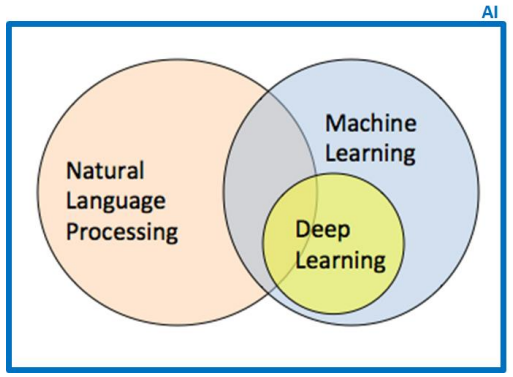
Advanced topics on Deep Learning

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

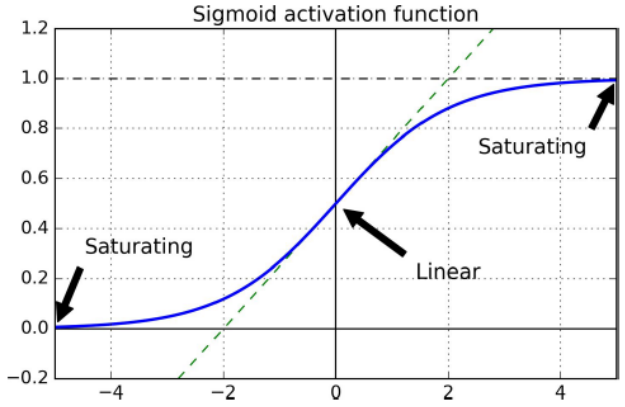
Today

- Advanced topics on Deep Learning
  - Vanishing Gradient Problem
  - Weight Initialization
  - Optimization Schemes
  - Dealing with overfitting: Dropout, Batch Norm
  - Transfer Learning
  - Data Augmentation
- NLP
  - Goal and history
  - Rule-based vs statistical approach
  - NLP tasks and pipeline
  - N-gram, TF-IDF
  - Word embeddings
  - RNN, LSTM



Vanishing Gradient Problem

- ทบทวน: ใน **Backpropagation** ค่า **gradient** ของ **loss** จะถูกส่งย้อนจาก **layer** ปลายกลับมายัง **layer** ต้น เพื่อ **update** ค่า **weight** ทั้งหมด
- ปัญหา:
  - ในขณะที่ส่งย้อน ค่า **gradient** ลดลงอย่างมาก เข้าใกล้ 0 *
  - ถ้า **gradient** เป็น 0 จะทำให้ **weight** หยุดการ **update**
  - ทำให้การเรียนรู้หยุดชะงัก ☹
- Solution
  - ใช้ **activation function** ที่ไม่ saturate (ใช้ ReLU, Leaky ReLU, ELU แทน Sigmoid)
  - เปลี่ยนวิธี **weight initialization** (ใช้วิธี Xavier หรือ He แทน random, หรือใช้ pre-trained network)
  - ใส่ **carry track** หรือ **skip connection** ในโมเดล (LSTM, ResNet)



* อาจเกิดปัญหาตรงข้าม (Exploding Gradient) ได้เช่นกัน หาก **gradient** โตขึ้นอย่างรวดเร็ว ทำให้การ **optimize** ไม่ converge

Better Optimization Schemes

- ปัญหา: SGD optimizer ใช้เวลา train นานและอาจได้คำตอบที่เป็น local minima
- Solution: เปลี่ยนไปใช้ optimizer ที่ดีกว่า เช่น
  - Momentum ใช้ momentum ทำให้หลุดจากหล่ม local minima
  - AdaGrad ค่อยๆ ลด (decay) learning rate ในทิศที่ชันที่สุด
  - RMSProp ปรับ AdaGrad ให้ไม่ decay แรงเกินไป
  - ADAM รวมข้อดีของ RMSProp กับ Momentum
- ในทางปฏิบัติให้ลองใช้ ADAM ไปเลย ไม่ก็ RMSProp

Regularization Techniques

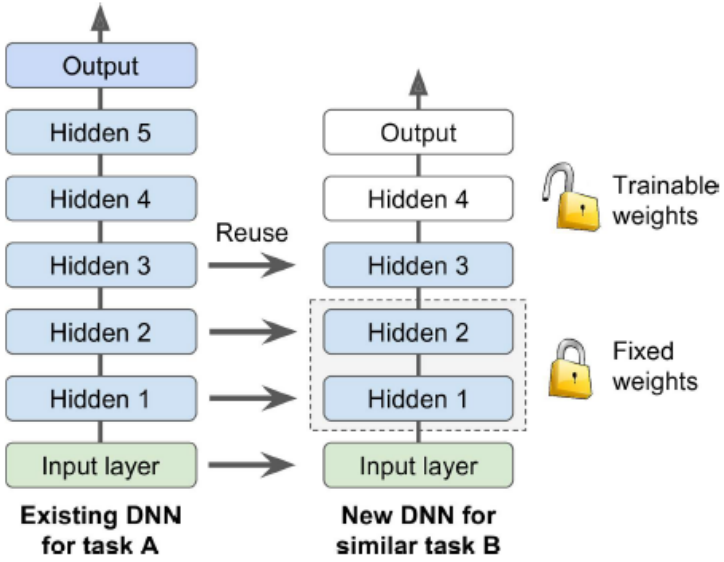
- ปัญหา: โมเดล **deep learning** เป็นฟังก์ชันที่มีความยืดหยุ่นสูง มักที่จะ **overfit** ข้อมูลได้ง่าย
  - ทบทวน: โมเดลที่ **overfit** จะทำนายข้อมูลชุด **train** ได้แม่นยำ แต่กลับทำนายข้อมูลชุด **test** ได้แย่ ☹
- วิธีลด **overfitting** ที่นิยมใช้กับ **DL models** มีดังนี้
  - เพิ่มจำนวน **training set**
  - **Batch Normalization**
    - แทรก **BN layer** เข้าไปก่อนทุกๆ **activation layer**
    - ทำการ **zero-center** ค่า **output** โดยใช้ **mean, variance** ของค่าใน **batch** นั้น
    - ช่วยทำให้ **train** เร็วและเสถียรขึ้นด้วย
  - **Early stopping:** หยุด **train** ถ้า **validation loss** เริ่มสูงขึ้นเกิน **threshold**
  - **Dropout:** สุ่ม **ignore** บาง **neuron** ในขณะ **train**
    - ทำง่าย, ช่วยให้ **train** เร็ว, ผลทำนายดีขึ้น

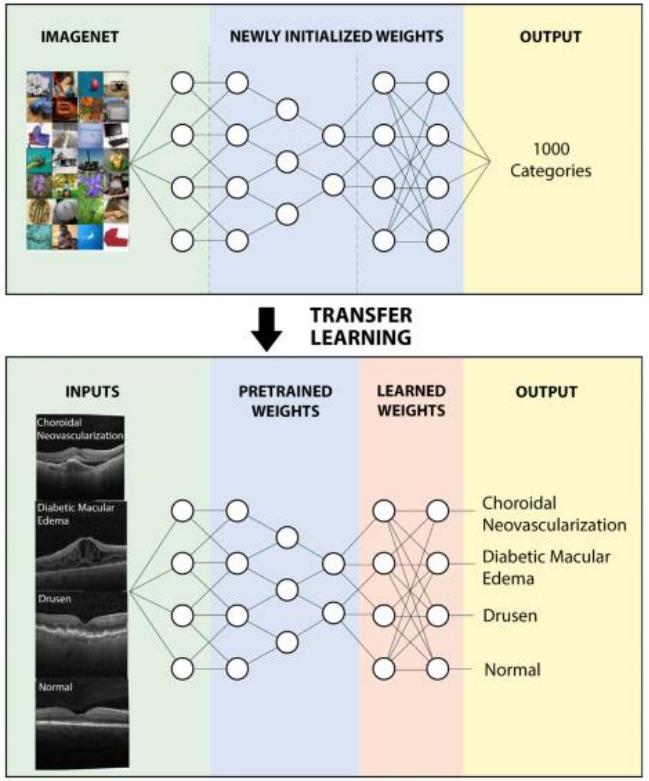
Equation 11-3. Batch Normalization algorithm

1. $\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} \mathbf{x}^{(i)}$
2. $\sigma_B^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(\mathbf{x}^{(i)} - \mu_B\right)^2$
3. $\widehat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
4. $\mathbf{z}^{(i)} = \gamma \widehat{\mathbf{x}}^{(i)} + \beta$

Transfer Learning

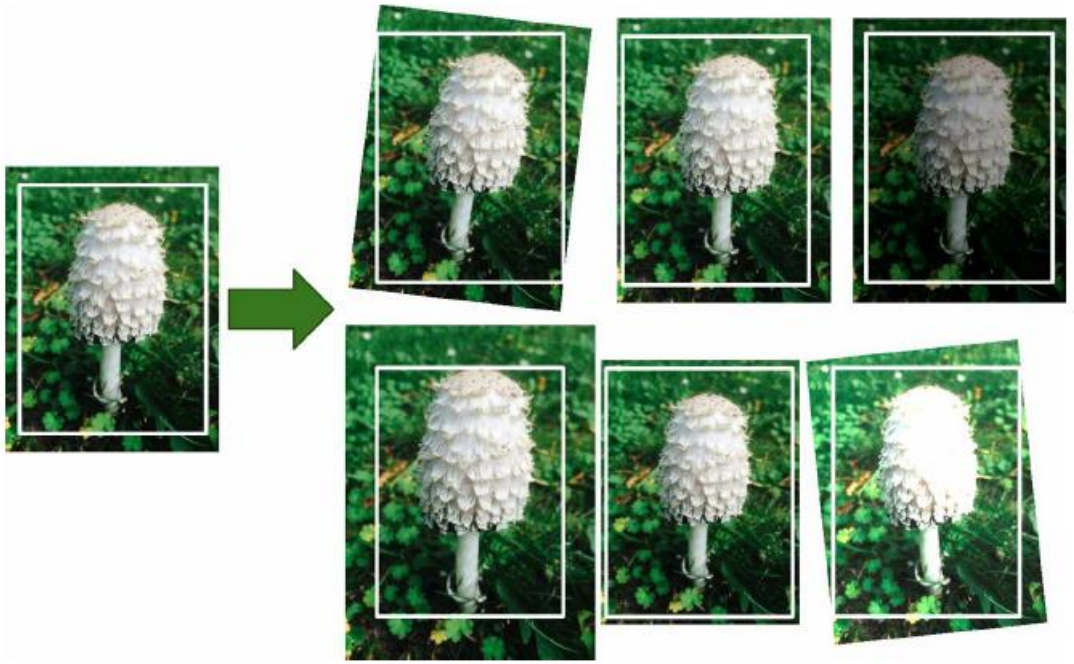
- ปัญหา: ต้องการนำโมเดลที่ **train** กับข้อมูลทั่วไป ไปใช้กับงานที่เฉพาะเจาะจง
- Solution: Transfer Learning
  - นำ **pre-trained weights** ที่ **train** กับข้อมูลทั่วไปมาใช้เป็นจุดเริ่มต้น
  - ทำการ **train** กับข้อมูลชุดใหม่ที่เป็นงานเฉพาะ
  - ระหว่าง **train** ทำการ **freeze layer** แรกๆ แล้ว **train** เฉพาะ **layer** ท้ายๆ
  - ช่วยให้ **train** เร็วขึ้นมาก และทำนายได้แม่นยำสูงขึ้นมาก





Data Augmentation

- ปัญหา: ข้อมูล **training set** ที่เป็นรูปภาพ **(CNN)** ไม่มากพอ
- **Solution:** สังเคราะห์รูปภาพใหม่ๆ ขึ้นมาจากภาพที่มีอยู่ โดยการ **transform** ภาพ
  - Shift
  - Rotate
  - Scale
  - ปรับสภาพแสง
    - Brightness
    - Contrast
    - Saturation



Intro to NLP

Natural Language Processing

- goal: understanding human language
- history
  - ก่อนปี 1980s ระบบ NLP ส่วนมากเป็นแบบ **rule-based** คือพัฒนาโดยการเขียนกฎเกณฑ์ต่าง ๆ (เกี่ยวกับความหมาย, ไวยากรณ์) ด้วยมือ
    - แตกปัญหาการเข้าใจภาษาแบบ high-level ออกเป็น low-level task ย่อยๆ
    - เช่น ระบบสนทนา chatbot ต้องประกอบด้วย โปรแกรมตัดคำ โปรแกรมแปลความหมายคำ โปรแกรมถอดโครงสร้างไวยากรณ์
  - ปลาย 1980s เริ่มหันมาใช้วิธีการทางสถิติ (**statistical approach**) มากขึ้น
    - ML algorithms (e.g. decision trees)
    - part-of-speech tagging โดยใช้ Hidden Markov Model (HMM)
  - ช่วงปี 2010s พบว่าสามารถใช้ Deep Neural Network ทำงาน NLP ต่าง ๆ ได้ผลดีเยี่ยมเป็น state-of-the-art
    - นิยมใช้เทคนิค word embeddings เพื่อ capture ความหมายของคำในภาษา
    - เริ่มสามารถแก้ปัญหา high-level ได้เลย โดยไม่ต้องแตกเป็น task ย่อย

Natural Language Processing

- เปรียบเทียบ rule-based vs statistical NLP
  - statistical NLP ทนต่อความหลากหลาย (variation) ของภาษาได้ดีกว่าแบบ rule-based
    - คำที่ไม่เคยพบมาก่อน
    - คำที่สะกดผิด
    - คำที่หายไป
    - ความถี่ในการใช้คำ
  - statistical NLP สามารถวิวัฒนาการไปตามข้อมูลที่มี ยิ่งป้อนข้อมูลให้มากเท่าใด ยิ่งเก่งขึ้นเท่านั้น ในขณะที่แบบ rule-based ต้องสร้างกฎให้ซับซ้อนขึ้นเรื่อย ๆ ซึ่งยาก
  - statistical NLP ต้องพึ่งแรงงานคนในการ label ข้อมูลสำหรับการ train

NLP tasks

- Syntax
  - grammar induction
  - stemming/lemmatization
  - part-of-speech tagging
  - parsing
  - word segmentation
- Semantics
  - lexicon/semantic networks
  - machine translation
  - sentiment analysis
  - topic segmentation
  - question answering

เกี่ยวกับ**ไวยากรณ์**

เรียนรู้กฎไวยากรณ์

ลดรูปคำ เช่น studied -> study

แท็กประเภทคำ เช่น noun, verb, adj

แปลงประโยคเป็นโครงสร้างแกรมมาแบบ tree

ตัดคำ (เฉพาะบางภาษา)

เกี่ยวกับ**ความหมายของคำ**

หาความสัมพันธ์ระหว่างคำ

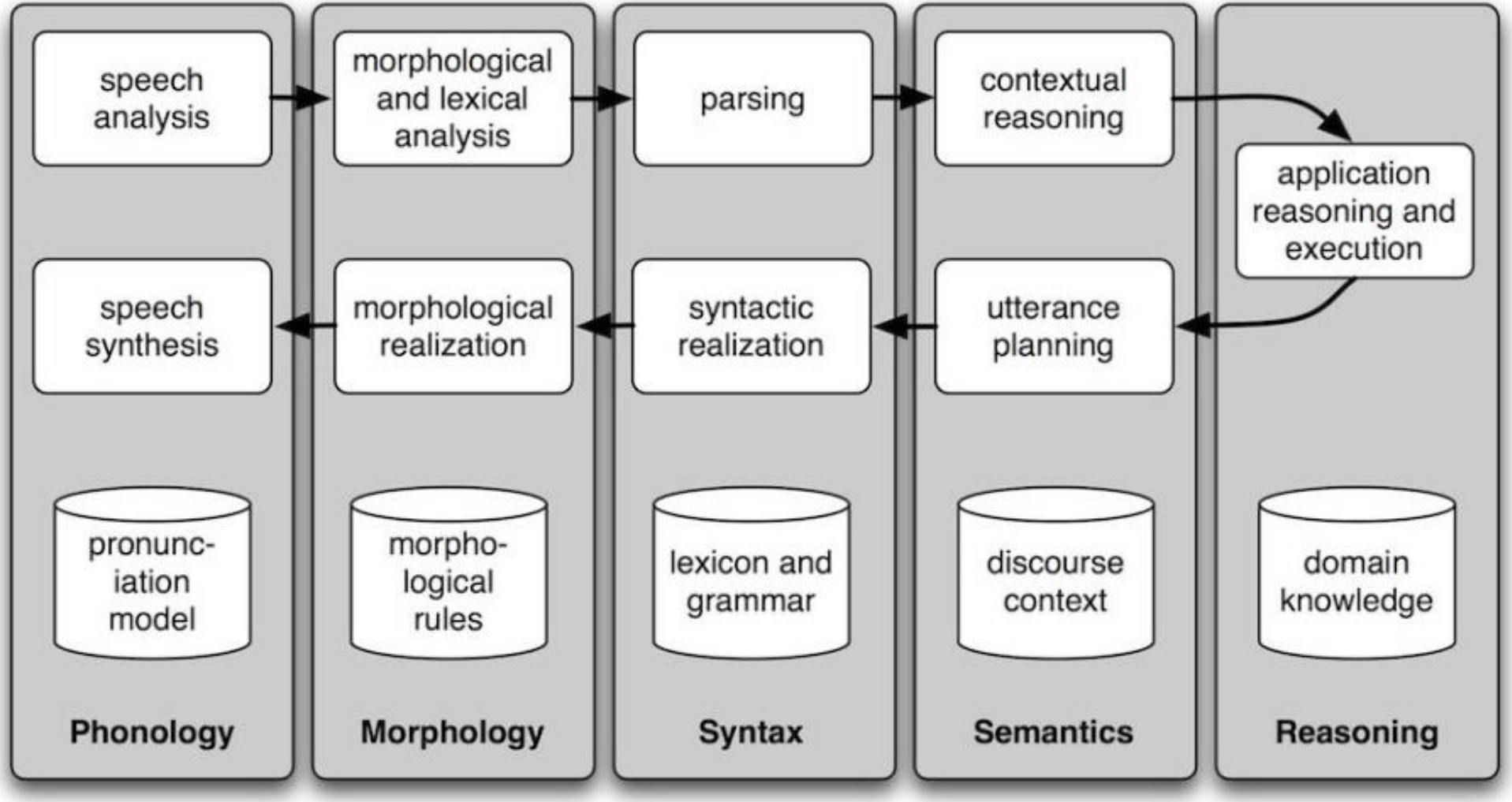
แปลภาษา

วิเคราะห์ความรู้สึก

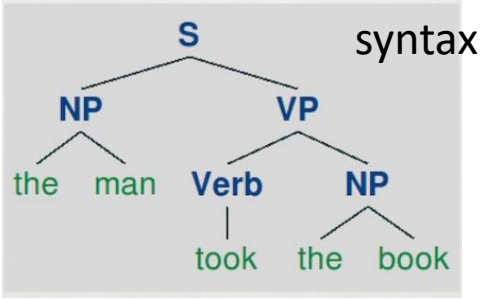
แบ่งเอกสารตามหัวข้อ

ถาม-ตอบ

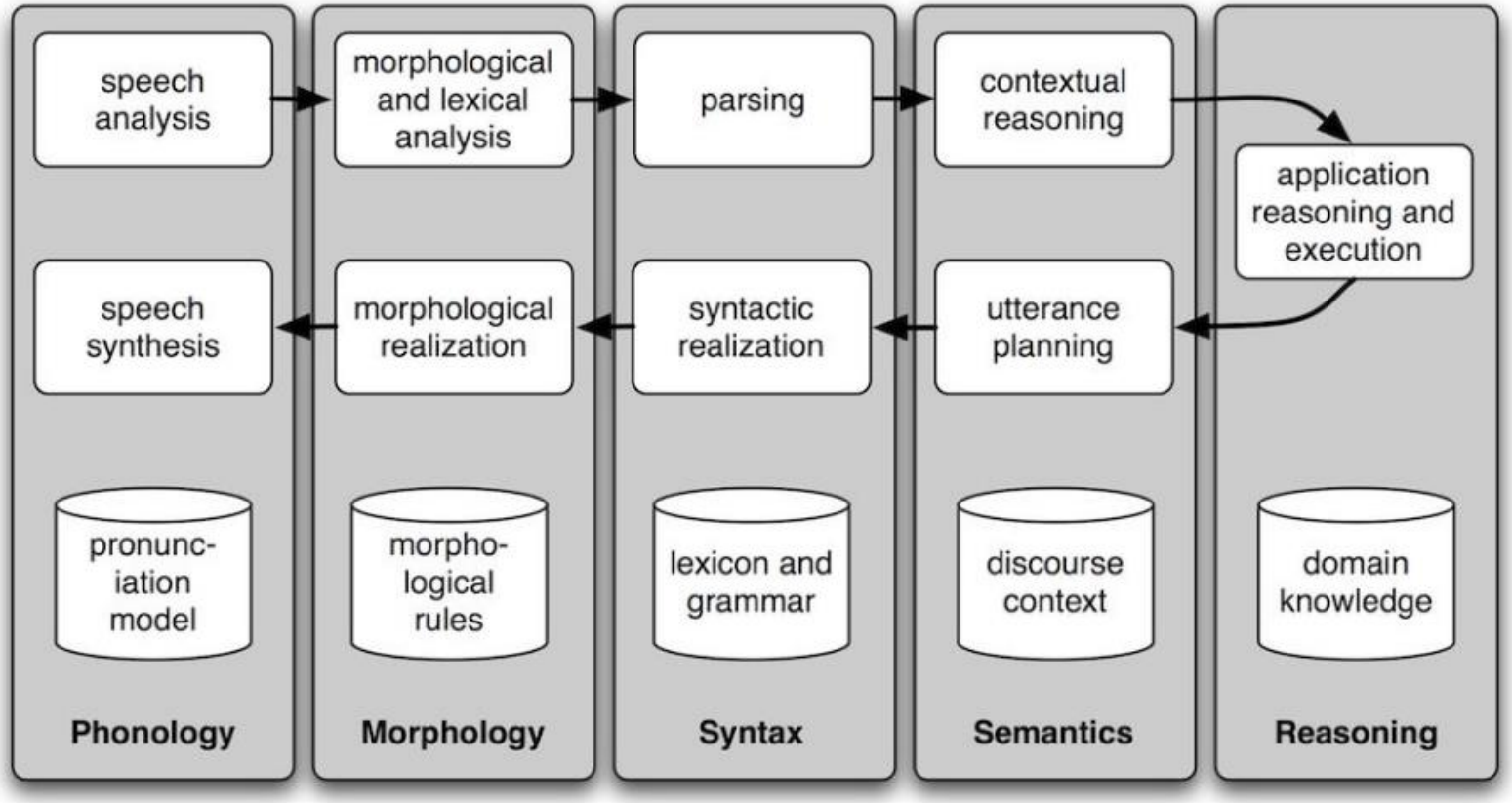
NLP pipeline



NLP pipeline



syntax structure



Stemming
Lemmatization
Segmentation

grammar
parsing
part-of-speech

semantic web
topic
sentiment
knowledge

NLP - Data Science vs AI/ML Research

- ในฐานะของนักวิจัย AI/ML งานทาง NLP คือ การออกแบบโมเดลหรืออัลกอริทึมที่สามารถทำ NLP task ต่าง ๆ ได้อย่างถูกต้องและแม่นยำ

- ในขณะเดียวกัน -

- ในฐานะของนัก Data Scientist/Engineer งานทาง NLP จะเน้นการประมวลข้อมูล raw text (ดูดมาจากเว็บหรือฐานข้อมูล) เพื่อสกัดเอาข้อมูลที่มีค่าออกมา แล้วนำมาสร้างเป็น Application หรือเพื่อตอบคำถามอะไรสักอย่าง
  - เช่น หา sentiment ของลูกค้าต่อ product โดยดูจาก comment ใน Facebook

Python NLP libraries

- NLTK
- TextBlob
- Spacy
- Gensim
- Stanford's CoreNLP
- BeautifulSoup

Text Classification problem

- ปัญหา text classification แบบ supervised คือการหาฟังก์ชัน f : Docs->labels เช่น
  - f : UserComments -> {happy, sad}
  - f : NewsArticles -> {economics, politics, sport}
- ลักษณะของฟังก์ชัน f ขึ้นกับ model ที่เราเลือกใช้ เช่น
  - Naïve Bayes
  - SVM
  - Neural Network

Lab: Text Classification (20newsgroup dataset)

https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

- ใช้ SVM
- ใช้ค่า TF-IDF ของคำ บ่งบอกถึงความสำคัญของคำนั้น ต่อเอกสารนั้น
  - TF(Term Frequency) = ความถี่ของคำที่พบในเอกสารนั้น
  - IDF(Inverse Document Frequency) = อินเวอร์สของความถี่ของคำในทุกเอกสารใน corpus
- training data คือค่า TD-IDF ของทุกคำในทุกเอกสาร
  - เก็บในรูปของ document term matrix

Word Representation

- Solution 1: represent word ด้วย one-hot vector ขนาด n
  โดยที่ n คือจำนวนคำศัพท์ทั้งหมดใน dictionary
  - dog -> [0 0 0 1 …….. 0 0 0]
  - loves -> [1 0 0 0 …….. 0 0 0]
  - coffee -> [0 0 0 0 …….. 0 1 0]
- represent document ด้วย Bag-of-Words
  - dog loves coffee -> [1 0 0 1 …….. 0 1 0]

- **ข้อเสีย:** ไม่นับจำนวนครั้งที่ปรากฏ, ลำดับก่อนหลังไม่มีผล, ไม่บ่งบอกถึงคำที่คล้ายกัน

Word Representation

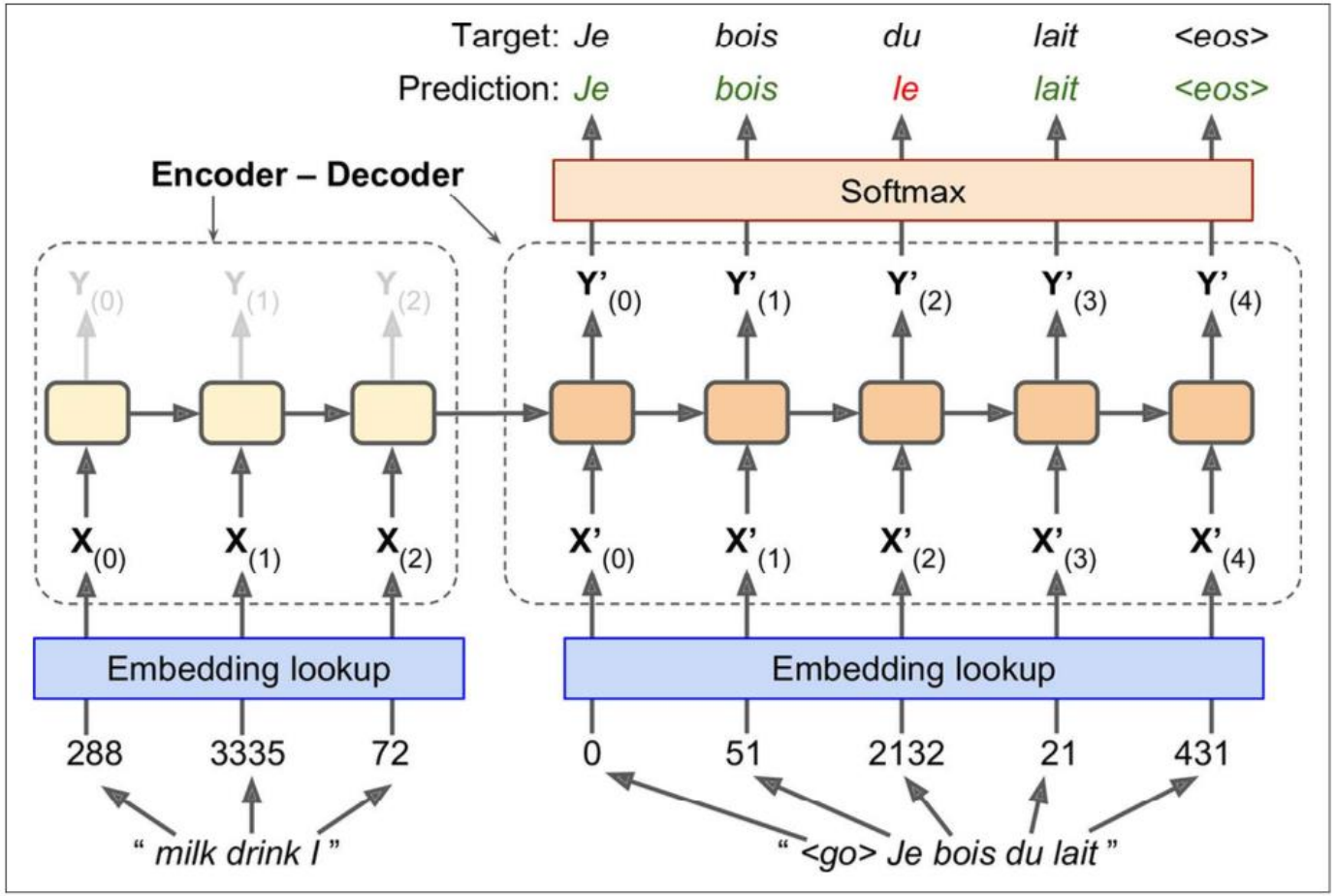
- Solution 2: ใช้ Word Embedding (word2vec)
  - represent word ด้วย vector ขนาดเล็ก (เช่น ขนาด 300)
  - ต้องการให้คำที่คล้ายคลึงกัน มี vector ดังกล่าวที่คล้ายกันด้วย
  - ใช้ Neural Network ในการหา
  - อ่านเพิ่มเติมที่ https://www.tensorflow.org/tutorials/representation/word2vec

Lab:

- ใช้ pre-trained word embedding จาก Google News
  - https://drive.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM&export=download
- ทำตามอาจารย์

Machine Translation with Encoder-Decoder Network (seq2seq)

- https://google.github.io/seq2seq/



Interesting Applications

- Generating fake YouTube comments with char-RNN
  - https://www.youtube.com/watch?v=oJeOvjJmKQ8