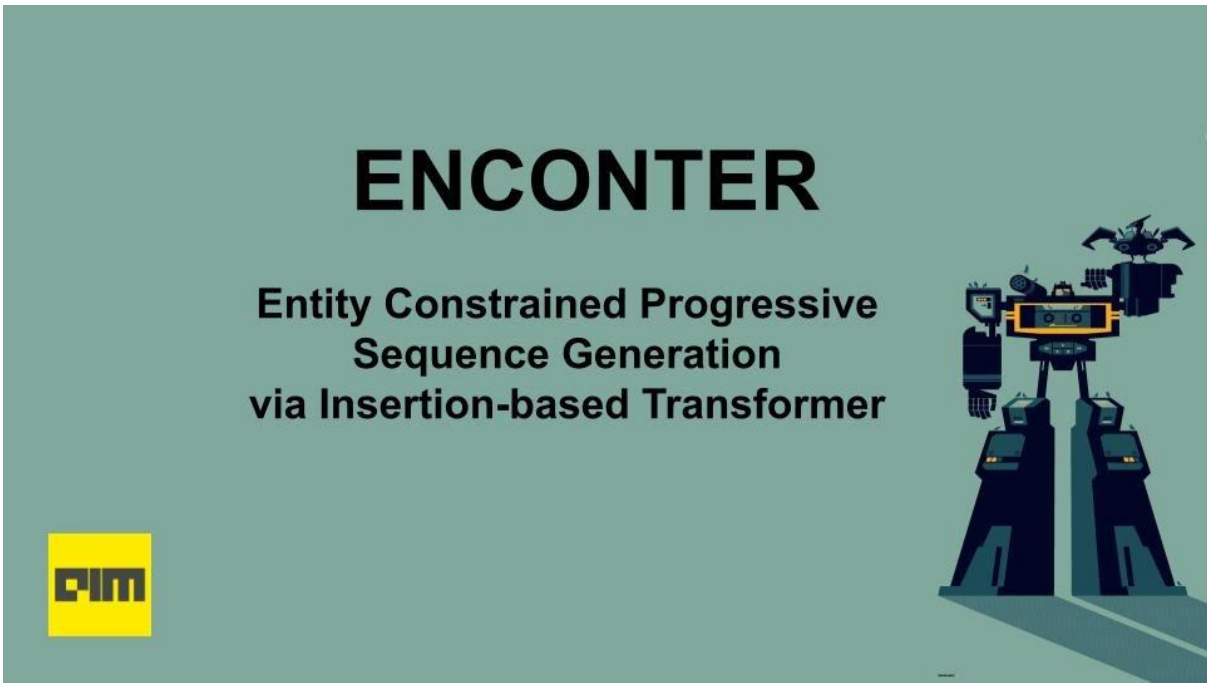
Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling


ENCONTER
Entity Constrained Progressive Sequence Generation via Insertion-based Transformer

Language modeling is one of the interesting machine learning problems. Some recent language modeling tasks that find remarkable improvement include neural machine translation, poem generation, text summarization and recipe generation. Despite continuous improvement, language modeling still lacks strong control over the text generation process. Text generation with control or governance via tokens as constraints is generally termed as the constrained text generation (CTG). Only countable research works have concentrated on improving the constrained text generation though it is so important in many real-world applications.

Constrained text generation is carried out with tokens or rules supplied as its constraints. A token may be an entity, a noun, a verb, a phrase or a sentence fragment. A rule may define the domain, length of the generated text, or quantification of the entries. Constraints are classified as soft-constraints and hard-constraints based on the way they are incorporated in a language model. Soft-constraints are a set of tokens or rules that can take any place in the generated text. Hard-constraints are a set of tokens or rules that must be positioned in the generated text as pre-defined. The soft-constrained models can incorporate the constraints as per their wish, but the hard-constrained models do not.

Autoregressive models generate text from left to right based on the previously generated tokens. Hence autoregressive models are unable to employ the hard-constraints. Therefore, by convention, soft-constrained models are autoregressive, whereas hard-constrained models are non-autoregressive. The recent state-of-the-art hard-constrained non-autoregressive text generation model, POINTER, uses an insertion transformer. This model generates text progressively using hard-constraints. During training, certain sparse tokens are masked, and the model is trained on the remaining tokens. After completion, the unmasked alternative tokens are masked, and the masked tokens are unmasked to continue the training process. However, this model fails to show good performance when using entities as hard-constraints.

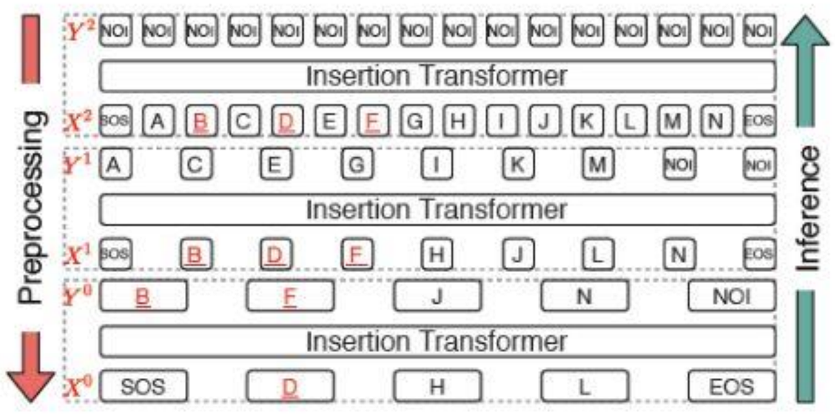
Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

Researchers from the Singapore Management University, Lee-Hsun Hsieh, Yang-Yin Lee and Ee-Peng Lim have introduced a successful model, the Entity Constrained Insertion Transformer, shortly known as the ENCONTER that addresses the problem of entity-constrained text generation. ENCONTER is able to generate a job description when entities such as candidate's desired skills and roles are provided. It can generate a recipe when entities such as ingredients and their quantities are provided. It is able to generate a poem when entities such as keywords are provided!

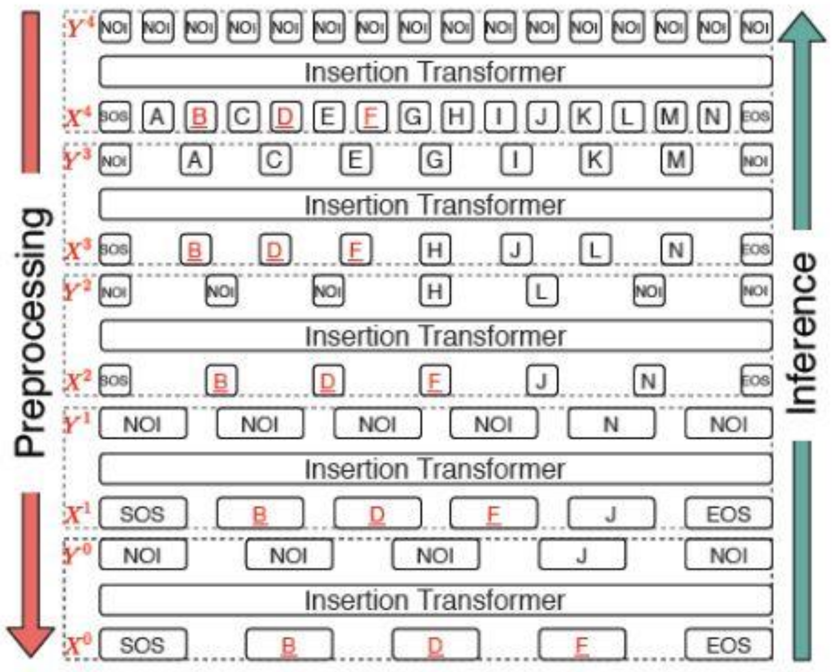
How does ENCONTER work?

The masking approach of the POINTER is developed further to force it to entertain entity constraints. This developed model is called POINTER-E. However, POINTER-E is severely suffering from cold start issues. Though it enables text generation with hard entity constraints, the initial tokens of the generated text are of poor quality and inappropriate to the context.

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling



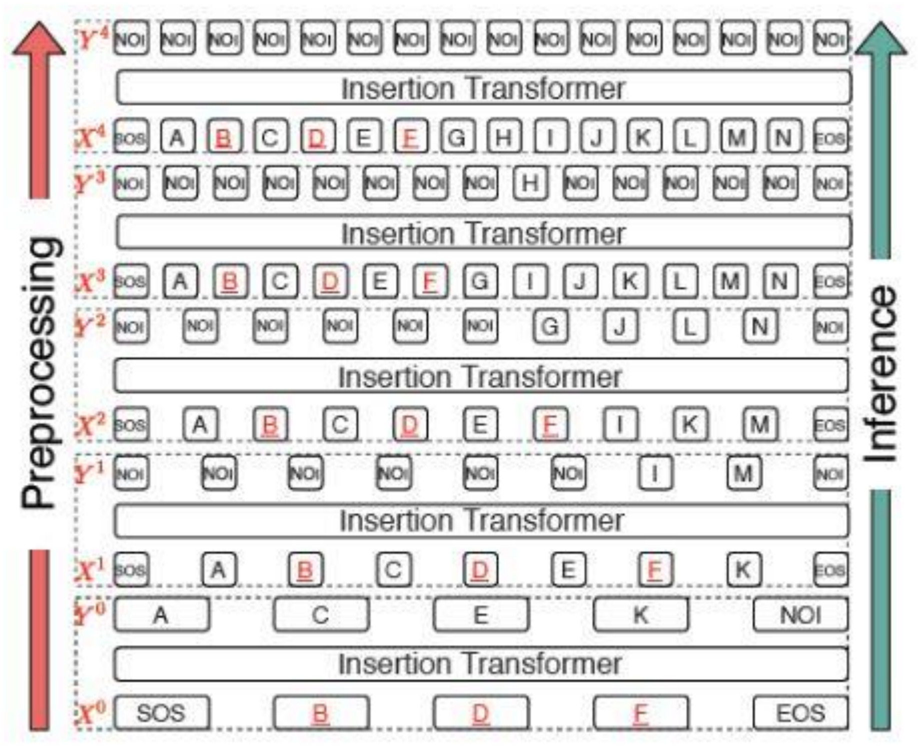
POINTER's masking strategy on generating a sequence of 'ABCDEFGHIJKLMN' with 'B', 'D' and 'F' as hard-constraints (Source)



POINTER-E's masking strategy on generating a sequence of 'ABCDEFGHIJKLMN' with 'B', 'D' and 'F' as hard-constraints (Source)

The major reason for the cold start of POINTER-E is the top-down preprocessing approach where the model is pre-trained to find out the masked entities, and the text generation is initiated with the hard entities. In ENCONTER, both the preprocessing and inference stages are performed in a bottom-up manner by providing the hard entities for initialization. ENCONTER demonstrates great success in avoiding the cold start by generating meaningful text throughout.

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling


Preprocessing
Inference
Insertion Transformer
Y^4 NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI NOI
X^4 SOS A B C D E F G H I J K L M N EOS
Y^3 NOI NOI NOI NOI NOI NOI NOI NOI H NOI NOI NOI NOI NOI NOI
X^3 SOS A B C D E F G I J K L M N EOS
Y^2 NOI NOI NOI NOI NOI NOI G J L N NOI
X^2 SOS A B C D E F I K M EOS
Y^1 NOI NOI NOI NOI NOI NOI I M NOI
X^1 SOS A B C D E F K EOS
Y^0 A C E K NOI
X^0 SOS B D F EOS

ENCONTER's masking strategy on generating a sequence of 'ABCDEFGHIJKLMN' with 'B', 'D' and 'F' as hard-constraints (Source)

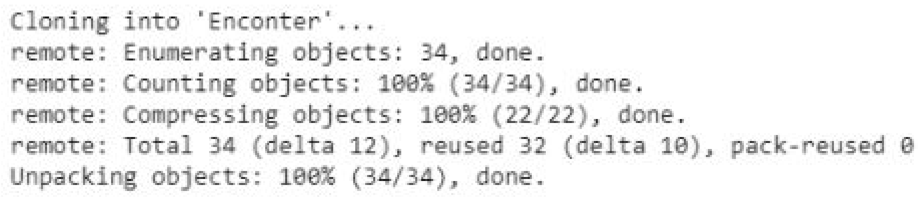
The text generation process is finished when the stage is full of [NOI] tokens, the no-insertion tokens. Further, a new version of the model called BBT ENCONTER with Balanced Binary Tree is introduced to save memory by minimizing the number of stages in a generation. The original version of the model is called Greedy ENCONTER.

Python Implementation
Step-1: Download source code
ENCONTER must be run in a python environment with a CUDA GPU runtime. The following command downloads the source codes to the local machine.

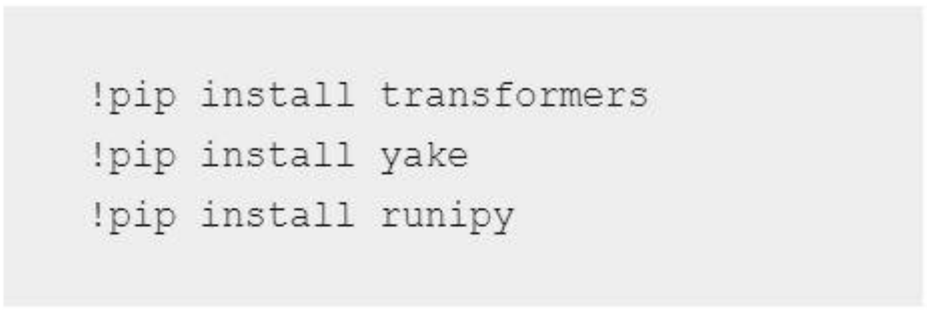
Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

```
!git clone https://github.com/LARC-CMU-SMU/Enconter.git
```

Output:


```
Cloning into 'Enconter'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 34 (delta 12), reused 32 (delta 10), pack-reused 0
Unpacking objects: 100% (34/34), done.
```

Step-2: Install dependencies
Install dependencies using the following commands. The preprocessing codes are provided in a notebook file. We need the runipy library to run the file at once with the bash command.


```
!pip install transformers
!pip install yake
!pip install runipy
```

Step-3: Preprocess dataset
Change the directory to refer the preprocessing notebook file.

```
%cd /content/Encounter/dataset
!ls
```

Output:

```
eng.testa  eng.testb  eng.train  preprocess_CoNLL.ipynb
```

Run the preprocessing notebook file using the following command.

```
!runipy preprocess_CoNLL.ipynb
```

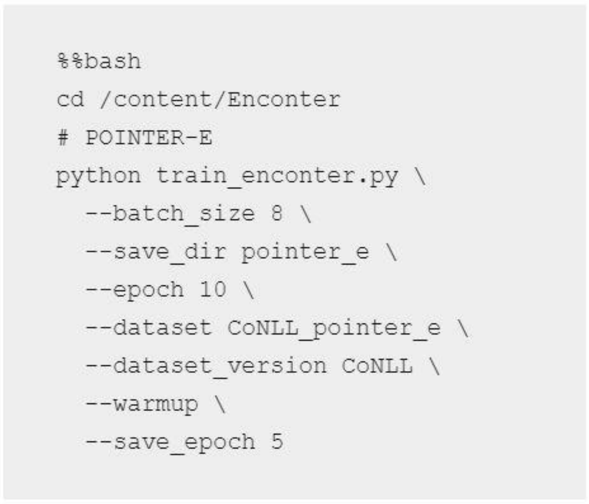
A portion of the output:

```
03/24/2021 12:31:48 PM INFO: Reading notebook preprocess_CoNLL.ipynb
03/24/2021 12:31:50 PM INFO: Running cell:
import spacy
from tqdm.notebook import tqdm
import pickle as pk
from collections import Counter
from nltk.corpus import stopwords
import re
from scipy.special import softmax
import numpy as np
import nltk

03/24/2021 12:31:51 PM INFO: Cell returned
03/24/2021 12:31:51 PM INFO: Running cell:
# Read CoNLL
```

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

Step-4: Train POINTER-E

Train the POINTER-E version for 10 epochs using the following command.



```
%%bash
cd /content/Enconter
# POINTER-E
python train_enconter.py \
  --batch_size 8 \
  --save_dir pointer_e \
  --epoch 10 \
  --dataset CoNLL_pointer_e \
  --dataset_version CoNLL \
  --warmup \
  --save_epoch 5
```

Step-5: Train Greedy ENCONTER

Train the Greedy ENCONTER version for 10 epochs using the following command.

```
%%bash

cd /content/Enconter

# Greedy Enconter

python train_enconter.py \

  --batch_size 8 \

  --save_dir greedy_enconter \

  --epoch 10 \

  --dataset CoNLL_greedy_enconter \

  --dataset_version CoNLL \

  --warmup \

  --save_epoch 5
```

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

Step-6: Train BBT ENCONTER

Train the BBT ENCONTER version for 10 epochs using the following command.

```
%%bash

cd /content/Enconter

# BBT Enconter

python train_enconter.py \

  --batch_size 8 \

  --save_dir bbt_enconter \

  --epoch 10 \

  --dataset CoNLL_bbt_enconter \

  --dataset_version CoNLL \

  --warmup \

  --save_epoch 5
```

Step-7: Infer with POINTER-E

Generate text with the POINTER-E version using the following commands.

```
%%bash

cd /content/Enconter

# POINTER-E

python predict_insertion_transformer_parallel.py \

  --save_dir "pointer_e" \

  --eval_dataset "./dataset/CoNLL_test" \

  --output_file "pointer_e"
```

```
python predict_insertion_transformer_parallel.py \

  --save_dir "pointer_e" \

  --eval_dataset "./dataset/CoNLL_test_esai" \

  --output_file "pointer_e_esai" \

  --inference_mode "esai"
```

Step-8: Infer with Greedy ENCONTER

Generate text with the Greedy ENCONTER version using the following commands.

```
%%bash

cd /content/Enconter

# Greedy Enconter

python predict_insertion_transformer_parallel.py \

  --save_dir "greedy_enconter" \

  --eval_dataset "./dataset/CoNLL_test" \

  --output_file "greedy_encontery"

python predict_insertion_transformer_parallel.py \

  --save_dir "greedy_enconter" \

  --eval_dataset "./dataset/CoNLL_test_esai" \

  --output_file "greedy_enconter_esai" \

  --inference_mode "esai"
```

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

Step-9: Infer with BBT ENCONTER

Generate text with the BBT ENCONTER version using the following commands.

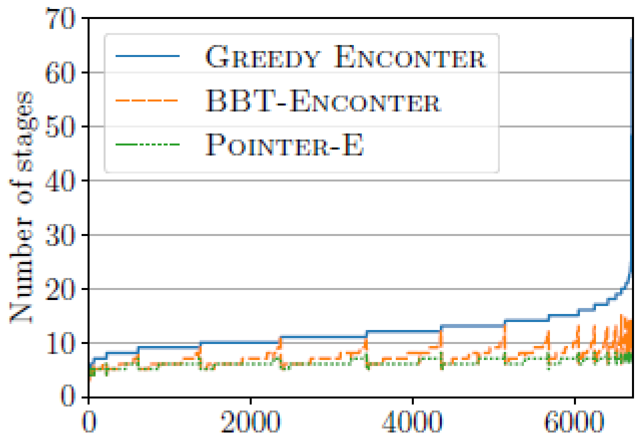
```
%%bash

cd /content/Enconter

# BBT Enconter

python predict_insertion_transformer_parallel.py \

  --save_dir "bbt_enconter" \

  --eval_dataset "./dataset/CoNLL_test" \

  --output_file "bbt_enconter"

python predict_insertion_transformer_parallel.py \

  --save_dir "bbt_enconter" \

  --eval_dataset "./dataset/CoNLL_test_esai" \

  --output_file "bbt_enconter_esai" \

  --inference_mode "esai"
```

Publication: Analytics India Magazine Online
Date: 28 March 2021
Headline: Complete Guide to ENCONTER: Entity Constrained Insertion Transformer for Language Modeling

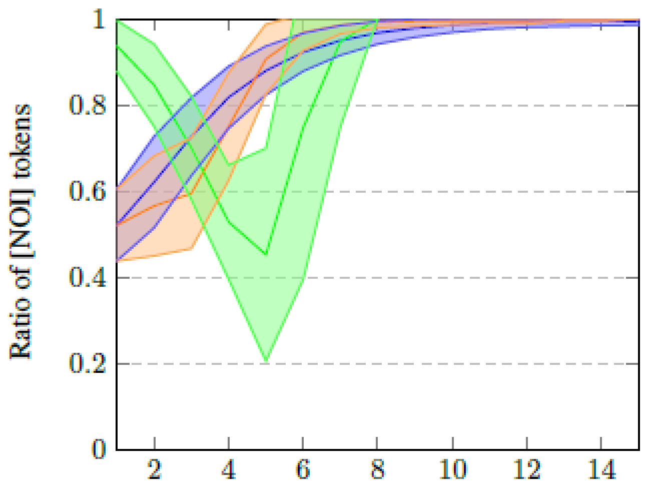
Performance of ENCONTER

ENCONTER is trained and evaluated on three public datasets:

1. CoNLL-2003
2. Singapore's Jobsbank – Software Developer (SD)
3. Singapore's Jobsbank – Sales and Marketing Manager (SM)



Number of stages vs documents on SD dataset (Source)



The ratio of [NOI] tokens vs number of stages on SD dataset (Source)

While comparing the performance of the BBT and the Greedy ENCONTER with other models, they greatly outperform recent state-of-the-arts on the BLEU, the MTR and the NIST metrics. Powerful models may be developed on top of the ENCONTER in the future to incorporate soft-constraints and rules too.