ปวีณ มานะกุล 6140122222

ภวิศภาคย์ อัครจรัสวงศ์ 6140177822

รายงาน Contest 2

# Data

เราได้ทดลองใช้ข้อมูลจากหลาย ๆ แหล่ง โดยเพิ่มขนาดของข้อมูลให้มากขึ้น เรียงจาก

* nyt\_eng\_2000 (195MB)
* nyt\_eng (1GB) (105724492 tokens)
* nyt\_eng (2GB) (502821258 tokens)
* One billion word benchmark dataset (4GB) (768648884 tokens)

# ตารางเทียบค่า

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Score/ Model | nyt\_eng\_2000 + pyspellchecker  (dictionary สร้างเอง) | nyt\_eng\_2000 + pyspellchecker  (AGID) | nyt\_eng\_2000 | nyt\_eng 1gb | nyt\_eng 2gb | One billion word benchmark dataset |
| Precision | 0.0983 | 0.1123 | 0.1224 | 0.1319 | 0.1411 | 0.1598 |
| Recall | 0.0305 | 0.0367 | 0.0357 | 0.0338 | 0.0417 | 0.0532 |
| F0.5 | 0.0681 | 0.0795 | 0.0824 | 0.0892 | 0.0955 | 0.1141 |

ซึ่งจากการทดลองพบว่า Data ที่มากขึ้น จะทำให้ผลลัพท์ดีขึ้นตามลำดับ และจากโมเดล จะเห็นว่าค่า recall น้อยมากเมื่อเทียบกับค่าอื่น จึงคิดว่าโมเดลนี้ มีการเดาคำตอบน้อย แต่มีความแม่นยำสูง

# Language model

เนื่องจากเวลาไม่พอ และโมเดลใช้เวลา run นาน เราจึงเลือกใช้ 5-gram ในการสร้างโมเดลนี้ และใช้ KenLM ในการ train

# Algorithm

เราได้สร้าง def หลาย ๆ อันในโมเดลนี้ เพื่อความสะดวกและไม่ให้โมเดลใช้เวลา run นานเกินไป โดยแบ่งdef หลัก คร่าว ๆ ได้ดังนี้

1. def ที่รับ input และสร้าง output

เพื่อความสะดวก เราจึงเขียนคำสั่งที่รับ string เป็นประโยคแล้วนำไปหา candidate และคำนวน score ใน def อื่น แล้วนำผลลัพท์มาสร้างไฟล์ใหม่เป็น output

1. def สร้าง dictionary

เราได้ใช้ตัวอย่างคำศัพท์โดยได้มีการอ้างอิงจาก[เปเปอร์นี้](https://www.aclweb.org/anthology/W18-0529.pdf) ที่ได้ใช้ Automatically Generated Inflection Database ([AGID](http://wordlist.aspell.net/other/)) ในการหา candidate ของคำต่าง ๆ ใน text และยังสร้าง dictionary ของ articles ,preposition, pronoun และ modal verb เอง พบว่า AGID ให้ผลลัพท์ที่ดีกว่าการสร้าง dictionary ของ verb และ noun เอง จึงคาดว่า AGID มีคลังคำศัพท์ที่สมบูรณ์กว่า

1. def สร้าง candidate

เริ่มหา candidate ด้วยการนำ text มา tokenize ด้วย spacy แล้วนำแต่ละคำไปวนหาว่า อยู่ใน dictionary อะไร เสร็จแล้วจึงนำคำอื่น ๆ ที่สามารถเป็น candidate ได้ มา generate เป็นประโยคใหม่ แล้วนำไป คำนวน score เพื่อหา best candidate แล้วส่งกลับไปยัง def ที่หนึ่ง เพื่อสร้าง output text

1. def คำนวน score

รับ candidate ทั้งหมดมาคำนวณหาประโยคที่ดีที่สุดโดยใช้KenLm

หมายเหตุ : เราได้เจอ [github](https://github.com/chrisjbryant/lmgec-lite) อันหนึ่งที่ทดลองทำโมเดลโดยอ้างอิงเปเปอร์ จึงได้นำมาเป็น guideline ในการทดลองสร้างโมเดล

หมายเหตุ 2 : เราได้ทดลองใส่ pyspellchecker ในโมเดลแล้ว พบว่า pyspellchecker ทำให้โมเดลมีประสิทธิภาพลดลง และใช้เวลา run นานขึ้นมาก จึงไม่นำมาใส่ในโมเดล

# ข้อสรุป

จากการทดลอง เราได้ข้อสรุปว่า คุณภาพของโมเดลจะเพิ่มขึ้นตาม แหล่งข้อมูลที่นำมา train ว่าใหญ่แค่ไหน และความสมบูรณ์ของ dictionary ที่สร้างว่ามีคำครอบคลุมมากแค่ไหน เพราะจากไฟล์ทดลองที่มีมากกว่า 50000 ประโยค ทำให้คาดว่า แหล่งข้อมูลที่น้อยเกินไป จะส่งผลให้โมเดลไม่มีแหล่งอ้างอิงถึงบางประโยค ทำให้ประโยคเหล่านั้นไม่ได้รับการแก้ไข

เราจึงเลือกใช้โมเดลที่ train ด้วยข้อมูล One billion word benchmark dataset (4GB) (768648884 tokens) และ dictionary จาก AGID ในการสร้างโมเดลนี้

แต่ก็มีข้อเสียคือ ข้อจำกัดของ colab ที่มี disk เพียง 100gb และ RAM 25gb เราคิดว่าโมเดลนี้สามารถพัฒนาได้อีกตามขนาดข้อมูลที่นำไป train กับโมเดล 