Week 1 – Intro and TF-IDF

EGCO467 Natural Language and Speech Processing

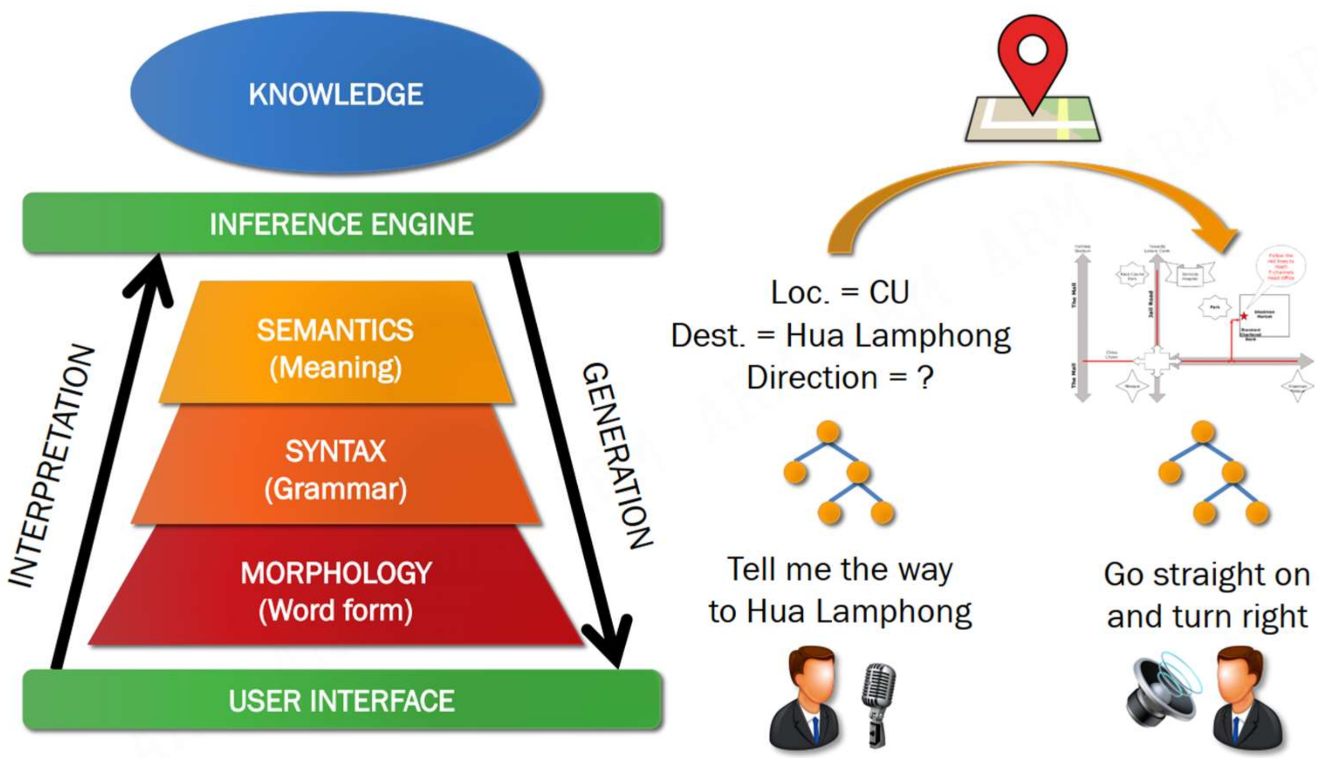
Topics covered (tentative)

- tf-idf
- word embedding
- neural networks / DL
- sentence classification
- language models
- NER/POS
- seq2seq models/attention models
- BERT and transformer models
- question answering models
- machine translation

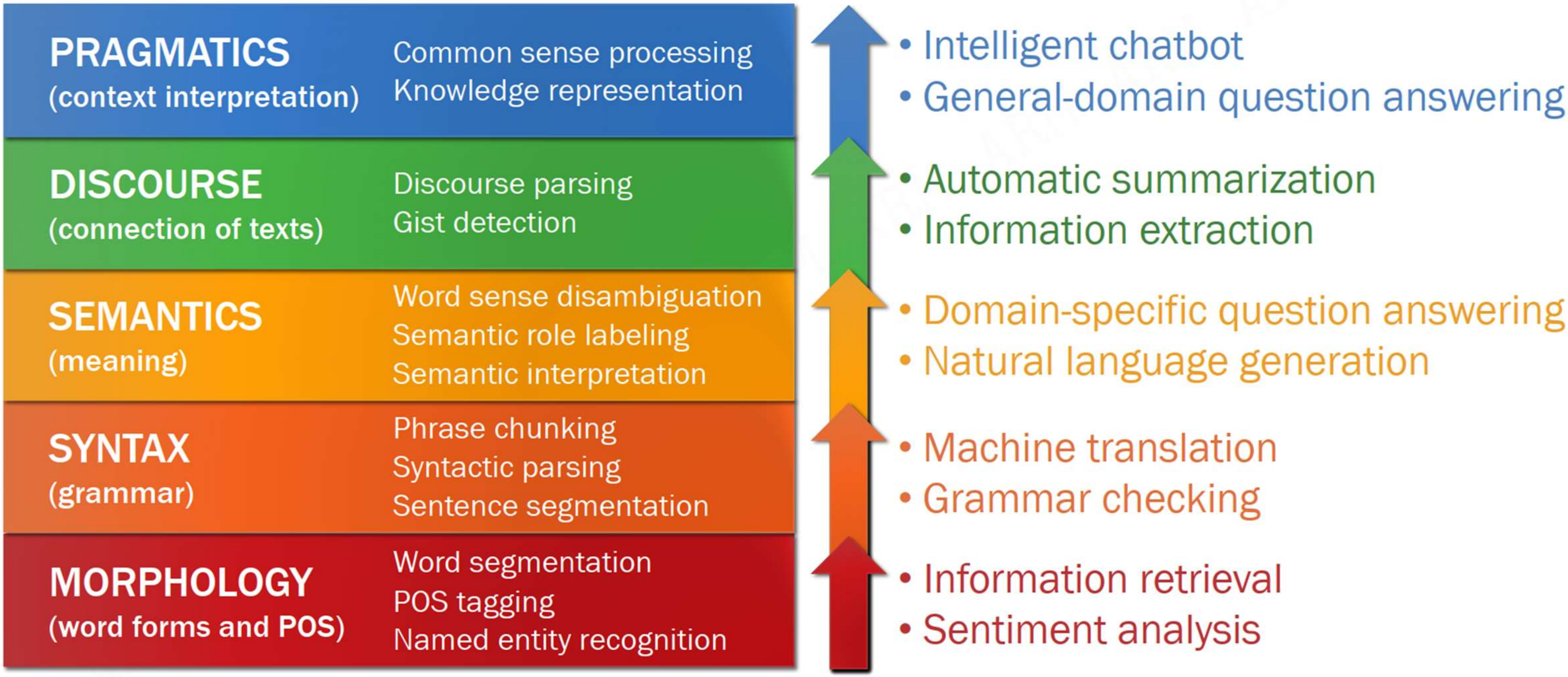
evaluation

- homework 30%
- project 70%

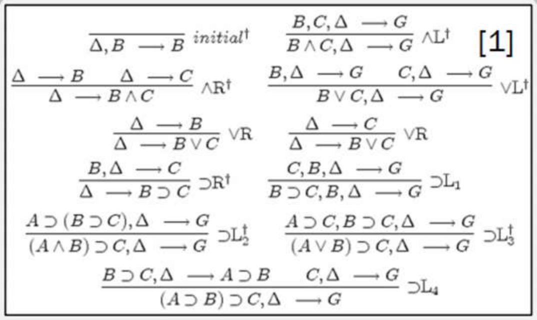
Natural Language Processing



Levels of NLP

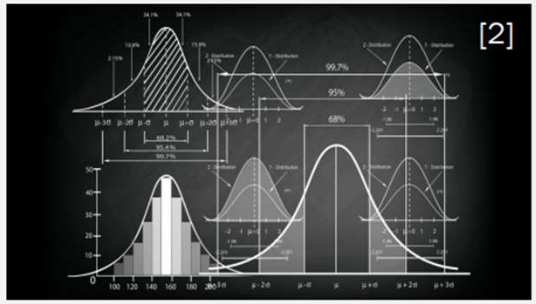


Eras of NLP



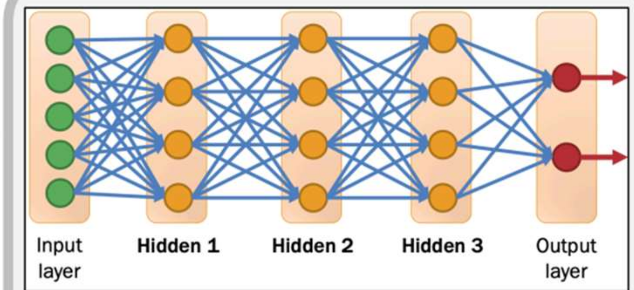
Symbolic Logic
(1949-1989)

- Handcrafted rules and constraints for logical deduction
- Computable knowledge representation
- Theoretical upper bound of language processing



Statistical Methods
(1989-2009)

- Learning to generalize from a large dataset
- Explicit features required
- *"Every time I fire a linguist, the performance of the speech recognizer goes up."* (F. Jelinek, 1985)



Deep Learning
(2009-now)

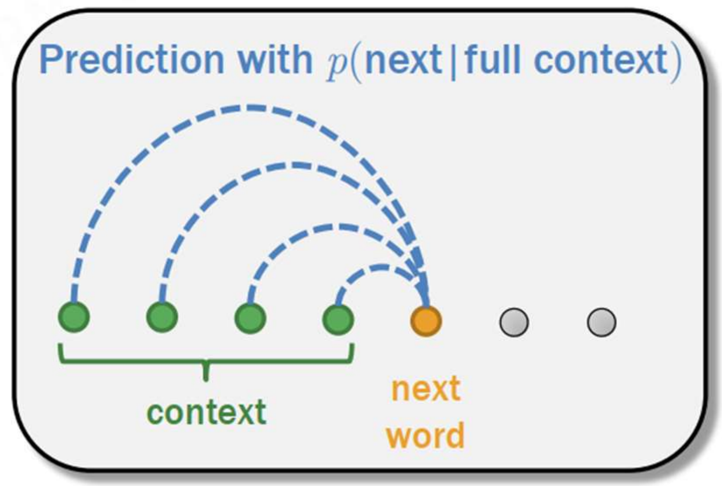
- Extracting features automatically from a very large dataset
- Yielding very high accuracy
- Requiring **powerful** hardware
- Lacking explainability because the models are rather treated as a blackbox

[1] https://philoslife.wordpress.com/2017/05/22/course-note-symbolic-logic-uses/
[2] https://online.stanford.edu/courses/hrp259-introduction-probability-and-statistics-epidemiology

Language Models

- Interpreted as a generative model
  1. Generate the first word w_1
  2. Keep generating the **next word** w_k based on the previous words (a.k.a. **context**) w_1 ... w_{k-1} until the whole sentence of length N is produced

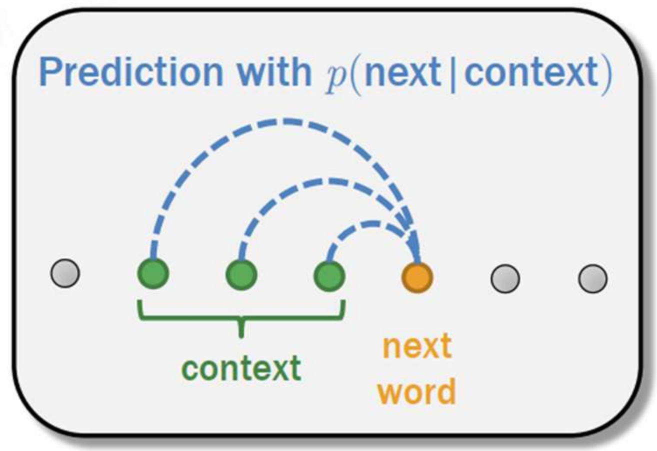
$$P(w_1 \ldots w_N) = p(w_1) \prod_{k=2}^{N} p(\overbrace{w_k}^{\text{next word}} | \overbrace{w_1 \ldots w_{k-1}}^{\text{context}})$$



n-gram models

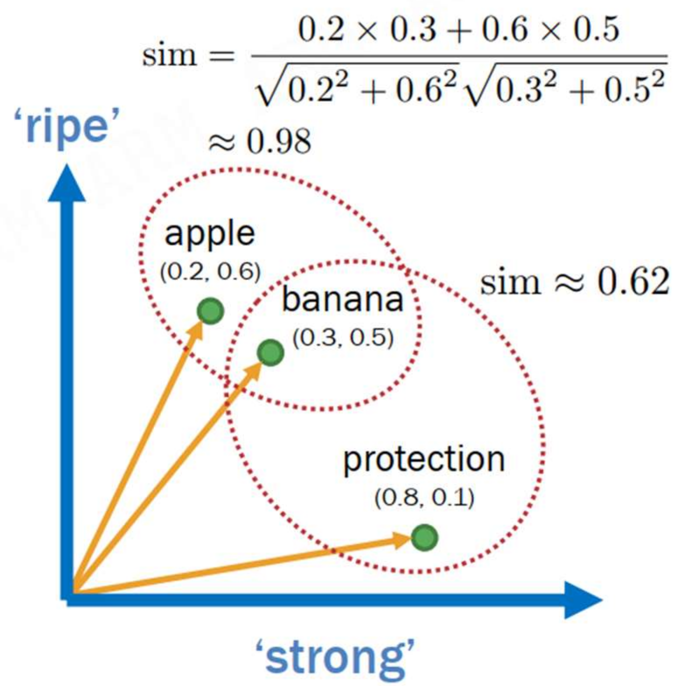
- Language models whose context is truncated to at most n–1 previous words

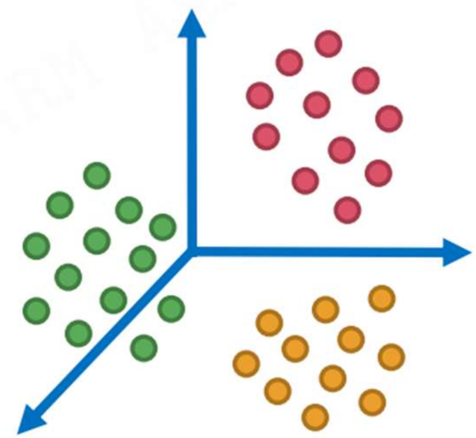
$$P(w_1 \ldots w_N) = p(w_1) \prod_{k=2}^{N} p(\overbrace{w_k}^{\text{next word}} | \overbrace{w_{k-n+1} \ldots w_{k-1}}^{n\text{–}1 \text{ prev words}})$$



- **Unigram** $(n=1)$: $P(w_1 \ldots w_N) = \prod_{k=1}^{N} p(w_k)$
- **Bigram** $(n=2)$: $P(w_1 \ldots w_N) = p(w_1) \prod_{k=2}^{N} p(w_k | w_{k-1})$
- **Trigram** $(n=3)$: $P(w_1 \ldots w_N) = p(w_1) p(w_2 | w_1) \prod_{k=3}^{N} p(w_k | w_{k-2}, w_{k-1})$

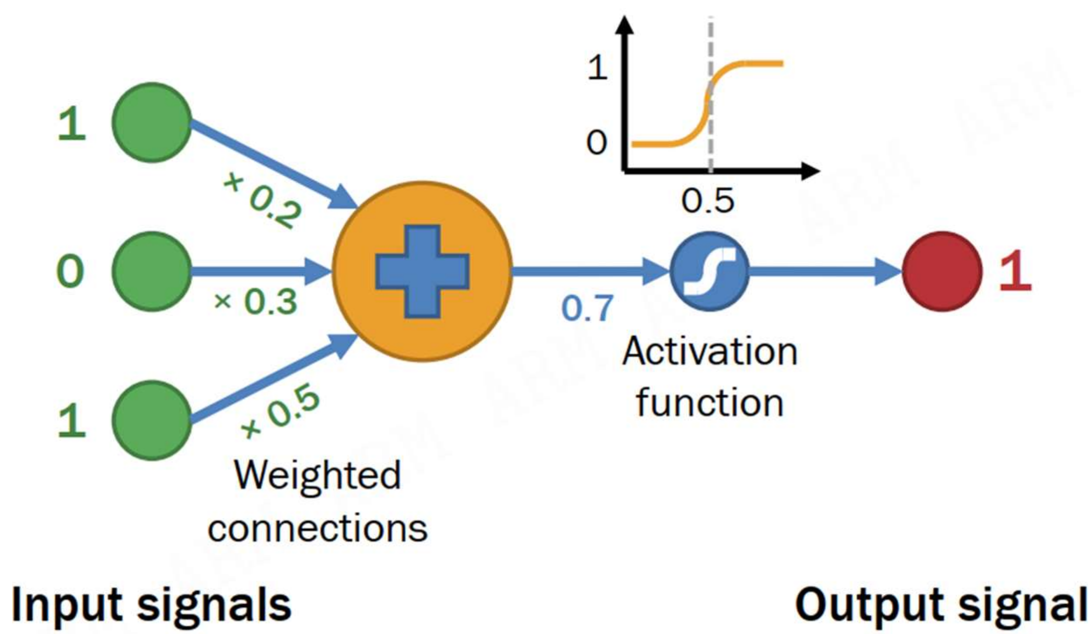
Representing Words





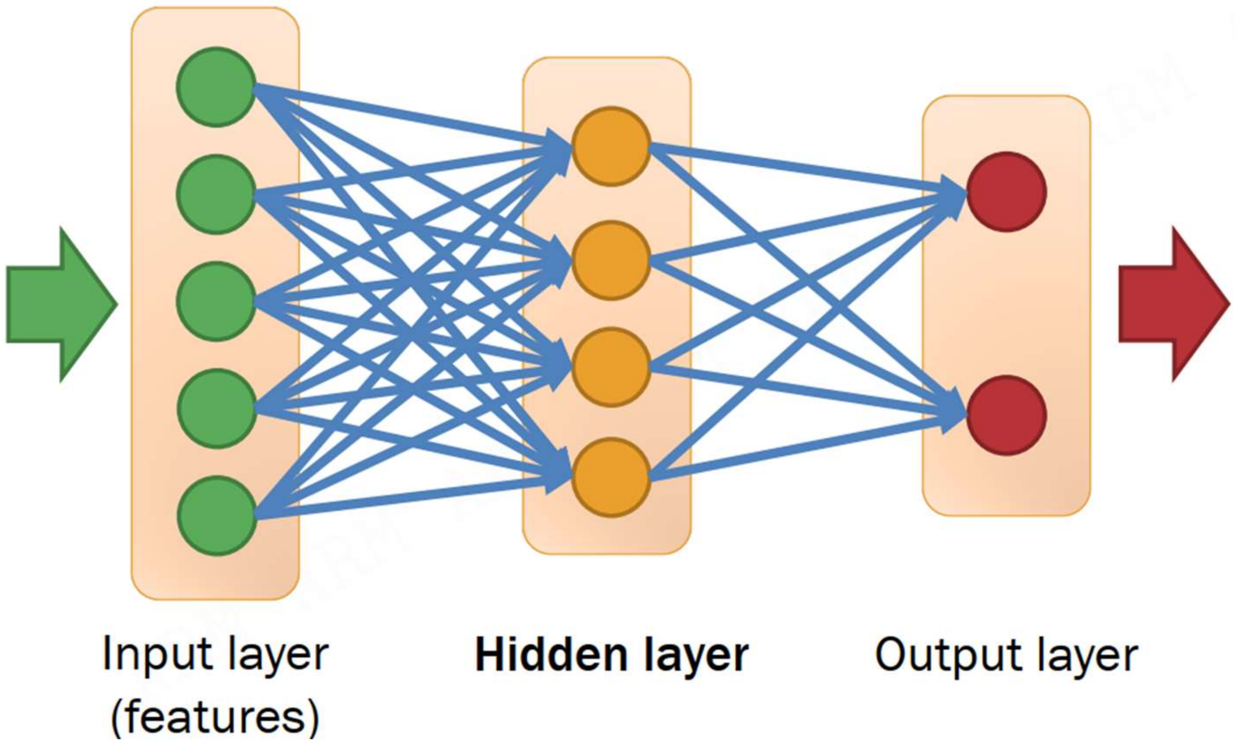
Vector Space Model

Perceptron

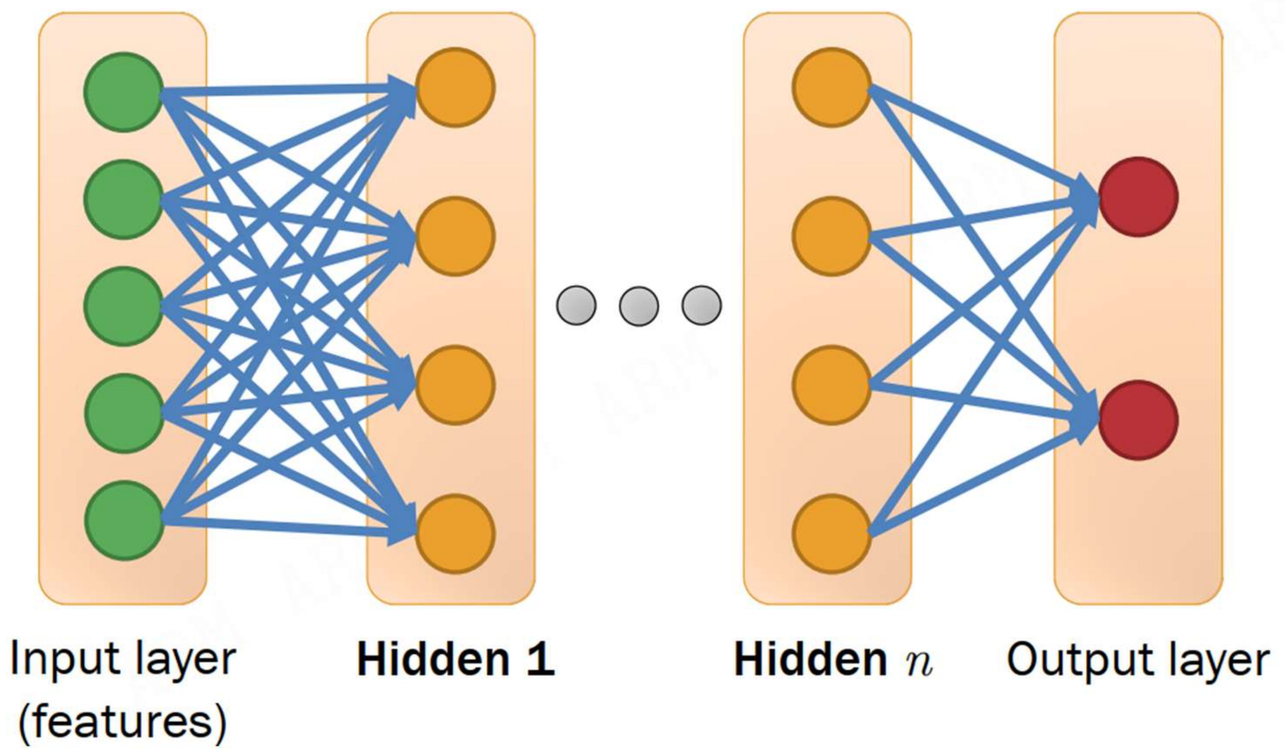


1
0
1
× 0.2
× 0.3
× 0.5
Weighted
connections
0.7
1
0
0.5
Activation
function
1
Input signals
Output signal

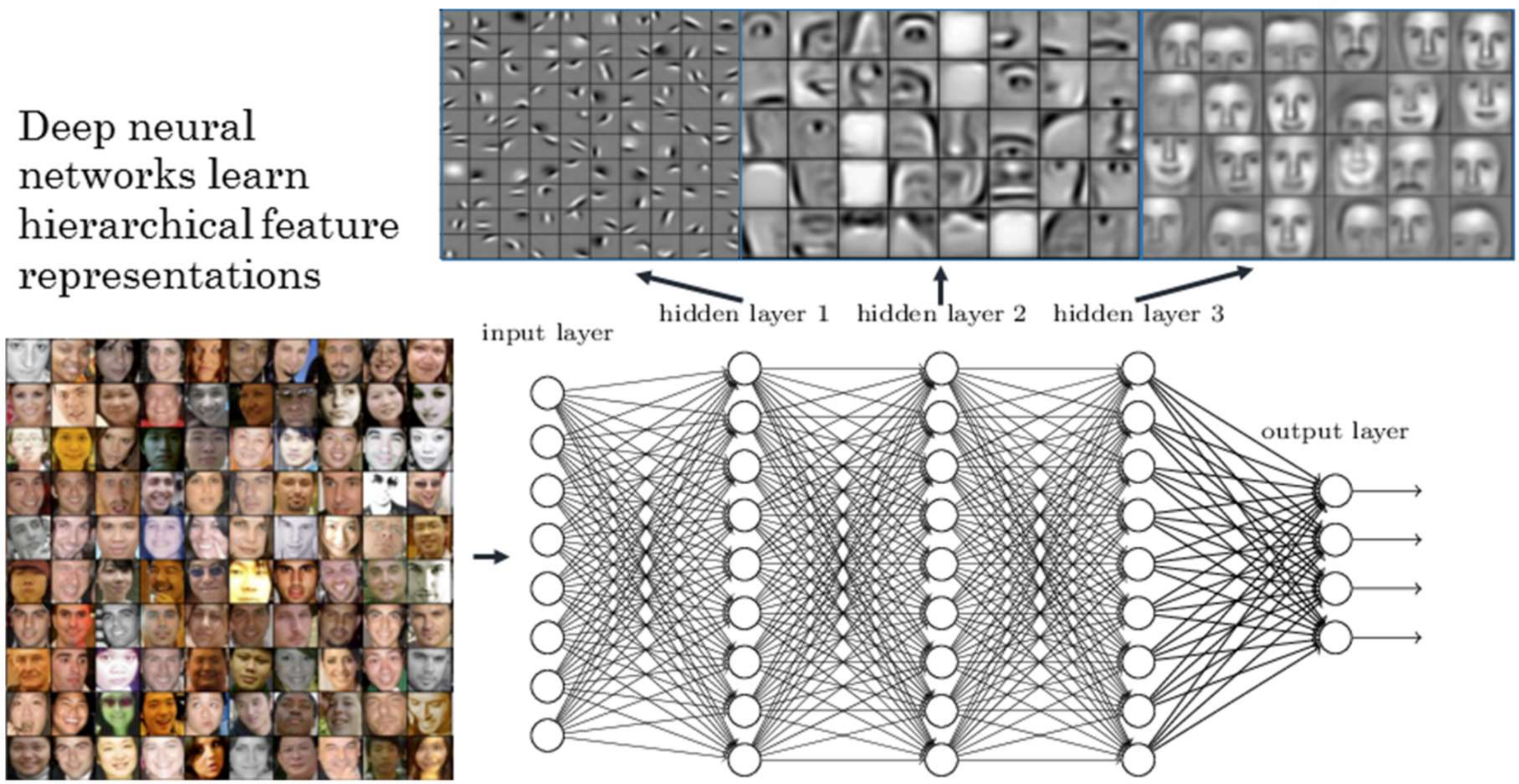
Multi-Layer Perceptron



Deep Neural Network



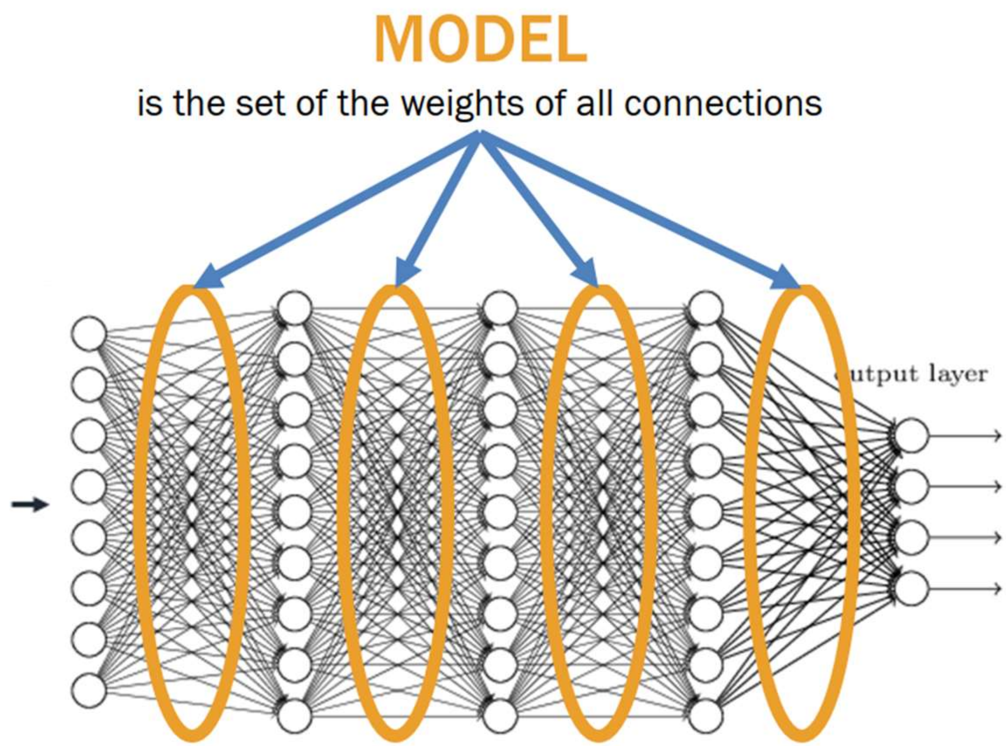
Learning Representation



http://www.strong.io/blog/deep-neural-networks-go-to-the-movies

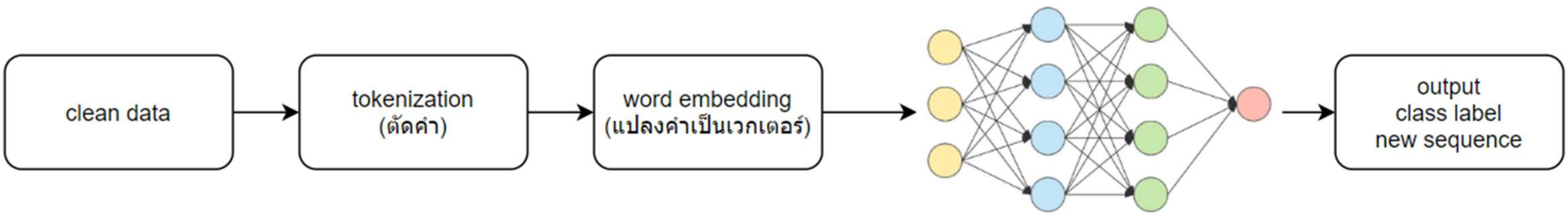
Learning Representation

Deep neural networks learn hierarchical feature representations



http://www.strong.io/blog/deep-neural-networks-go-to-the-movies

Workflow for NLP



Libraries

- Python
- Spacy https://spacy.io/
- Gensim https://radimrehurek.com/gensim/
- scikit-learn https://scikit-learn.org/stable/index.html
- Tensorflow/Pytorch
- Huggingface https://huggingface.co/transformers/

TF IDF

(Term frequency inverse document frequency)

TF-IDF

- TF = term frequency
- IDF = inverse document frequency
- TF-IDF = TF x IDF

Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." *Information processing & management* 24.5 (1988): 513-523.

Definition

- document = 1 piece of text *d*
- corpus = set of documents $D = \{d_1, d_2, \ldots d_N\}$
- term = a word in a document

Example TF

- d_1 = ฉันชอบกินข้าวมันไก่ร้านนี้ -> ฉัน ชอบ กิน ข้าว มัน ไก่ ร้าน นี้
- d_2 = ฉันไม่ชอบหนังเรื่องนี้เลย -> ฉัน ไม่ ชอบ หนัง เรื่อง นี้ เลย

term\document	tf(t,d1)	tf(t,d2)
ฉัน	1	1
ชอบ	1	1
กิน	1	0
ข้าว	1	0
มัน	1	0
ไก่	1	0
ร้าน	1	0
นี้	1	1
ไม่	0	1
หนัง	0	1
เรื่อง	0	1
เลย	0	1

TF has many forms

term\document	tf(t,d1)	tf(t,d2)
ฉัน	1	1
ชอบ	1	1
กิน	1	0
ข้าว	1	0
มัน	1	0
ไก่	1	0
ร้าน	1	0
นี้	1	1
ไม่	0	1
หนัง	0	1
เรื่อง	0	1
เลย	0	1

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

IDF

term\document	tf(t,d1)	tf(t,d2)	มีอยู่ในกี่ document
ฉัน	1	1	2
ชอบ	1	1	2
กิน	1	0	1
ข้าว	1	0	1
มัน	1	0	1
ไก่	1	0	1
ร้าน	1	0	1
นี้	1	1	2
ไม่	0	1	1
หนัง	0	1	1
เรื่อง	0	1	1
เลย	0	1	1

$$\mathrm{idf}(t, D) = \log \frac{N}{\boxed{|\{d \in D : t \in d\}|}}$$

term\document	tf(t,d1)	tf(t,d2)	มีอยู่ในกี่ document	idf(t,D)	tf-idf(t,d1,D)	tf-idf(t,d2,D)2
ฉัน	1	1	2	0.0000	0.0000	0.0000
ชอบ	1	1	2	0.0000	0.0000	0.0000
กิน	1	0	1	0.3010	0.3010	0.0000
ข้าว	1	0	1	0.3010	0.3010	0.0000
มัน	1	0	1	0.3010	0.3010	0.0000
ไก่	1	0	1	0.3010	0.3010	0.0000
ร้าน	1	0	1	0.3010	0.3010	0.0000
นี้	1	1	2	0.0000	0.0000	0.0000
ไม่	0	1	1	0.3010	0.0000	0.3010
หนัง	0	1	1	0.3010	0.0000	0.3010
เรื่อง	0	1	1	0.3010	0.0000	0.3010
เลย	0	1	1	0.3010	0.0000	0.3010

$$\mathrm{idf}(t, D) = \boxed{\log \frac{N}{\boxed{|\{d \in D : t \in d\}|}}}$$

IDF also has many forms

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log \left(\frac{N}{1 + n_t} \right) + 1$
inverse document frequency max	$\log \left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t} \right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Concept of TF-IDF

- TF words that appear a lot in a document is indicative of the topic of that document. E.g. the word "interest rate" in an economic news article.
- IDF words that appear in every documents are not indicative of the topic. E.g. common words such as: "the", "a", "that", "is"
- Therefore, words with high TF-IDF values, are words that appear often in *some* documents only such as: cabinet (politics) gold (economic).

Exercise

calculate the TF-IDF table (by hand or using Excel)

- d1 = the man went out for a walk
- d2 = the children sat around the fire

- Use the *raw count* form for TF and the *inverse document frequency* form for IDF.

Solution

term	tf(t,d1)	tf(t,d2)	nt	idf	tf-idf(d1)	tf-idf(d2)
the	1	2	2	0	0	0
man	1	0	1	0.30103	0.30103	0
went	1	0	1	0.30103	0.30103	0
out	1	0	1	0.30103	0.30103	0
for	1	0	1	0.30103	0.30103	0
a	1	0	1	0.30103	0.30103	0
walk	1	0	1	0.30103	0.30103	0
children	0	1	1	0.30103	0	0.30103
sat	0	1	1	0.30103	0	0.30103
around	0	1	1	0.30103	0	0.30103
fire	0	1	1	0.30103	0	0.30103

Homework 1

- Write Python function
  tf_idf(str1, str2)
- Function takes two string. These are the two documents.
- Return the tf-idf table as Numpy array or Pandas DataFrame
- Submit your work as .ipynb file