, impacts the values of the other evaluation metrics.

- **Maximum Sentence Length:** We have taken the maximum sentence length as 500 characters, which is less than what was required in the paper, due to computational constraints.

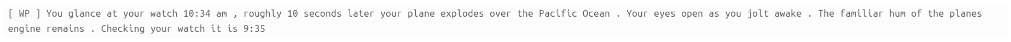
NOTE: We have taken readings for both the architecture from the hugging face gpt2 models and the transformers we made from scratch, but we noticed no significant difference in the values generated by the two.

From the outputs, we also notice that the outputs of the second transformer are not as coherent as those of the first transformer. This may be due to the fact that the error in the output from the first transformer propagates into the second transformer and reflects in its output.

Outputs

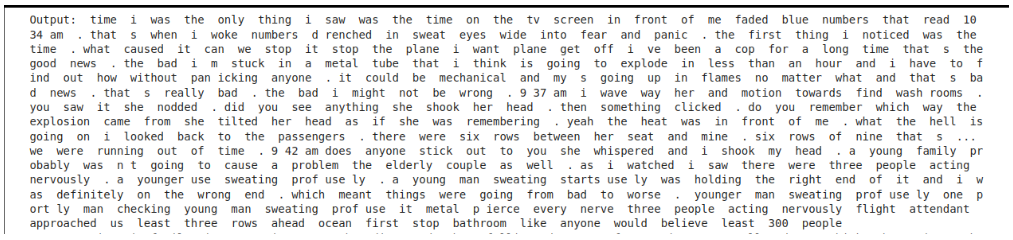
First transformer

Input Prompt



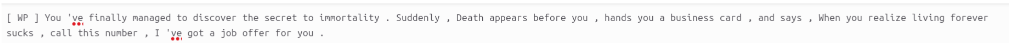
```
[ WP ] You glance at your watch 10:34 am , roughly 10 seconds later your plane explodes over the Pacific Ocean . Your eyes open as you jolt awake . The familiar hum of the planes
engine remains . Checking your watch it is 9:35
```

Generated Outline



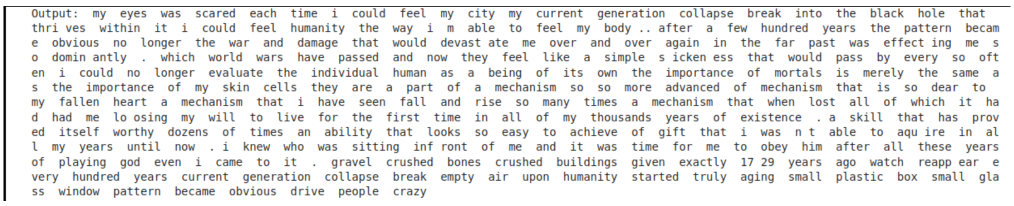
```
Output: time i was the only thing i saw was the time on the tv screen in front of me faded blue numbers that read 10
34 am . that s when i woke numbers d renched in sweat eyes wide into fear and panic . the first thing i noticed was the
time . what caused it can we stop it stop the plane i want plane get off i ve been a cop for a long time that s the
good news . the bad i m stuck in a metal tube that i think is going to explode in less than an hour and i have to f
ind out how without pan icking anyone . it could be mechanical and my s going up in flames no matter what and that s ba
d news . that s really bad . the bad i might not be wrong . 9 37 am i wave way her and motion towards find wash rooms .
you saw it she nodded . did you see anything she shook her head . then something clicked . do you remember which way the
explosion came from she tilted her head as if she was remembering . yeah the heat was in front of me . what the hell is
going on i looked back to the passengers . there were six rows between her seat and mine . six rows of nine that s ...
we were running out of time . 9 42 am does anyone stick out to you she whispered and i shook my head . a young family pr
obably was n t going to cause a problem the elderly couple as well . as i watched i saw there were three people acting
nervously . a younger use sweating prof use ly . a young man sweating starts use ly was holding the right end of it and i w
as definitely on the wrong end . which meant things were going from bad to worse . younger man sweating prof use ly one p
ort ly man checking young man sweating prof use it metal p ierce every nerve three people acting nervously flight attendant
approached us least three rows ahead ocean first stop bathroom like anyone would believe least 300 people
```

Input Prompt



```
[ WP ] You 've finally managed to discover the secret to immortality . Suddenly , Death appears before you , hands you a business card , and says , When you realize living forever
sucks , call this number , I 've got a job offer for you .
```

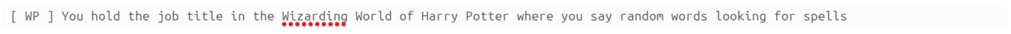
Generated Outline



```
Output: my eyes was scared each time i could feel my city my current generation collapse break into the black hole that
thri ves within it i could feel humanity the way i m able to feel my body .. after a few hundred years the pattern becam
e obvious no longer the war and damage that would devast ate me over and over again in the far past was effect ing me s
o domin antly . which world wars have passed and now they feel like a simple s icken ess that would pass by every so oft
en i could no longer evaluate the individual human as a being of its own the importance of mortals is merely the same a
s the importance of my skin cells they are a part of a mechanism so so more advanced of mechanism that is so dear to
my fallen heart a mechanism that i have seen fall and rise so many times a mechanism that when lost all of which it ha
d had me lo osing my will to live for the first time in all of my thousands years of existence . a skill that has prov
ed itself worthy dozens of times an ability that looks so easy to achieve of gift that i was n t able to aqu ire in al
l my years until now . i knew who was sitting inf ront of me and it was time for me to obey him after all these years
of playing god even i came to it . gravel crushed bones crushed buildings given exactly 17 29 years ago watch reapp ear e
very hundred years current generation collapse break empty air upon humanity started truly aging small plastic box small gla
ss window pattern became obvious drive people crazy
```

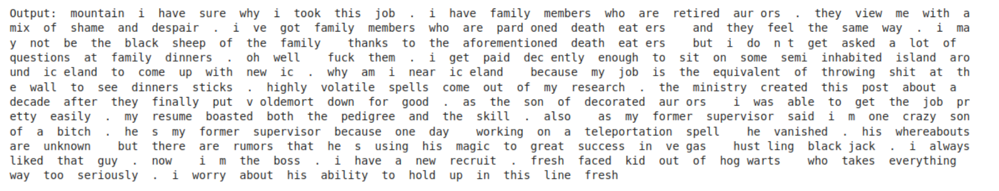
Second Transformer

Input Prompt



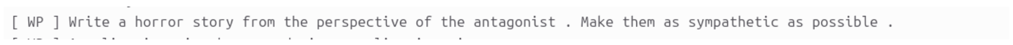
```
[ WP ] You hold the job title in the Wizarding World of Harry Potter where you say random words looking for spells
```

Generated Story



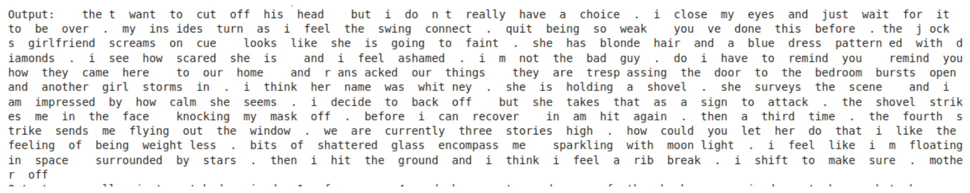
```
Output: mountain i have sure why i took this job . i have family members who are retired aur ors . they view me with a
mix of shame and despair . i ve got family members who are pard oned death eat ers and they feel the same way . i ma
y not be the black sheep of the family thanks to the aforementioned death eat ers but i do n t get asked a lot of
questions at family dinners . oh well fuck them . i get paid dec ently enough to sit on some semi inhabited island aro
und ic eland to come up with new ic . why am i near ic eland because my job is the equivalent of throwing shit at th
e wall to see dinners sticks . highly volatile spells come out of my research . the ministry created this post about a
decade after they finally put v oldemort down for good . as the son of decorated aur ors i was able to get the job pr
etty easily . my resume boasted both the pedigree and the skill . also as my former supervisor said i m one crazy son
of a bitch . he s my former supervisor because one day working on a teleportation spell he vanished . his whereabouts
are unknown but there are rumors that he s using his magic to great success in ve gas hust ling black jack . i always
liked that guy . now i m the boss . i have a new recruit . fresh faced kid out of hog warts who takes everything
way too seriously . i worry about his ability to hold up in this line fresh
```

Input Prompt



```
[ WP ] Write a horror story from the perspective of the antagonist . Make them as sympathetic as possible .
```

Generated Story



```
Output:   the t  want  to  cut  off  his  head   but  i  do  n t  really  have  a  choice  .  i  close  my  eyes  and  just  wait  for  it
to  be  over  .  my  ins ides  turn  as  i  feel  the  swing  connect  .  quit  being  so  weak   you  ve  done  this  before  . the  j ock
s  girlfriend  screams  on  cue   looks  like  she  is  going  to  faint  .  she  has  blonde  hair  and  a  blue  dress  pattern ed  with  d
iamonds  .  i  see  how  scared  she  is   and  i  feel  ashamed  .  i  m  not  the  bad  guy  .  do  i  have  to  remind  you   remind  you
how  they  came  here   to  our  home   and  r ans acked  our  things   they  are  tresp assing  the  door  to  the  bedroom  bursts  open
and  another  girl  storms  in  .  i  think  her  name  was  whit ney  .  she  is  holding  a  shovel  .  she  surveys  the  scene   and  i
am  impressed  by  how  calm  she  seems  .  i  decide  to  back  off   but  she  takes  that  as  a  sign  to  attack  .  the  shovel  strik
es  me  in  the  face   knocking  my  mask  off  .  before  i  can  recover   in  am  hit  again  .  then  a  third  time  .  the  fourth  s
trike  sends  me  flying  out  the  window  .  we  are  currently  three  stories  high  .  how  could  you  let  her  do  that  i  like  the
feeling  of  being  weight less  .  bits  of  shattered  glass  encompass  me   sparkling  with  moon light  .  i  feel  like  i  m  floating
in  space   surrounded  by  stars  .  then  i  hit  the  ground  and  i  think  i  feel  a  rib  break  .  i  shift  to  make  sure  .  mothe
r  off
```

Links to the Models

Models from Scratch

First Transformer: first_transformer.pt

Second Transformer: second_transformer.pt

Golden BERT: golden_bert.pt

Models using the GPT2 Config

First Transformer: first_transformer_gpt2.pt

First Transformer Small: first_transformer_small.pt

Golden BERT: golden_bert.pt

Second Transformer: second_transformer_gpt2.pth

Second Transformer Small: second_transformer_small.pth