โครงการทดลองสร้างเครื่องผลิตกลอนสุภาพ ด้วย character-level LSTM language Model

จิตคพัฒน์ สวัสดิ์ผล

5940030622

บทนำ

ปัจจุบัน ในวงการ machine learning นั้น neural network เป็นโมเดลที่กำลังเป็นที่จับตามองและเป็นที่นิยม เพราะมีประสิทธิภาพดี ในทาง NLP การประยุกต์ใช้ neural network อย่างหนึ่งที่สำคัญคือการสร้าง language model ซึ่งเป็นการทำนายหน่วยทางภาษา (ตัวอักษร พยางค์ หรือคำ) ต่อไปจากบริบทที่เป็นหน่วยทางภาษาอื่น ๆ ในข้อมูล เมื่อฝึกสามารถนำไปใช้เสริมงาน NLP อื่น ๆ ไม่ว่าจะเป็น spell checker, machine translation, หรือ chatbot เพื่อให้ภาษา output นั้นเป็นธรรมชาติยิ่งขึ้น

ดังนั้น ผู้จัดทำโครงการจึงอยากทดสอบว่า language model ที่ใช้ neural network ในการฝึกนั้น เมื่อนำมาฝึกบนข้อมูลที่เป็นคำประพันธ์ ซึ่งมีรูปแบบภาษาที่แตกต่างไปจากบทความหรืองานเขียนทั่วไป จะสามารถเรียนรู้ได้มากน้อยเพียงใด เช่น สามารถเรียนรู้ฉันทลักษณ์ได้หรือไม่ สามารถเรียนรู้โครงสร้างของคำประพันธ์ได้หรือไม่ (จำนวนพยางค์แต่ละวรรค แต่ละบาท ขึ้นบรรทัดใหม่/วรรคใหม่ตรงไหน เป็นต้น) ซึ่งได้รับแรงบันดาลใจมาจาก *https://medium.com/@u41ppp/มารู้จักอักขราอาร์เอ็นเอ็น-โดยลองเล่นแต่งเป็นกลอนอักษรสาร-1264028567a5* ซึ่งได้ทำการทดลองคล้ายกันโดยฝึกบนเรื่องพระอภัยมณีของสุนทรภู่

จึงลองนำคำประพันธ์เรื่อง *ขุนช้างขุนแผน* ของสุนทรภู่ ซึ่งเป็นคำประพันธ์ประเภทกลอนสุภาพ (เพื่อไม่ให้ฉันทลักษณ์ยุ่งยากเกินไป) มาทดสอบสร้าง character-level language model และผลิต text จาก language model นั้น เพื่อดูประสิทธิภาพการเรียนรู้ของโมเดลแต่ละโมเดลว่าเป็นอย่างไร แตกต่างจากที่มีคนทำไว้อย่างไรบ้าง

วิธีการดำเนินงาน

โครงการนี้ใช้ข้อมูลเรื่อง*ขุนช้างขุนแผน* ซึ่งนำมาจากเว็บไซต์ของหอสมุดวชิรญาณ โดยใช้ไฟล์ที่อาจารย์ scrape ไว้แล้ว นำมาจัดรูปแบบให้เหลือแต่ตัวบทคำประพันธ์ บรรทัดละสองวรรค(หนึ่งบาท) แต่ละวรรคคั่นด้วย tab (\t) โดยข้อมูลทั้งหมดมี 3,109,782 ตัวอักษร แบ่งเป็น 2,165,343 ตัวอักษรสำหรับ training data, 636726 ตัวอักษรสำหรับ validation data และ 308,163 ตัวอักษรสำหรับ test data โดยไม่ shuffle

จากนั้นจึงนำข้อมูลมา preprocess โดยแบ่งข้อมูลทั้งหมดใน training data เป็น input-output sequence สำหรับฝึกโมเดล sequence ละ 64 ตัวอักษร (สันนิษฐานว่าเป็นปริมาณที่พอดี ไม่มากจนเปลืองหน่วยความจำ) โดยแต่ละ sequence คือการดึง 64 ตัวอักษรออกมาจาก text และขยับไปทีละ 1 ตำแหน่ง จากนั้นนำแต่ละตัวอักษรไปแปลงเป็น unique integer และแบ่งเป็น input-output (x,y) โดยใช้ output เป็นตัวอักษรแต่ละตัว และ input คือตัวอักษรก่อน output (maximum 64 ตัว) จากนั้นใช้ keras.utils.to_categorical เพื่อ one-hot encode แต่ละตัวอักษร เพื่อนำไปสร้าง array สำหรับฝึก บน Colab เพราะประสิทธิภาพ GPU และ RAM ดีกว่าคอมพิวเตอร์ที่มี

แต่เนื่องจากว่ามีปัญหาในการ preprocess data เพราะ RAM ของ google colab ไม่เพียงพอตอนที่ one-hot encode (และคอมพิวเตอร์ของตัวเองก็ไม่เพียงพอเช่นกัน) ดังนั้นจึงใช้ data แค่ 20,000 ตัวอักษรจาก training set เท่านั้น ทำให้มีความคาดหวังว่าผลการฝึกจะไม่ดีเท่าที่ควร และไม่สามารถหาเครื่องสำหรับฝึกที่ดีกว่านี้ได้ ด้วยเวลาที่จำกัด

สำหรับการฝึก จะฝึกโดยใช้โมเดล Sequential() ของ Keras และท้ายที่สุดการ generate text จะ input seed struing จากนั้นจะเลือกตัวอักษรที่มี probability สูงที่สุด เป็น output ซึ่ง output ก่อนหน้าจะนำไปต่อกับ input ถัดไปด้วย การ generate text นี้ไม่เขียนกฎเพิ่มเติม เพื่อให้เห็นประสิทธิภาพของโมเดลในการเรียนรู้โครงสร้างคำและฉันทลักษณ์ชัดที่สุด

Model 1: 1-layer LSTM stacked on a Dense layer

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 75)                44100
_________________________________________________________________
dense_5 (Dense)              (None, 71)                5396
=================================================================
Total params: 49,496
Trainable params: 49,496
Non-trainable params: 0
_________________________________________________________________
None
```

โมเดลนี้นำมาจากตัวอย่างใน https://machinelearningmastery.com/develop-character-based-neural-language-model-keras/

เหตุผลที่เลือกใช้โมเดลนี้ เพราะดูเป็นโมเดลที่ไม่ซับซ้อน (ไม่มี embedding layer, dropout) จึงฝึกได้ค่อนข้างเร็ว เพื่อนำไปทดลองใช้ผลิตข้อความในเบื้องต้น และยังเป็นพื้นฐานที่ใช้เทียบกับโมเดลอื่นที่ซับซ้อนขึ้นเพื่อดูประสิทธิภาพที่เพิ่มขึ้นจากการตั้ง parameter ต่าง ๆ รวมถึง layer ที่เพิ่มขึ้นได้

Model 2: Embedding Layer stacked on 3-Layer LSTMs with dropout

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_43 (Embedding)     (None, 64, 32)            2272
_________________________________________________________________
dropout_113 (Dropout)        (None, 64, 32)            0
_________________________________________________________________
lstm_91 (LSTM)               (None, 64, 512)           1116160
_________________________________________________________________
dropout_114 (Dropout)        (None, 64, 512)           0
_________________________________________________________________
lstm_92 (LSTM)               (None, 64, 512)           2099200
_________________________________________________________________
dropout_115 (Dropout)        (None, 64, 512)           0
_________________________________________________________________
lstm_93 (LSTM)               (None, 512)               2099200
_________________________________________________________________
dropout_116 (Dropout)        (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 71)                36423
=================================================================
Total params: 5,353,255
Trainable params: 5,353,255
Non-trainable params: 0
_________________________________________________________________
None
```

เนื่องจากผู้จัดทำไม่คุ้นเคยกับ neural network และการตั้งค่ามากนัก จึงทดลองใช้โมเดลโดยนำต้นแบบมาจากที่ใช้ในบทความ *https://medium.com/@u41ppp/มารู้จักอักขราอาร์เอ็นเอ็น-โดยลองเล่นแต่งเป็นกลอนอักษรสาร-1264028567a5* ซึ่งเป็น character-level language model ฝึกบนคำประพันธ์ไทยเหมือนกันเพื่อทดสอบประสิทธิภาพ ซึ่งคาดหวังไว้ว่าน่าจะดีกว่าโมเดลแรกเพราะมี embedding layer ทำให้สื่อความหมายได้ดีขึ้น และ dropout เพื่อป้องกัน overfitting

ผลการดำเนินงาน

Model 1 เมื่อนำโมเดลไป evaluate ด้วยคำสั่ง model.evaluate() บน validation set ได้ค่า loss, accuracy ดังนี้

```
[5.542001793510841, 0.16873996789727128]
```

ทดสอบกับ test set ได้ค่า loss, accuracy ดังนี้

```
[10.327602405609326, 0.020064205457463884]
```

และเมื่อทดลองสร้าง text โดยใช้ seed string เป็น '๏จะ' ทั้งหมด 500 ตัวอักษร ได้ผลดังนี้

```
๏จะคุค้ฟ้วกบุบหยายพันจะไป
ขุนไกรได้ฟังให้แม่เข้าไป       หัวนักนั้นแล้วไปมาถึงมา
ครั้นถึงลุกนั้นเป็นเห็นนาง      เป็นเล่นหมุมขึงจะคลอน
พลิ้งโร่งคร่อยว่องเมือนสงา     พระเดชรับหวานอุกใสรรรยา      พลังนั่นมีแล้วก็ไม่
ทิ่งตระบี่นนี่ควาดรรดคร        ดังไร
พระสวนั่นอกลามอุหมาบเพียงแต้          ปู่ย่าตายายสบายใจ     ข้าไกให้เกิดมาช้า      อักเฉิดเห็นเห็นผิดตระสายร่วงาย
แล้วก็กเที่นายส่านสองไม้มา
เลี้ยงมาก็เดือนสนร้องไป        ขุนช้างเอาเห็นเห็นผิดกลับถ้วนาน
นางแต่ลัวหน้ายวลยองคลอนค์ผิงดนกลายดแล้วนิน
เที่ยวถึงเย่นไม่
อยู่ในแล้วก็ได้
เที่ยงมาก็สมรับพลาน
```

สังเกตได้ว่ามีตัวอักษรที่ไม่เป็นคำ เช่น คุค้ฟ้วยุบหยาย ในบรรทัดแรก แต่ส่วนมากคำที่สร้างเป็นคำที่มีจริง อย่างไรก็ตาม การเว้นวรรคกับขึ้นบรรทัดใหม่ไม่ค่อยดี บางบรรทัดสั้น บางบรรทัดยาว มีเว้น tab เป็นวรรคที่สองเพียงไม่กี่ครั้ง และบางบรรทัดก็มีหลายวรรคเกินไป และแทบไม่พบการใช้คำสัมผัส ทั้งสัมผัสนอก-ใน โดยรวมอ่านไม่ค่อยรู้เรื่อง

ในขณะที่ Model 2 เมื่อนำโมเดลไป evaluate ด้วยคำสั่ง model.evaluate() บน validation set ได้ค่า loss, accuracy ดังนี้

```
[9.633312054660118, 0.017054574638844303]
```

ทดสอบกับ test set ได้ค่า loss, accuracy ดังนี้

```
[5.480580599311841, 0.17325441412520065]
```

และเมื่อทดลองสร้าง text โดยใช้ seed string เป็น '๏จะ' ทั้งหมด 500 ตัวอักษร ได้ผลดังนี้

```
๏จะกล่าวถึงนางศรีประจัน        เป็นเศรษฐีมีพันธุ์ด้วยกันมา
อยู่ท่าพี่เลี้ยงเมืองสุพรรณ     ให้ห้าวันตามที่มีหมายมาฯ
๏ครานั้นพระองค์ผู้ทรงเดช       ทุกประเทศฦาลบสยบสยอน
ครั้นจวนแจ้งแสงศรีรวีวร        พระภูธรเสด็จสู่ที่สรงชล
ไขสุหร่ายหยัดย้อยเป็นสองสายบ่าน          เอาเบี้ยบนลนลานเหน็บฝาเกลื่อน
บ้างเร่งหมอตำแยอย่าแชเชือน    ข่มท้องร้องเตือนลูกขวางตัว
บ้างก็เข้าหนุนหลังนั่งเคียงข้าง  กลายแก้วโดดแหวกเข้าแทรกกลาง          ชกหัวขุนช้างที่กลางเกลี้ยง
ขุนช้างทำหลับอยู่กับเตียง      ฝ่ายนางพิมนอนเคียงค่อยเมียงมอง
ขุนช้างวางร้องก้องกู่โวย       ขโมยลักเมียกูจู่จาก
```

จะเห็นได้ว่าแทบทุกคำอ่านได้ และโครงสร้างดีขึ้นมาก แทบทั้งหมดเป็นโครงสร้าง 1 บาท 2 วรรค มีการใช้สัมผัสนอกที่ค่อนข้างดี แม้คำที่สัมผัสกันพยางค์อาจจะคลาดเคลื่อนเล็กน้อย แต่โดยรวมอ่านแล้วยังพบคำสัมผัสตามแบบที่กลอนสุภาพควรจะ

เป็น ส่วนสัมผัสในนั้นมีบ้างไม่ทุกวรรค ด้านเนื้อหาอ่านรู้เรื่องมากกว่า model1 มาก พอมีโครงสร้างประโยคชัดเจนว่าใครทำอะไรอย่างไร แต่หากเอาแต่ละท่อนมาต่อกันก็ยังคงไม่สามารถสื่อเรื่องราวได้ดีใกล้เคียงกับคนเขียน

อภิปรายผล

ในส่วนการวัดผลเชิงปริมาณ สังเกตได้ว่าได้ค่า loss ที่สูงมาก และ accuracy ต่ำมาก และตัวเลขของทั้งสองโมเดลแทบไม่ต่างกัน ทั้งที่ model 2 หากดูด้วยตาเปล่าจะรู้สึกว่าเขียนได้ดีขึ้นมากในแง่โครงสร้างคำประพันธ์ โครงสร้างประโยค และโครงสร้างคำที่มีอยู่จริง อาจเป็นเพราะการวัดผลนั้นทำโดย predict output จาก input ที่เป็นตัวอักษร ซึ่งในแต่ละส่วนของ text จะมีการกระจายตัวและความเกิดขึ้นบ่อยของตัวอักษรแต่ละตัวไม่เท่ากัน จึงเกิด overfitting (ซึ่ง dropout ในโมเดลที่สองแทบไม่ส่งผลแตกต่าง) รวมถึงธรรมชาติของงานประเภทร้อยกรองที่ยืดหยุ่นกว่างานเขียนที่เป็นร้อยแก้วทั้งในด้านการใช้คำและโครงสร้างประโยค ทำให้อาจจะประเมินคุณค่ายากในแง่ความถูกต้องของภาษา ด้วยเวลาที่จำกัด จึงไม่สามารถค้นคว้าหามาตรวัดหรือวิธีประเมินผลที่มีประสิทธิภาพกว่านี้ได้ ส่วนด้านคุณภาพพบว่าข้อความที่ผลิตออกมาดีขึ้นมากใน model 2 ดังที่กล่าวไปแล้ว อาจเป็นเพราะการมี embedding layer ที่ทำให้เก็บความหมายและบริบทของแต่ละตัวอักษรที่อยู่รวมกันได้ดีขึ้น รวมถึง lstm ที่จำนวนชั้นมากขึ้น แต่ก็แลกมาด้วยเวลาฝึกโมเดลที่นานขึ้นมากเช่นกัน

ในเบื้องต้นตั้งใจจะทำ word-level แล้วเปรียบเทียบประสิทธิภาพ แต่เนื่องด้วยเวลา และทรัพยากร(vocab size ที่จะใหญ่มาก ทำให้เวลา encode ใช้พื้นที่เยอะ) ที่จำกัด (รวมถึงข้อสันนิษฐานว่าอาจทำไม่ได้ดีการผลิตคำประพันธ์ เพราะกลอนสุภาพโครงสร้างหลักขึ้นอยู่กับพยางค์ไม่ใช่จำนวนคำ เวลาผลิตข้อความน่าจะดูไม่ค่อยเหมือนบทกลอนที่ควรจะเป็น ซึ่งอันที่จริง character-level ก็เกิดปัญหานี้เช่นกันเพราะไม่ได้ยึดพยางค์ แต่น่าจะทำได้ดีกว่า เพราะความใกล้เคียงระหว่างจำนวนตัวอักษรต่อจำนวนพยางค์น่าจะมากกว่าจำนวนคำต่อจำนวนพยางค์ เพราะเวลาตัดคำบางครั้งจะมีคำประสมที่ยาวมาด้วย) จึงไม่ได้ทำและทำเปรียบเทียบ character-level สองโมเดลแทน

อุปสรรคหลักคือไม่สามารถใช้ข้อมูลทั้งหมดเพื่อฝึกบน colab หรือคอมพิวเตอร์ของตนเองได้ เพราะ memory จะเต็มตอนที่ one-hot encode แต่ละ sequence ในครั้งถัดไปควรเช่าคอมพิวเตอร์ที่ประสิทธิภาพดีขึ้นมาใช้ฝึก (อาจเป็น google cloud computing ที่ครั้งนี้ไม่มีเวลาเตรียมการ)

นอกจากนั้นรู้สึกว่า keras ใช้ค่อนข้างยากในด้านการเตรียมข้อมูลเพื่อ train แต่ก็ปรับแต่ง parameter ได้มาก ควรศึกษาเพราะเป็น front end หนึ่งที่ดูจะนิยมใช้ในการฝึกโมเดล neural network

สรุปผล

ผลที่ได้รับจากการทำโครงการนี้คือ ได้ทราบว่า RNN สามารถใช้ฝึก language model สำหรับผลิตข้อความที่เป็นคำประพันธ์ได้ค่อนข้างดี หากทดลองและปรับแต่ง parameter ต่าง ๆ ให้เหมาะสม สามารถเรียนรู้ฉันทลักษณ์ โครงสร้างคำ โครงสร้างประโยคได้ค่อนข้างดีมากโดยแทบไม่ต้องเขียนกฎทางภาษาศาสตร์เพิ่มเติม อย่างไรก็ตามในองค์รวมยังไม่สามารถสื่อความหมายออกมาเป็นเรื่องได้ จะมีแค่ส่วน ๆ ที่แต่ละส่วนสื่อความหมายโดยไม่ขึ้นต่อกันเท่านั้น ดังนั้นแม้ว่าจะเห็นได้ว่า RNN จะมีประสิทธิภาพเพียงใด ความรู้ทางภาษาศาสตร์ยังคงมีประโยชน์ในการกำกับให้โมเดลทางภาษาทำงานได้ดีขึ้น และควรใช้ควบคู่กันไปกับ machine learning เพื่อประสิทธิภาพสูงสุด เป็นหลักฐานยืนยันได้อย่างดีว่า แม้ A.I. กำลังจะกลายเป็นกระแสหลักในการทำงานต่อในอนาคต แต่ความรู้ทางมนุษย์ศาสตร์ โดยเฉพาะภาษาศาสตร์ยังคงสำคัญและใช้คู่กับคอมพิวเตอร์ได้