AUTOMATIC SCRIPT GENERATION USING DEEP LEARNING

Milestone-2 Presentation

Prof. Hans-Peter Bischof
Faculty Advisor
hpb@cs.rit.edu

Komal Atul Sorte
Student
ks7229@rit.edu

Prof. Matthew Fluet
Course Advisor
mtf@cs.rit.edu
Idea

Target
Build and enable an AI agent such that it learns a piece of script and automatically generates text based on appropriate context.

Further Accuracy Improvement
By considering,
- Contextual meaning of words
- Synonyms (e.g. check or validate)
- Word embeddings
Two approaches – word Level and character level

Data Pre-processing
1. Import the data and relevant libraries for data manipulation
2. Perform high level data inspection followed by detailed exploratory data analysis
3. Conduct data cleaning and treatment to get data in consumable format
4. Perform text analysis

Web Scraping
1. Search and identify apt data sources and elements
2. Understand how the data is organized and fetched on/from the source
3. Develop a crawler to extract data elements from source
4. Import data into the storage system in a form which could be further processed

Prediction
- Input Line
- Model
- Must predict and write a script
- Output Text/Dialogue

Building AI Agent
- Divide data into train, validation and test
- 60%
- 20%
- 20%

Evaluation
- Model is ready!

Milestone 2
Two approaches – word Level and character level
Scheduled Deliverables

WEEK 8
Final Language Model
- Encodings
- Train LSTM based word level language model
- Train LSTM based character level language model
- Fine tuning
- Compare results of word level approach vs character level approach
- Final model
Encodings
Mapper
- Represent every word to integer for input
- Represent every integer to word for after prediction
- Represent every character to integer for input
- Represent every integer to character for after prediction

Update batch files
- Encoding sequences of words into integers
- Encoding sequences of characters into integers

Reason
- Computers (NN) are good with numbers but not text.
- Embedding layer in LSTM needs input to be in integer format.
What is One-Hot Encoding?

representation of categorical variables as binary vectors.

Red = [1, 0, 0]
Blue = [0, 1, 0]
Green = [0, 0, 1]

Reason

For predicting the probability distribution for the next character/word.
Language Models & Fine-Tuning
Baseline Language Models

Word level language model - Underfits

- Embeddings
  - embedding dimensions: 60

- LSTM
  - units: 100

- Dense
  - activation: relu

- Dense
  - activation: softmax

Performs well on batch 0 but on batch 1:
- Overall training accuracy: 56.66%
- Overall training loss: 2+

Character level language model - Overfits

- LSTM
  - units: 256
  - layers: 1

- Dense
  - activation: softmax

- Dense
  - activation: softmax

- Dense
  - layers: 1

Overall training accuracy: 99.84%
Overall training loss: 0.014
LSTM based Word Level Language Model

**Batch 0: 1 episode**
- No. of words: 4384
- No. of unique words: 1016

**Batch 1: 1 season**
- No. of words: 82560
- No. of unique words: 6353

**Batch 2: 1-5 seasons**
- No. of words: 433334
- No. of unique words: 14798

**Observation**
Initially underfitting but performs comparatively better after fine-tuning. Validation accuracy never really took off. Needs more generalization and fine-tuning.

**Training Details**
- **Epochs:** 100
- **Batch size:** 120

**Training Results**

- **Batch 0**
  - Training accuracy: 99.81%
  - Validation accuracy: 2%
  - Training loss: 0.02
  - Validation loss: 19.71

- **Batch 1**
  - Training accuracy: 88.48%
  - Validation accuracy: 6%
  - Training loss: 0.36
  - Validation loss: 19.66

- **Batch 2**
  - Training accuracy: 44%
  - Validation accuracy: 10%
  - Training loss: 2.3
  - Validation loss: 11
**LSTM based Character Level Language Model**

Batch 0: 1 episode
- No. of characters: 22704
- No. of unique characters: 32

Training accuracy: 97%
Validation accuracy: 45%
Training loss: 0.07
Validation loss: 2

Batch 1: 1 seasons
- No. of characters: 431701
- No. of unique characters: 40

Training accuracy: 76%
Validation accuracy: 55%
Training loss: 0.6
Validation loss: 1.5

Batch 2: 1-5 seasons
- No. of characters: 2242157
- No. of unique characters: 41

Training accuracy: 67.5%
Validation accuracy: 65%
Training loss: 1
Validation loss: 1.2

**Observation**
Initially overfitting but performs better as we fine-tune and as data increases. Validation accuracy never really took off. Needs more generalization.

Epochs: 100 & Batch-size: 120
Challenges
Challenges

Environment

weasley.cs.rit.edu server

- Mismatch with the versions being used in the code vs the versions available on the machine.
- Docker container – but was discarded due to security reasons.
- Miniconda

Model

- Character level - Overfit
- Word level - Underfit
- Time consumption for running each batch
Next Steps
Milestones

- **Milestone 1**: Backend analytical data is created
- **Milestone 2**: Final model is ready
- **Milestone 3**: Final API is ready, and model is tested
- **Final**: Final Presentation & Poster
Milestone 3

- Testing model for prediction
- Conducting a human survey to evaluate model
- API to store all the outputs
References


Thank You