CIS 6930 Topics in Computing for Data Science
Week 5: NLP Basics & Word Embeddings

9/23/2021

Yoshihiko (Yoshi) Suhara

2pm-3:20pm & 3:30pm-4:50pm

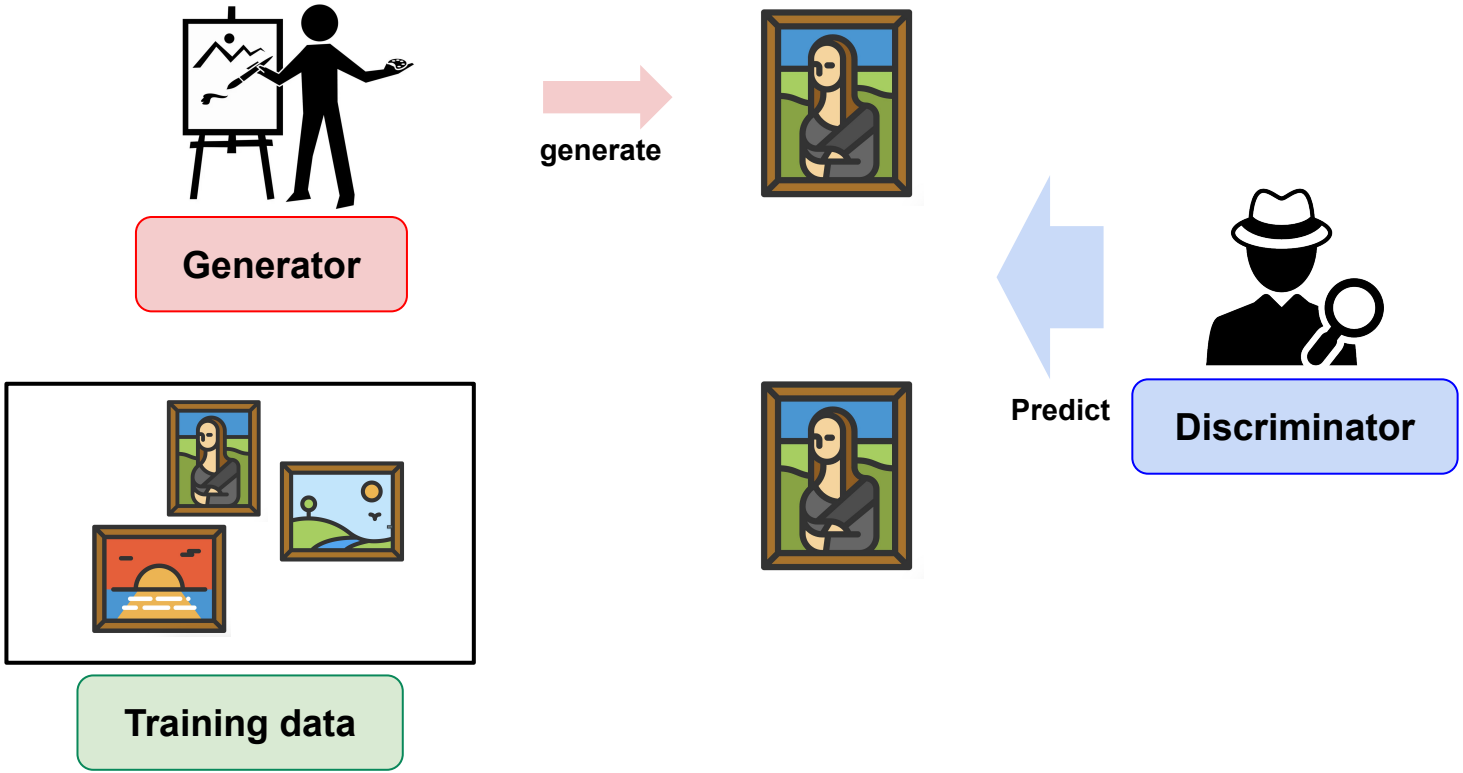
Week 3!

- ~~Week 1: Deep Learning Basics (Thu 9/9)~~
- ~~Week 2: AutoEncoder (Tue 9/14)~~
- ~~Week 3: Convolutional Neural Networks (Thu 9/16)~~
- ~~Week 4: GAN (Tue 9/21)~~
- **Week 5: Word embeddings: Word2vec, GloVe (Thu 9/23)**
- Week 6: Recurrent Neural Networks (Tue 9/28, Thu 9/30)
- Week 7: Review/Project pitch & Mid-term (Tue 10/5, Thu 10/7)
- Fall Break
- ...

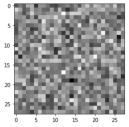
Recap: Generative Adversarial Networks (GANs)

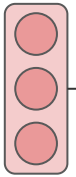
Generative Adversarial Network (GAN)

Generator + Discriminator + Training data

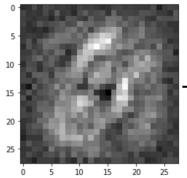


GAN Architecture





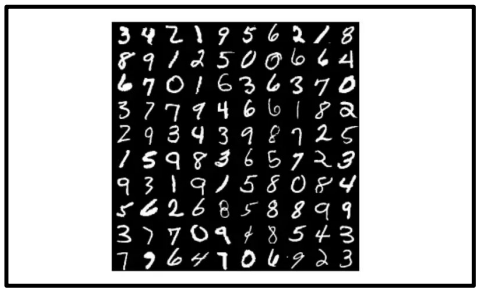
Decoder



Generator

Encoder

Real/Fake

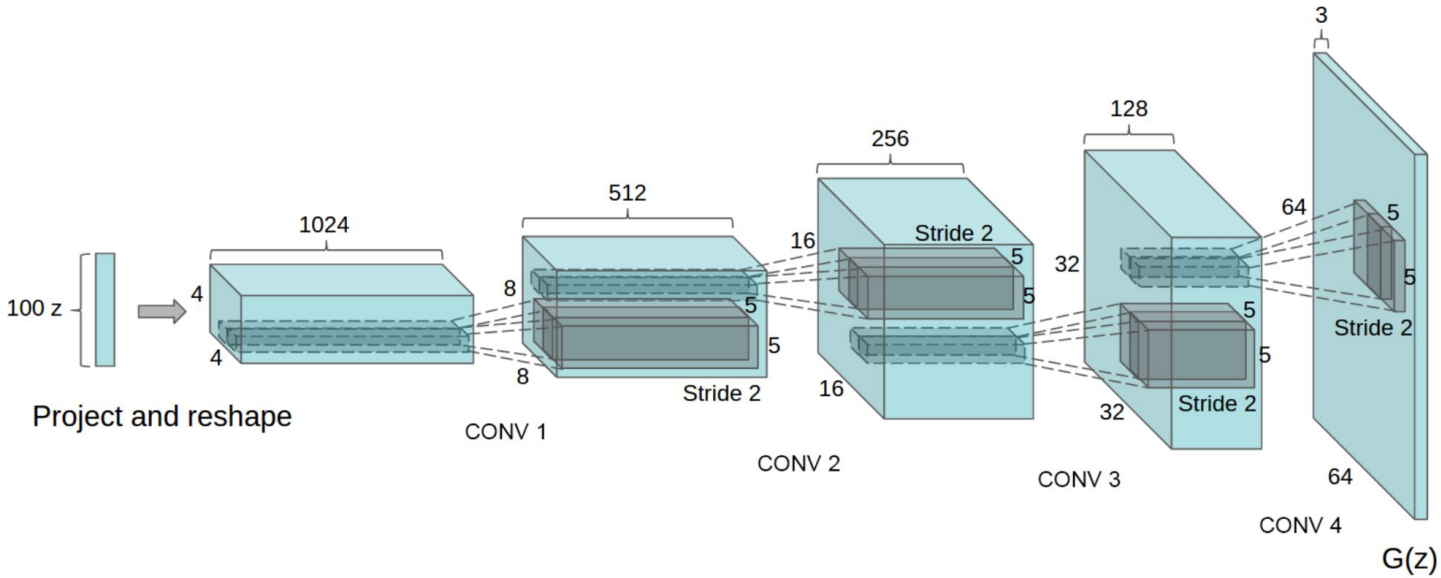


Discriminator

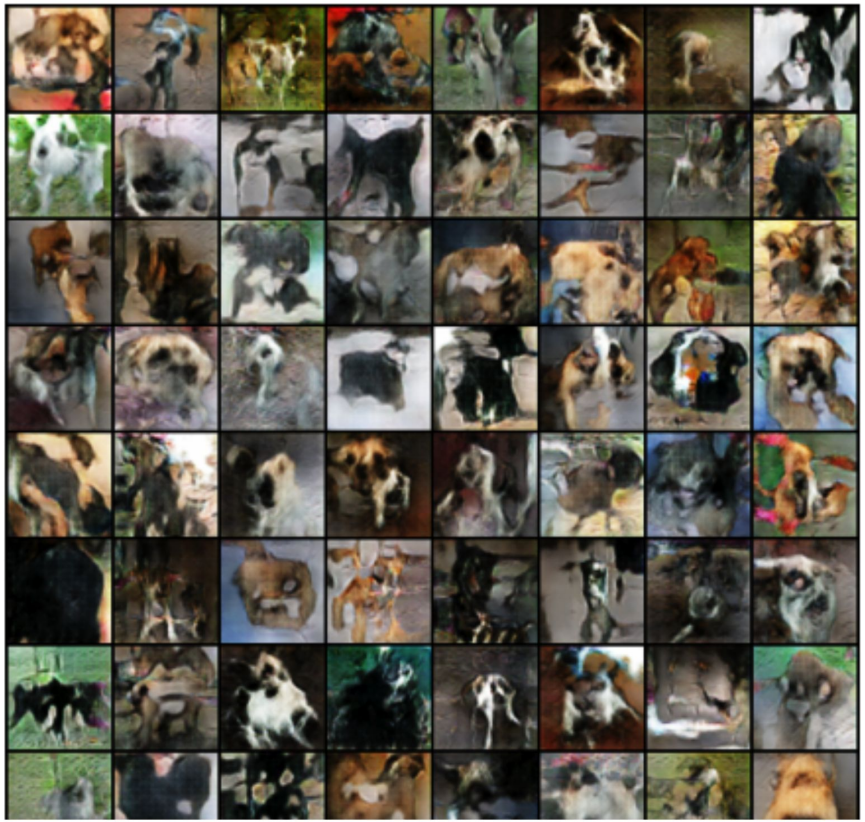
Training data

Deep Convolutional GAN (DCGAN) [Radford et al. 2016]

- The use of (Transposed) Convolutional layers
  - In the analogy of Autoencoder vs. Convolutional Autoencoder



Final Results

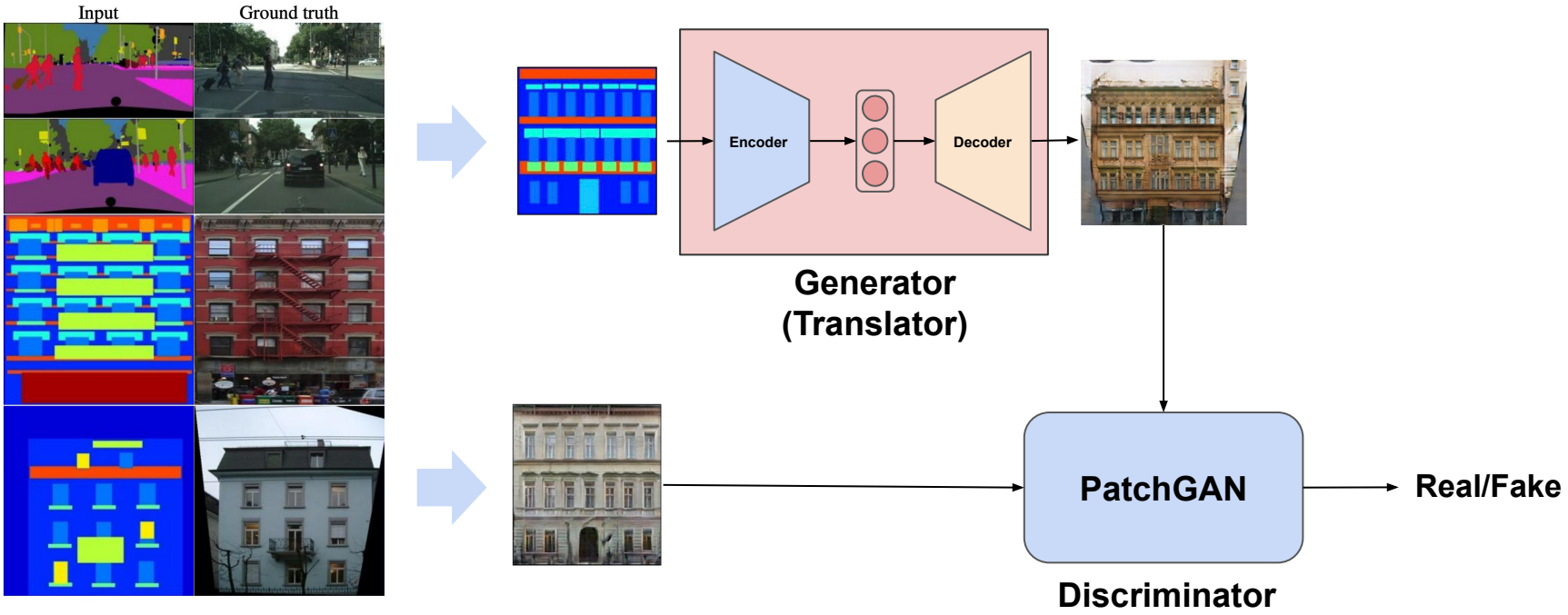


Advanced GANs

- ✅ DCGAN
- ✅ Conditional GAN
- ✅ Pix2Pix
  - ✅ U-Net (Generator)
  - ✅ PatchGAN (Discriminator)
- ✅ CycleGAN

Pix2Pix [Isola et al. 2017]

- U-Net (Generator) + PatchGAN (Discriminator) + Paired Images



CycleGAN: 2x Generator + 2x Discriminator

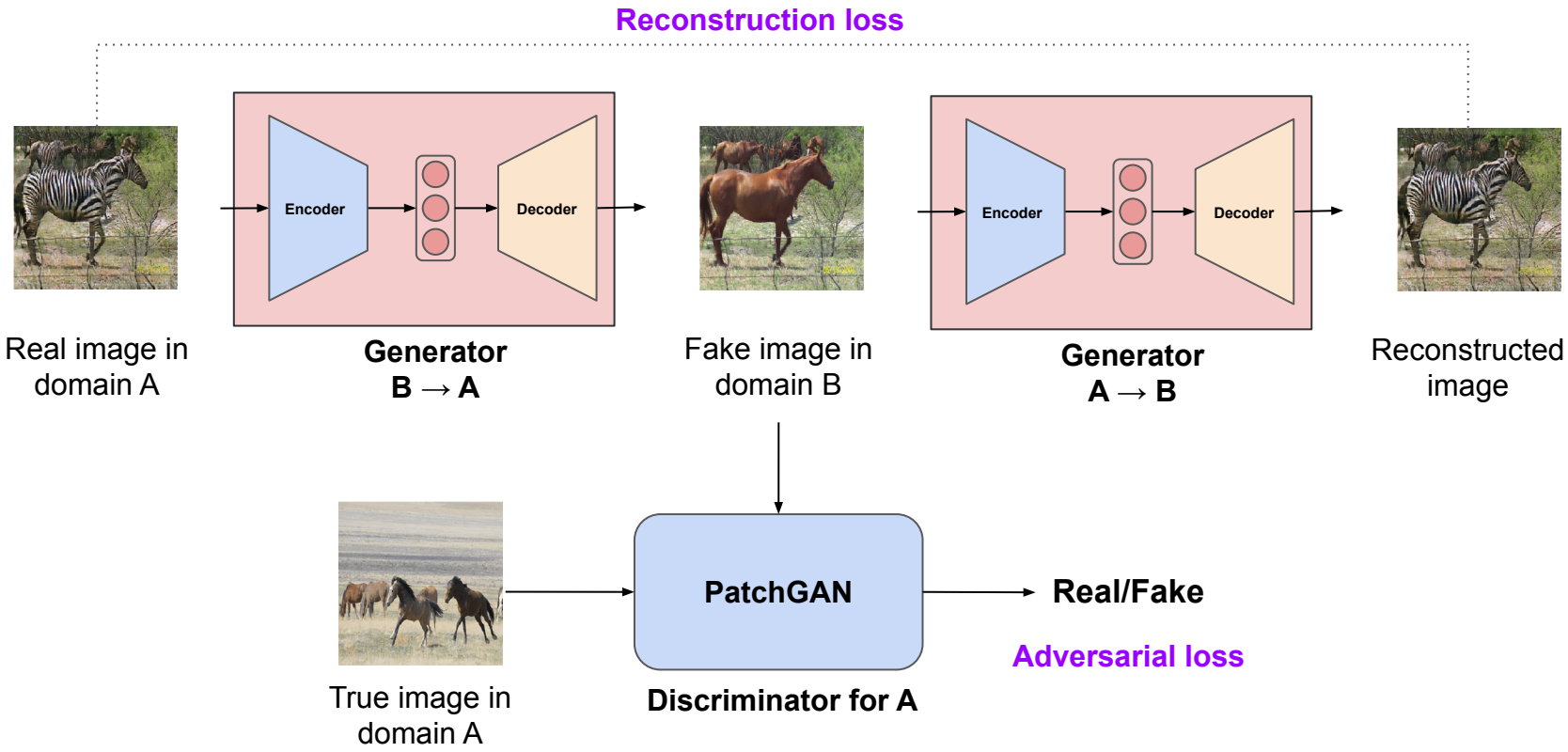
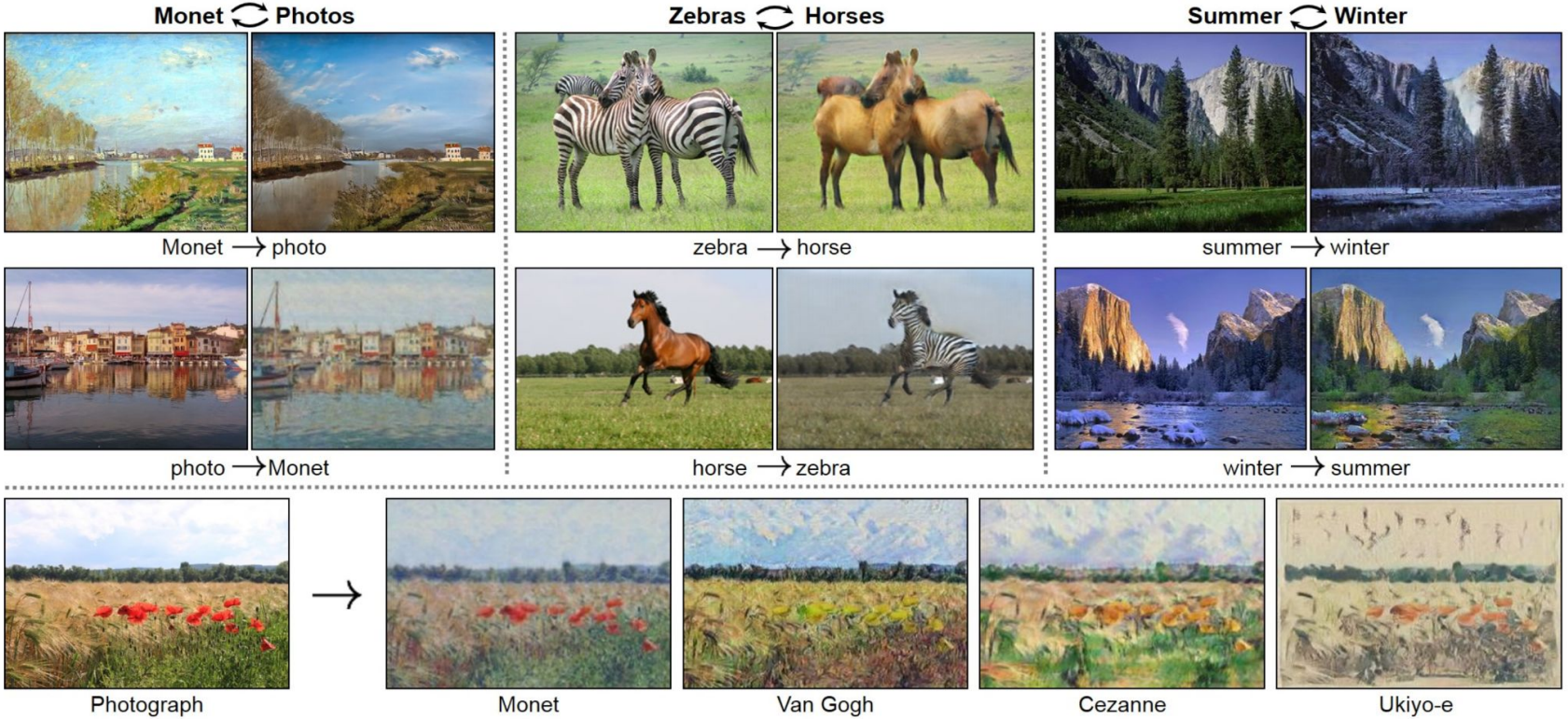


Image Translation by CycleGAN [Zhu and Park et. al. 2017]



GAN Applications

Sketch-to-Image (e.g., pix2pix, CycleGAN)

Image Translation (e.g., CycleGAN)

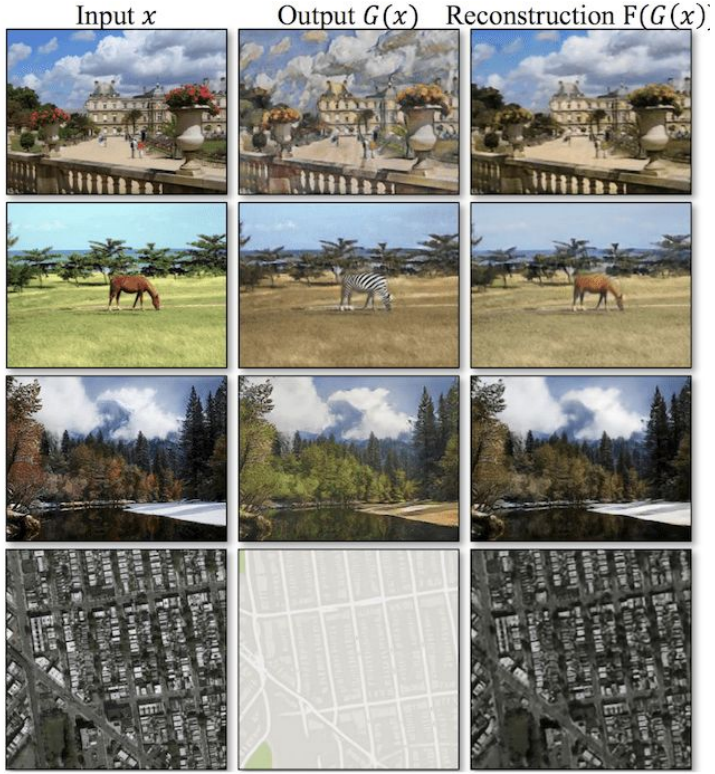
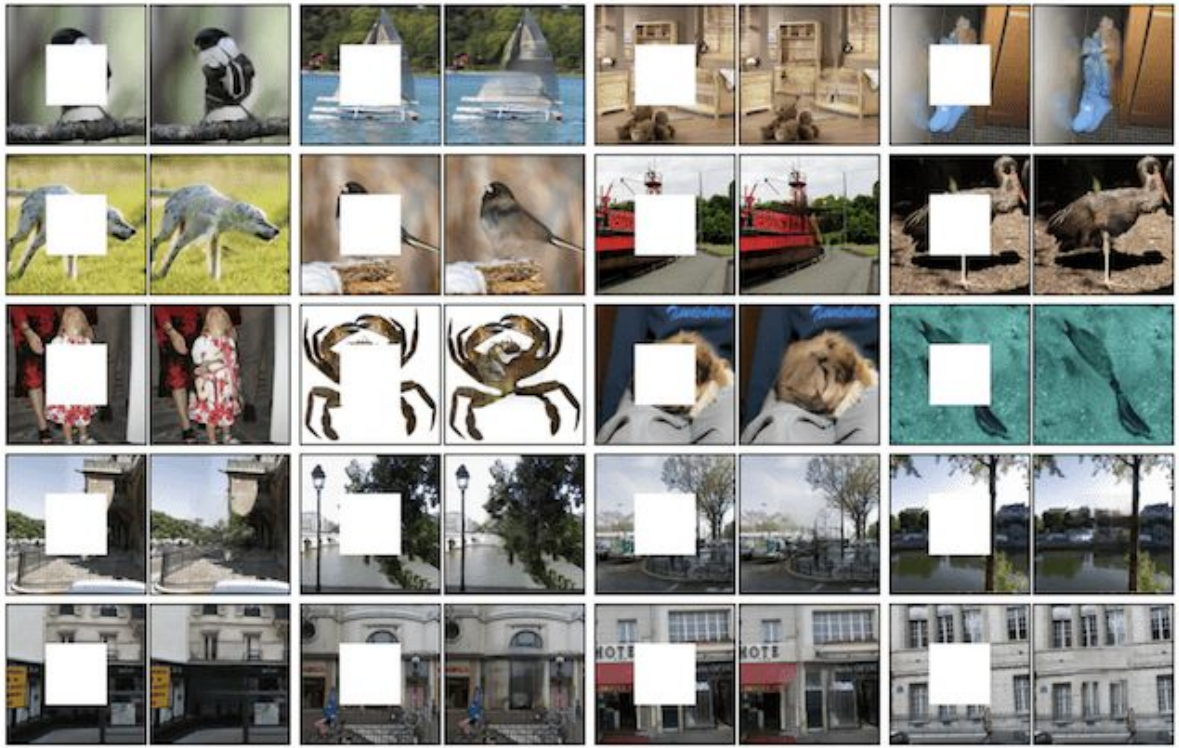


Image De-raining

[Zhang et al. 2017]

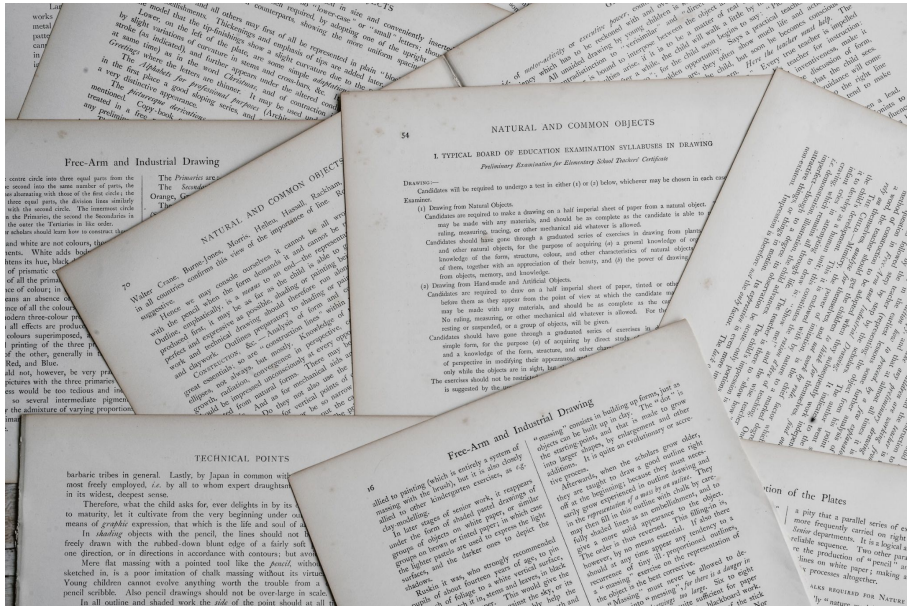
Photo Inpainting



[Pathak et al. 2016]

Questions?

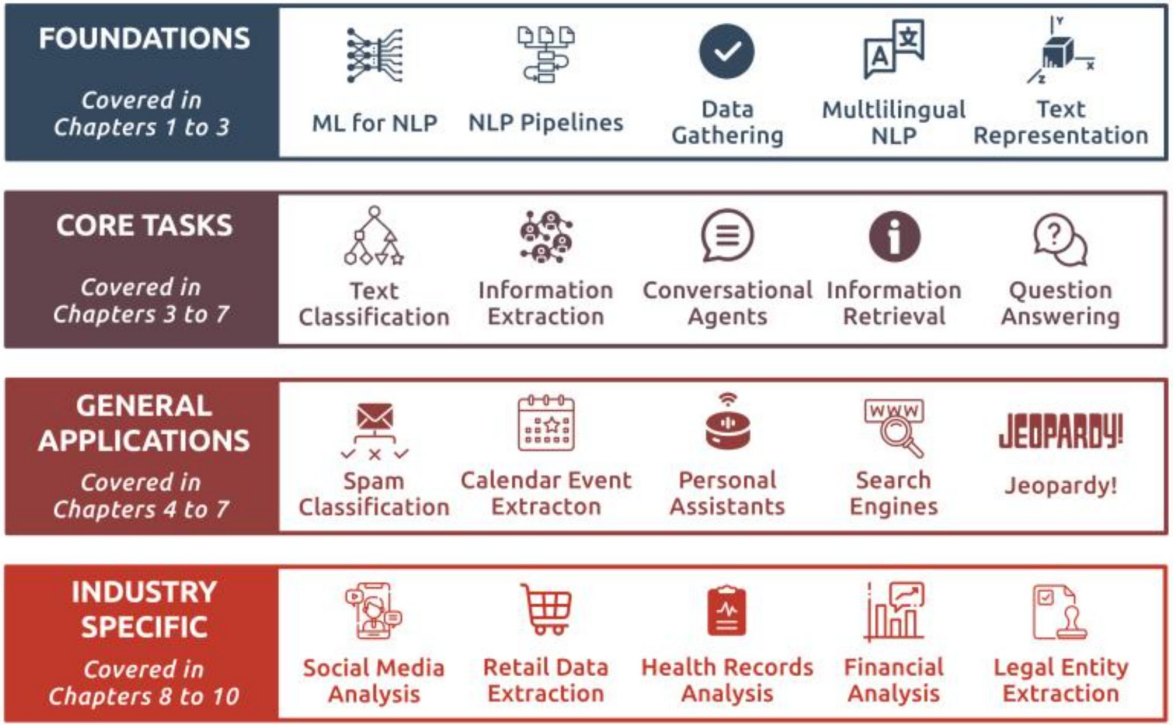
Text data!


NATURAL AND COMMON OBJECTS
Free-Arm and Industrial Drawing
TECHNICAL POINTS

What is NLP?

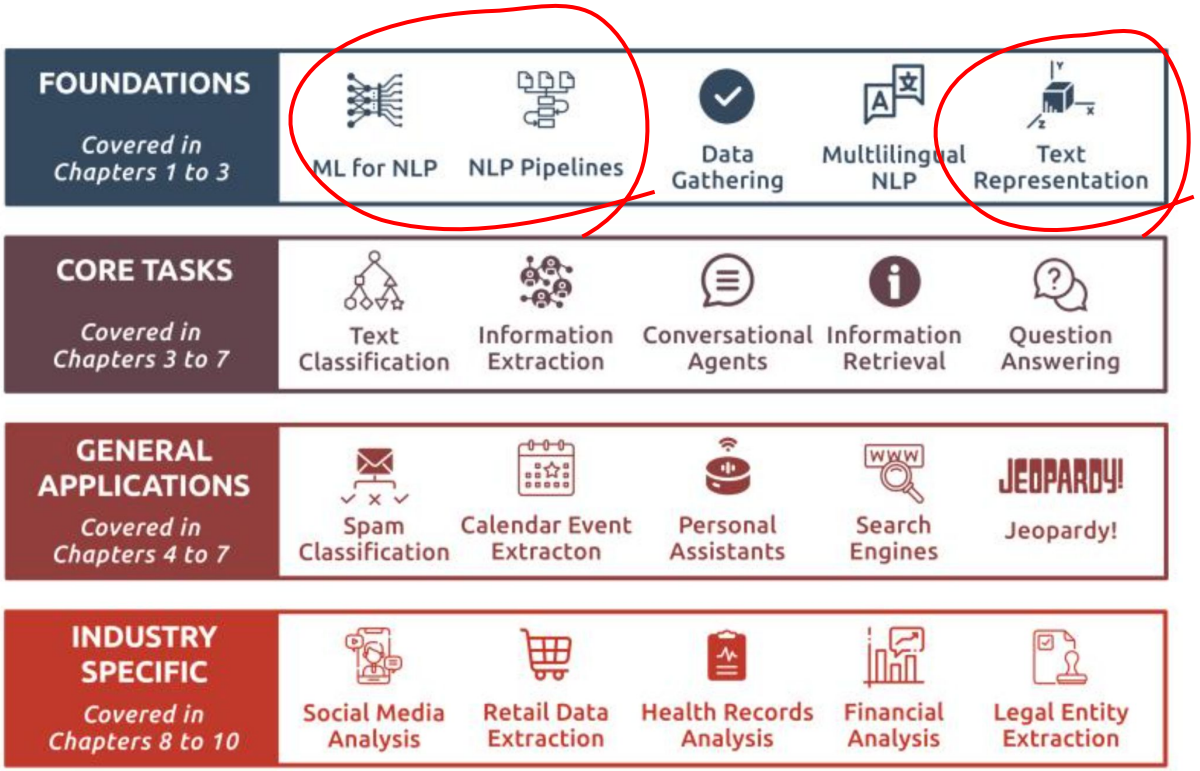
- What do you think?

NLP Tasks and Applications



[Practical Natural Language Processing | O'Reilly]

NLP Tasks and Applications



Practical Natural Language Processing | O'Reilly

Natural Language != English

- That said, most research papers use English as the target language
- **Bender Rule**: “Always name the language(s) you're working on”

- Recent advances in multilingual techniques/low-resources settings

- Natural Language != Text
  - Audio/Speech (will not be covered)

Emily Bender

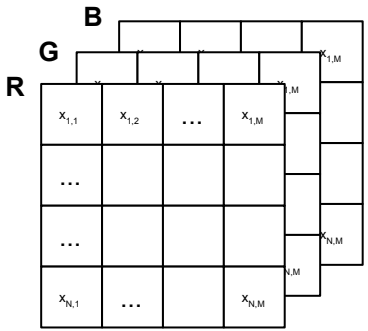
Agenda

- NLP Basics
  - Hands-on session: NLP Pipeline with spaCy
  - Hands-on session: Pandas 101
- Word embeddings
  - Word2vec, GloVe, fastText
- Beyond word embeddings

What Makes Text Processing Difficult?

What Makes Text Processing Difficult for Computers?

- Symbolic representations cannot be directly converted into vector/tensor representations
  - cf. Images



Images

"A cat is starting at me" → ???

Text

One-hot Vector Representation

- Assign one dimension to each word

“A cat and a dog”

a	cat	and	dog		
1	0	0	0	...	0
0	1	0	0	...	0
0	0	1	0	...	0
1	0	0	0	...	0
0	0	0	1	...	0

Vocabulary size

One-hot Vector Representation

- Assign one dimension to each word

“A cat and a dog”

a	cat	and	dog		
1	0	0	0	...	0
0	1	0	0	...	0
0	0	1	0	...	0
1	0	0	0	...	0
0	0	0	1	...	0

Vocabulary size

Dictionary Example

```
token2id = {"a": 1,
            "cat": 2,
            "and": 3,
            "dog": 4,
            …
           }
```

cf. Bag-of-Words Representations (for sentences/documents)

- e.g., count vector

“A cat and a dog”

a	cat	and	dog		
2	1	1	1	...	0

Vocabulary size

Limitations of One-hot Vector Representation

- The vocabulary must be constructed in advance
- No way to represent “unseen” words
  - known as Out-of-vocabulary (OOV) Problem

- Any workaround?

Limitations of One-hot Vector Representation

- The vocabulary must be constructed in advance
- No way to represent “unseen” words
  - known as Out-of-vocabulary (OOV) Problem

- Any workaround?
  - character-based representations
  - sub-token representations (common choice for pre-trained language models)

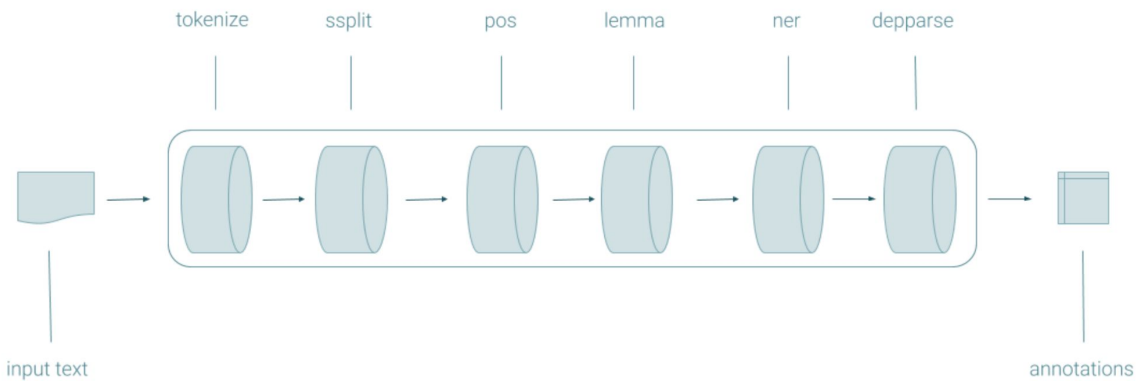
NLP Basics

Traditional NLP Pipeline

- Tokenization
- Sentence splitting
- Lemmatization
- POS tagging
- Chunking
- Dependency parsing

Shallow

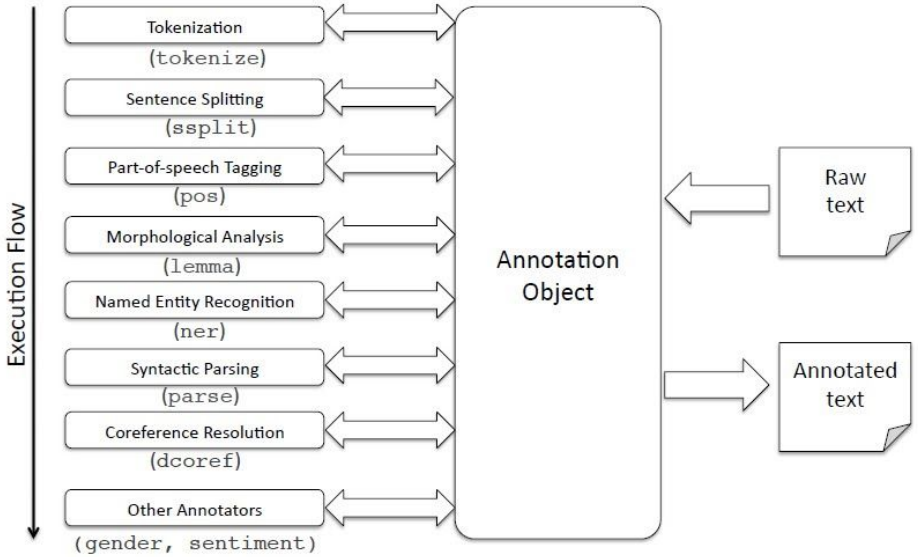
Deep



https://stanfordnlp.github.io/CoreNLP/

NLP Pipeline: Text Processing as Annotation

- Annotating semantics to the input text



https://stanfordnlp.github.io/CoreNLP/

Tokenization

- A process of separating text into a sequence of tokens

```
"A cat is starting at me."

→ ["A", "cat", "is", "staring", "at", "me", "."]
```

Is `"A cat is starting at me.".split(" ")` not sufficient?

Tokenization is NOT Trivial for Some Languages
Japanese Tokenization Example

東京都の人口は約1400万人です。

The population of **Tokyo Metropolis** is about 14 million.

東京　都 (Tokyo Metropolis)
tokyo to

東　京都 (East Kyoto)
higashi kyoto

Lemmatization (cf. Stemming)

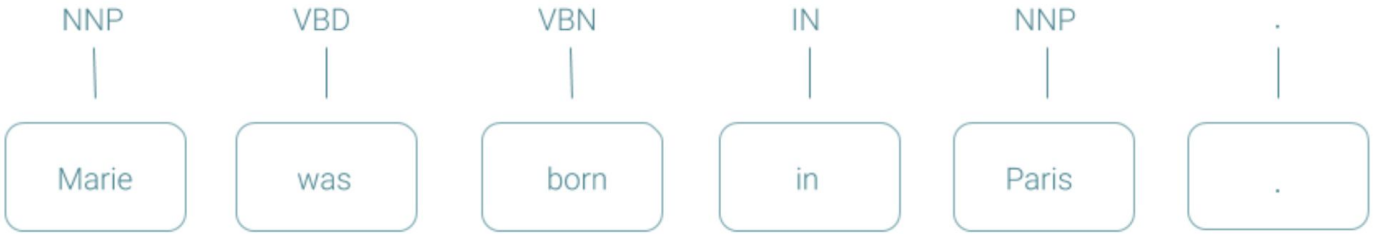
- Lexical normalization
  - Stemming: Rule-based word truncation
  - Lemmatization: Mapping to a word based on morphological analysis

Word	Porter	Lancaster	Lemmatiser
apples	appl	appl	apple
pears	pear	pear	pear
tasks	task	task	task
children	children	childr	child
earrings	ear	ear	earring
dictionary	dictionari	dict	dictionary
marriage	marriag	marry	marriage
connections	connect	connect	connection
universe	univers	univers	universe
universities	univers	univers	university

Introduction to NLP - Part 2: Difference betaween lemmatisation and stemming

Part-of-Speech Tagging

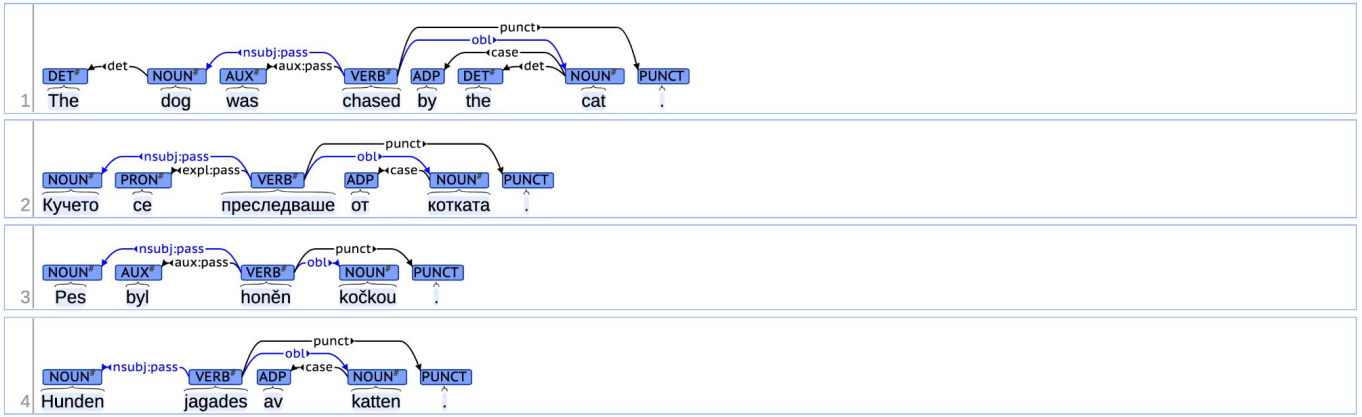
- Meta information of how a word is used in a sentence
  - e.g., nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions, interjections
  - Some tagging schema define more fine-grained POS tags (e.g., NNP = Proper Noun)



Wait. You Said NLP is Not Limited to English!

- Universal Dependencies: A unified annotation framework for any languages

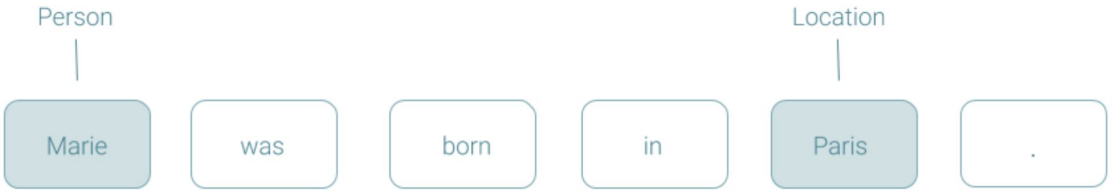
Universal Dependencies



https://universaldependencies.org/

Chunking / Named Entity Recognition (NER)

- Extracting **phrases** and assigning **type** information
- Technically, Chunking != NER, but NER is often formulated as Chunking



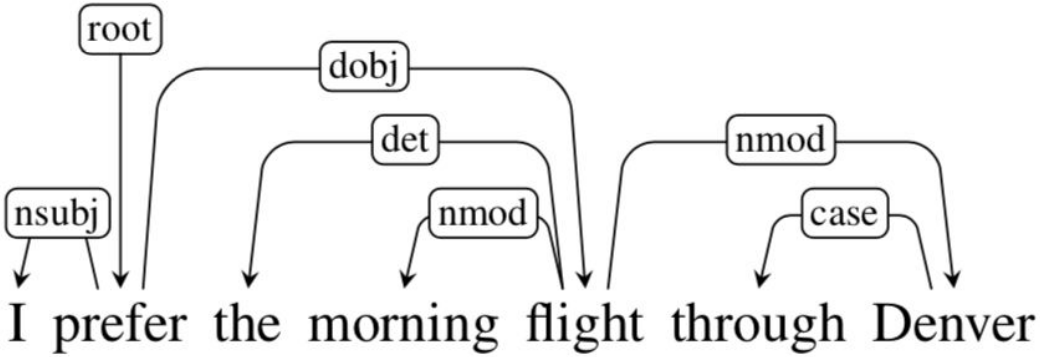
IOB (Inside-Outside-Beginning) Format

- Commonly used annotation format for chunking problems

Alex	**B-PER**
is	O
going	O
to	O
Los	**B-LOC**
Angeles	**I-LOC**
in	O
California	**B-LOC**

Dependency Parsing (aka Syntactic Parsing)

- Extracting the grammatical structure of a sentence based on the **dependencies between the words**



Questions?

Popular NLP Tools

- NLTK
- Stanford coreNLP
- **spaCy**

spaCy

spaCy

- Offer a comprehensive NLP pipeline with a simple/consistent interface

```
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

Hands-on session: Processing Text with spaCy

- Google Colab

Pandas 101

My Rule: CSV-In CSV-Out

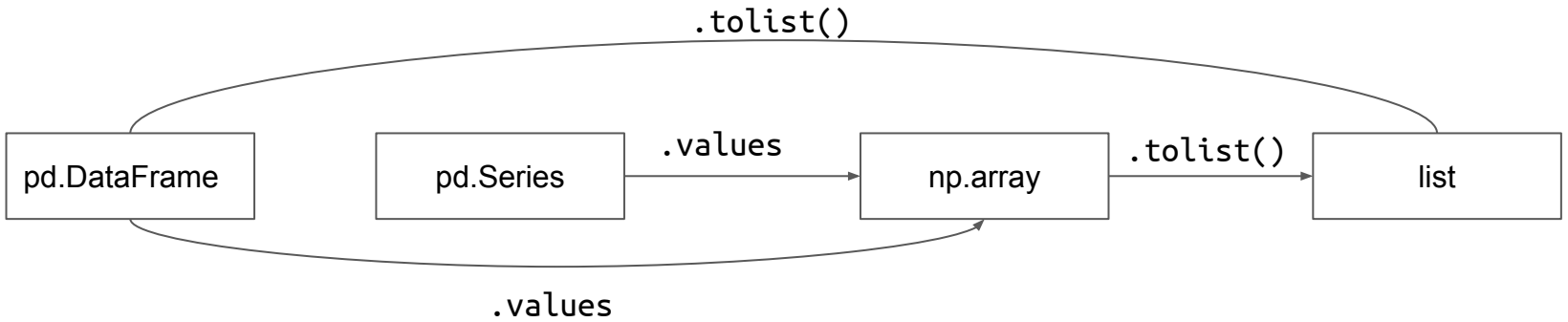
```
import pandas as pd

df = pd.read_csv(input_filepath)
…
df.to_csv(output_filepath)
```

Pandas offers APIs to load/save DataFrame from/to a variety of formats!

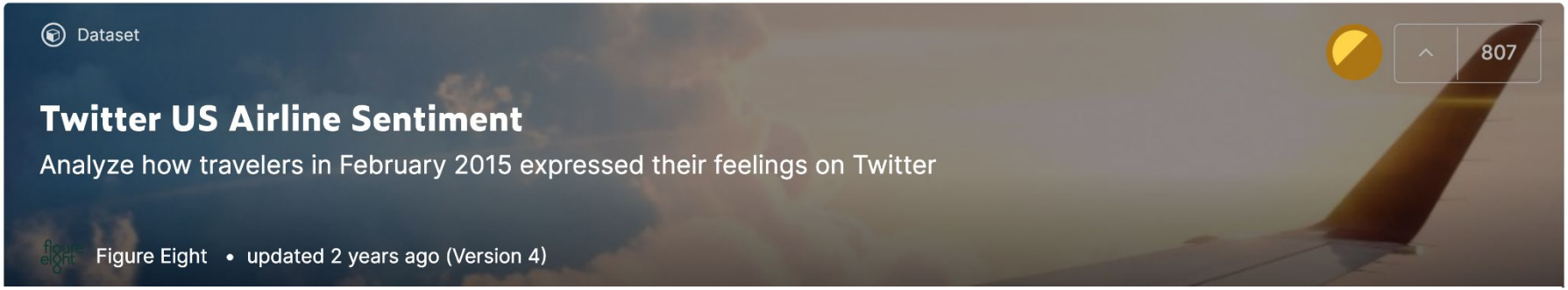
Pandas Objects

- pandas.DataFrame
- pandas.Series
- numpy.array
- list



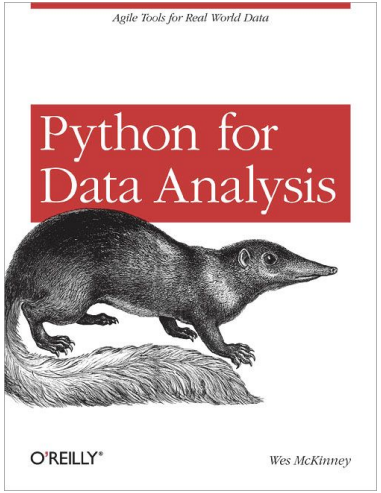
Hands-on session: Pandas 101

- Google Colab



Further Reading

- 10 Minutes to Pandas
  http://pandas.pydata.org/pandas-docs/stable/10min.html
- Python for Data Analysis, O'Reilly



Neural Networks for NLP

Neural Networks for NLP: Big Picture

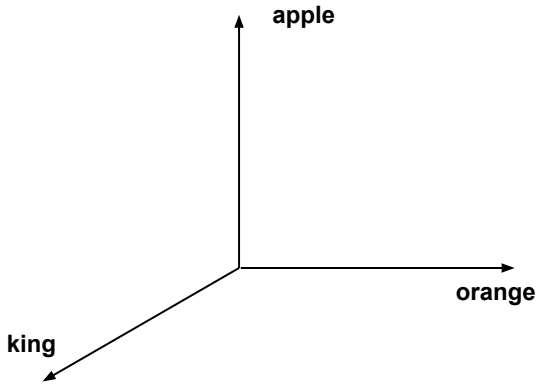
- 1) Input representations → Word embeddings [Today!]
- 2) Sequential input → Recurrent Neural Networks (RNNs) [Next week!]
- ...

What's the issue with Vector Space Model?

- The idea of vector representations already exist (since 1960's)

DOCUMENTS / TERMS	C0 403	F0 603	IS 411	J0 316	J0 413	J0 607	J0 609	J0 611	LY 612	WA 412
CDRUN				•	•	•				
CRECT							•			
LOKUP							•			
EDITG							•	•		
HADIC	•		•				•			
INFLE			•							•
RFVBS									•	
SORTG			•							
STRAN		•								
SUFAN				•						
SYNAR		•	•							
TRNSL		•							•	

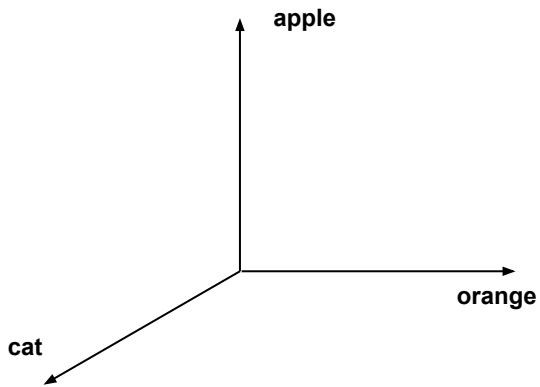
TERM-DOCUMENT MATRIX T FOR DOCUMENT SET

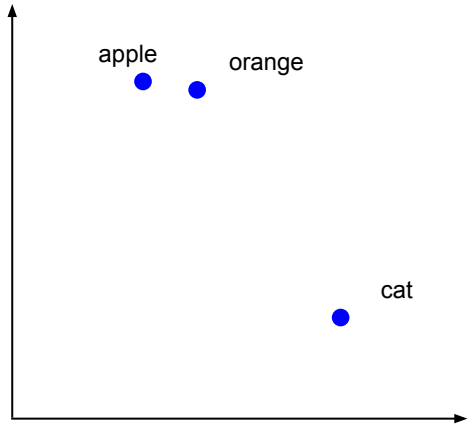


Each dimensions is **orthogonal**
Not appropriate to calculate the similarity between wods

Can We Represent Words as Dense Vectors?

- Each dimension has a certain type of semantic information

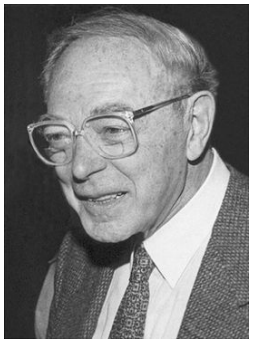




But, how?

Distributional Hypothesis (in Linguistics)

- Words that occur in the same contexts tend to have similar meanings [Harris 1954]
- The idea “a word is characterized by the company it keeps” was popularized by [Firth 1957]

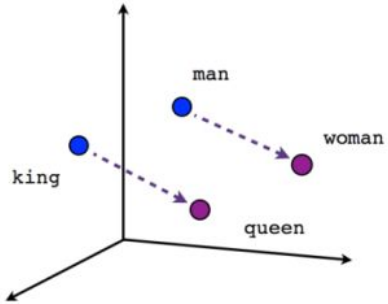


Zellig Harris

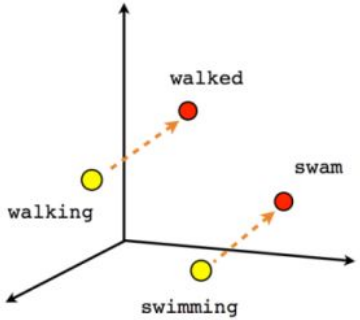
John Rupert Firth

Word2vec: Embedding Meanings into Dense Vectors

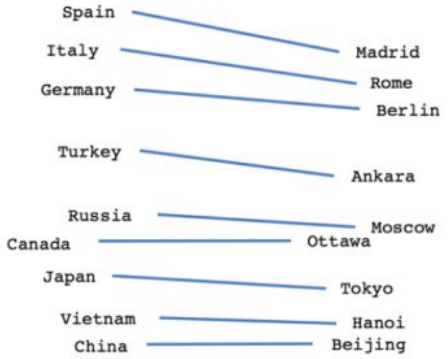
- Word2vec (2013) presents that we can **meaningfully** embed words into the vector space



Male-Female



Verb tense

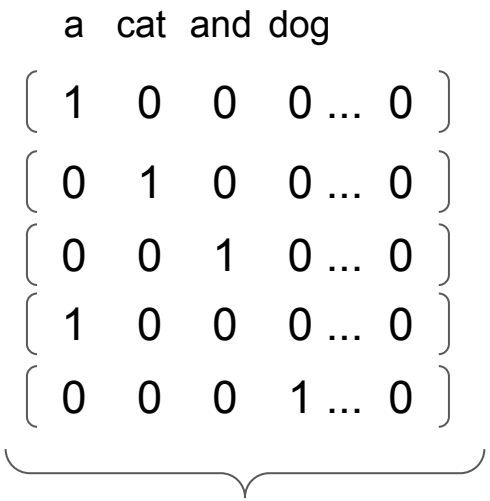


Country-Capital

e.g., v("king") - v("man") + v("woman") =~ v("queen")

Word Embeddings: *Dense* Vector Representations

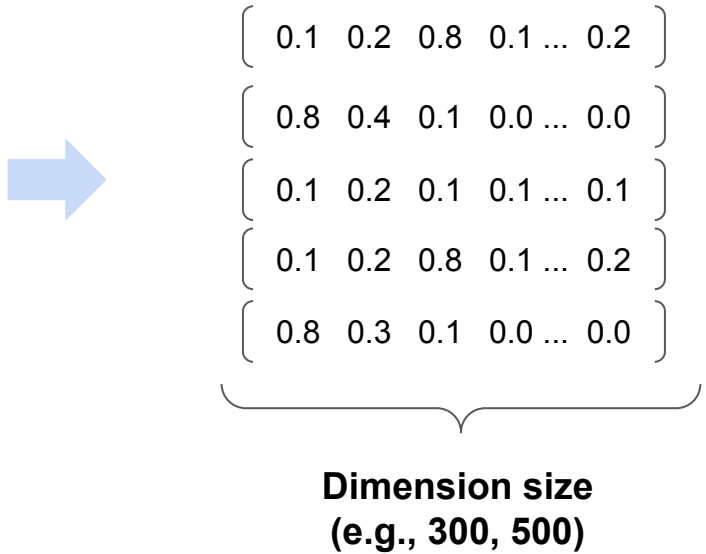
"A cat and a dog"



One-hot vector representations

Word Embeddings: *Dense* Vector Representations

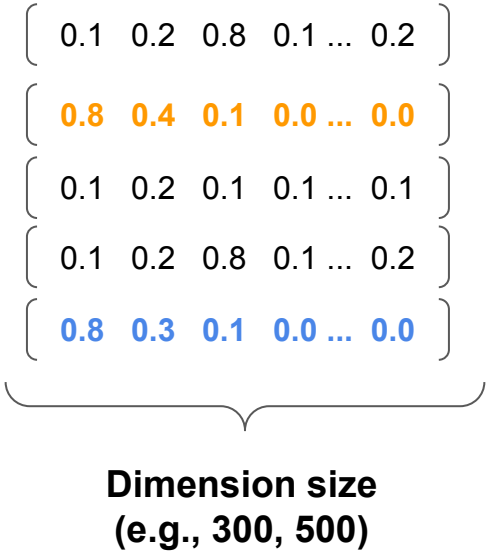
“A cat and a dog”



Dense vector representations

Word Embeddings: *Dense* Vector Representations

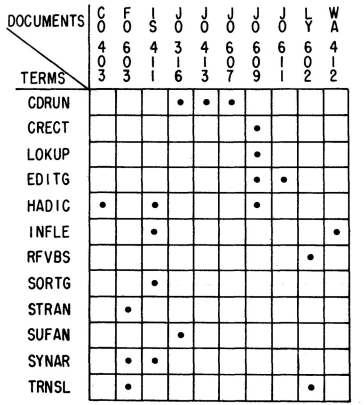
"A cat and a dog"



Good at capturing semantic similarity between words

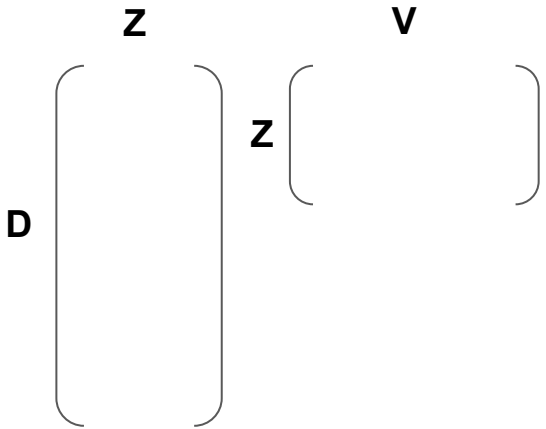
cf. Traditional Solutions

- Latent Semantic Indexing/Analysis (LSI/LSA)
  - Essentially, Singular Value Decomposition (SVD)
- Non-negative Matrix Factorization (NMF)
- Topic Models (e.g., Latent Dirichlet Allocation (LDA))



DOCUMENTS / TERMS	C0 403	F0 63	IS 411	J0 316	J0 413	J0 607	J0 609	J0 61	LY 602	WA 412
CDRUN				•	•	•				
CRECT							•			
LOKUP							•			
EDITG							•	•		
HADIC	•		•				•			
INFLE			•							•
RFVBS									•	
SORTG			•							
STRAN		•								
SUFAN				•						
SYNAR		•	•							
TRNSL		•							•	

TERM-DOCUMENT MATRIX I FOR DOCUMENT SET

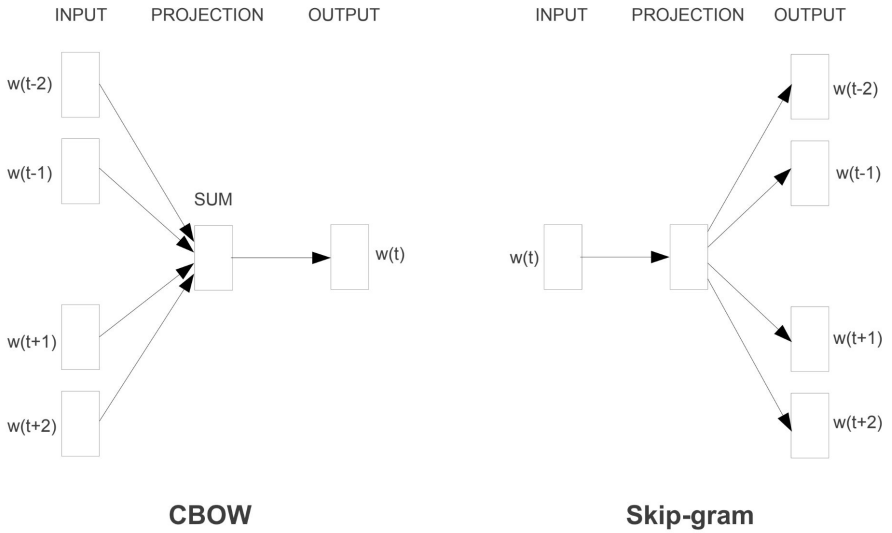


Word Embedding Models

- Word2vec
- GloVe
- fastText

Word2vec [Mikolov et al. 2013]

- Two options for training (often misunderstood)
  - CBoW
  - Skip-gram



Context Window

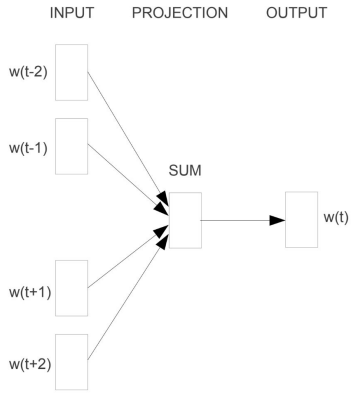
- Trying to represent a target word by context words

Deep Learning is fun

Deep Learning is fun

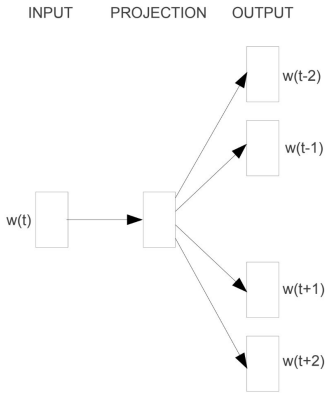
Deep Learning is fun

Deep Learning is fun



CBOW

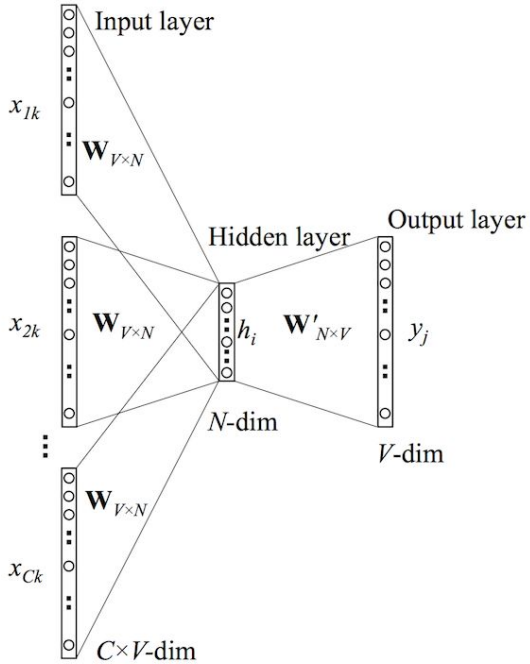
(Context words → target word)



Skip-gram

(Target word → context words)

Continuous Bag-Of-Words Model (CBOW)




```python
class CBOW(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear = nn.Linear(embedding_dim, vocab_size)

    def forward(self, X):
        emb = torch.sum(self.embeddings(X), dim=1)
        out = self.linear(emb) # nonlinear + projection
        log_probs = F.log_softmax(out, dim=1) # softmax compute log probability

        return log_probs
```

[New!] nn.Embedding

```
nn.Embedding(vocab_size, dim_size)
```

EMBEDDING

CLASS `torch.nn.Embedding(num_embeddings, embedding_dim, padding_idx=None, max_norm=None, norm_type=2.0, scale_grad_by_freq=False, sparse=False, _weight=None, device=None, dtype=None)` [SOURCE]

A simple lookup table that stores embeddings of a fixed dictionary and size.

This module is often used to store word embeddings and retrieve them using indices. The input to the module is a list of indices, and the output is the corresponding word embeddings.

Wait. Can't we use nn.Linear?

PyTorch Implementation

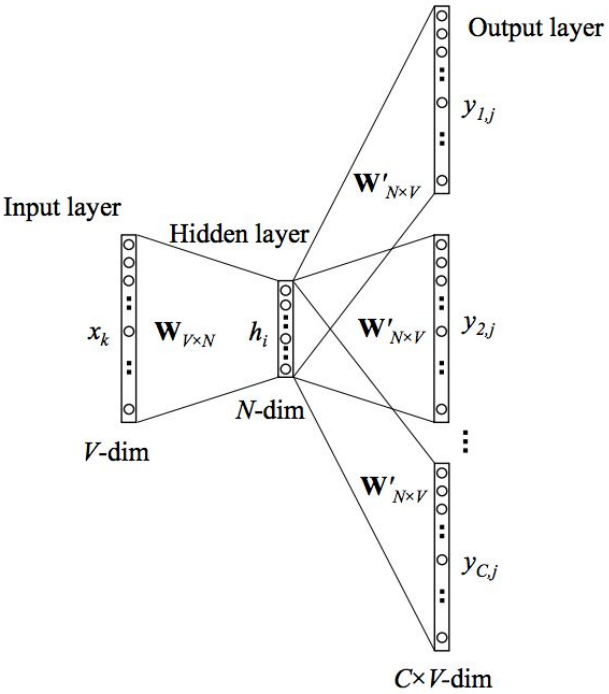
https://srijithr.gitlab.io/post/word2vec/

```
class CBOW(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear = nn.Linear(embedding_dim, vocab_size)

    def forward(self, X):
        emb = torch.sum(self.embeddings(X), dim=1)
        out = self.linear(emb) # nonlinear + projection
        log_probs = F.log_softmax(out, dim=1) # softmax compute log probability

        return log_probs
```

Skip-gram




```python
class SkipGram(nn.Module):
    def __init__(self, vocab_size, embedding_dim, context_size):
        super().__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear1 = nn.Linear(embedding_dim, 128)
        self.linear2 = nn.Linear(128, context_size * vocab_size)
        self.context_size = context_size

    def forward(self, X):
        embeds = self.embeddings(X).view((1, -1))
        out1 = F.relu(self.linear1(embeds))
        out2 = self.linear2(out1)
        log_probs = F.log_softmax(out2, dim=1).view(self.context_size, -1)
        return log_probs
```

Limitations of word2vec

- Limited window size: Either CBOW/Skip-gram does NOT consider context beyond the window size

GloVe [Pennington et al. 2014]

- Word-word co-occurrence matrix

I like deep learning.
I like NLP.
I enjoy flying.

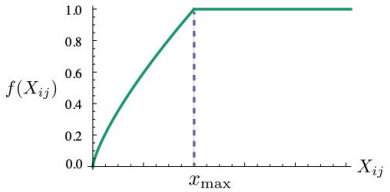
counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

GloVe: Modeling Co-occurrence by 2x Embedding Vectors

- Co-occurrence probability: $P_{ij} = X_{ij} / X_i$
- Word embedding vectors: u_i and v_j

$$J(\theta) = \frac{1}{2} \sum_{i,\, j=1}^{W} f\left(P_{ij}\right)\left(u_i{}^T v_j - \log P_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$



GloVe: Summary

- Final vector: $z_i = u_i + v_i$

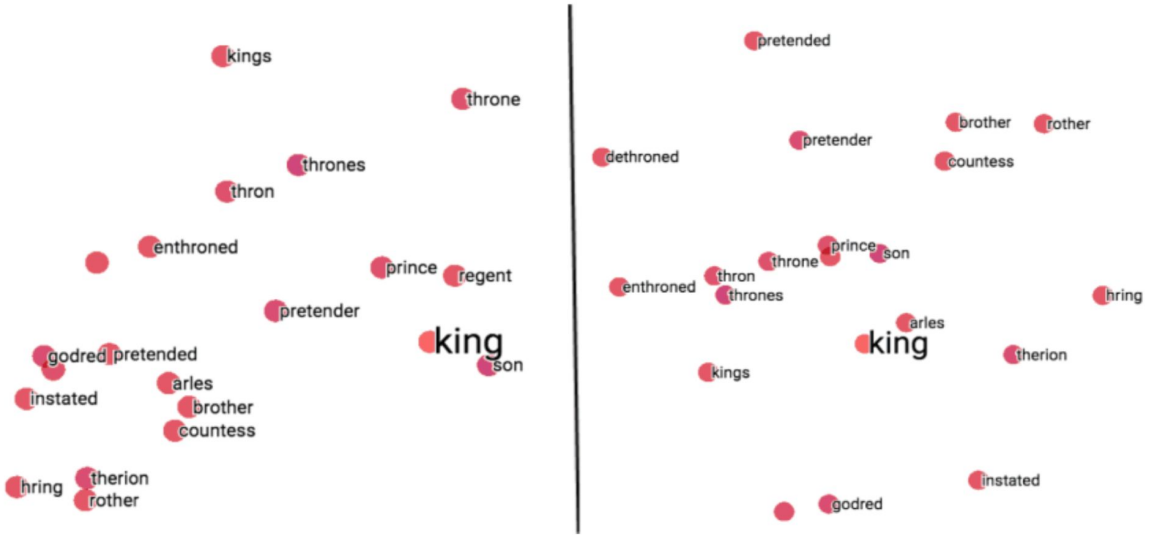
- Strengths
  - Fast training
  - Scalable to huge corpora
  - Good performance even with small corpus, and small vectors

Limitations of Word2vec/GloVe

- Both of them still suffer Out-of-Vocabulary issues
- Words with even little differences are assigned different token IDs
  - e.g., apple vs. apples, beautiful vs. beautifully, biceps vs. triceps

fastText: Subword Embeddings

- N-gram embeddings instead of “word” embeddings
  - eating → (<s>ea, eat, ati, tin, ing, ng</s>)
- CBoW/Skip-gram can be used for training



Is fixed n-gram the best? → Learned sub-tokenization (Week 8)

spaCy for word embeddings

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
    print([token.text, token.lemma_, token.pos_, token.dep_, token.ent_iob_, token.ent_type_])
```

```
token.vector
```

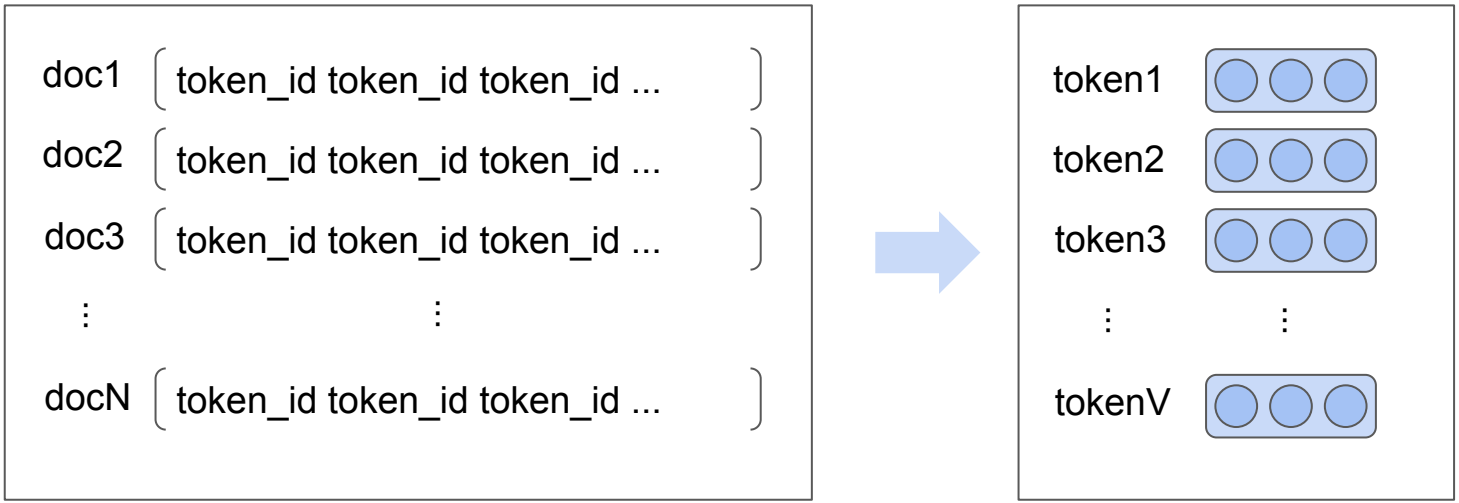
```
array([-1.535186  , -0.4060119 ,  2.9251206 ,  1.4862046 , -0.5711231 ,
        0.04101598,  2.2709746 , -2.7806776 , -0.5188239 , -1.9200461 ,
        0.6264886 ,  5.113931  , -4.4795375 , -3.3172297 , -2.591983  ,
        0.2952245 ,  3.0970662 , -2.1948025 , -1.0125057 , -1.9231814 ,
        3.3866777 , -1.1537448 , -1.3051802 ,  0.10659629, -2.4653244 ,
        4.1557007 , -4.0343957 ,  0.96222323, -0.5591469 , -2.192143  ,
       -0.194722  , -1.6126934 ,  1.8915527 ,  0.05074558, -1.1258242 ,
       -1.311002  , -0.22932756,  0.86395764,  2.1479762 , -0.04867677,
        1.0269833 , -1.6562294 , -3.2395341 , -0.8505454 , -0.7531296 ,
       -0.29259783, -0.24896976, -3.0125833 ,  4.453686  ,  2.874859  ,
        4.92746   , -3.9719796 , -1.8743998 , -0.7232732 ,  0.68904847,
        1.0461929 ,  1.0565916 ,  2.1449113 ,  2.593386  , -0.22828588,
        4.1223226 ,  0.56345993,  3.7295504 , -2.460603  ,  3.2813878 ,
       -3.3111894 ,  2.4160173 , -1.5893238 , -2.481103  ,  2.9629679 ,
        5.0079627 , -3.640224  ,  4.620039  , -0.13005406,  1.9256897 ,
       -1.8064855 ,  0.5464196 , -0.21171951, -1.3014529 , -4.0162244 ,
        0.98979425,  0.5273107 , -2.2770624 ,  2.0088112 ,  2.5486505 ,
        1.4092522 ,  2.6082263 , -1.2317483 , -2.1186576 , -2.004669  ,
       -0.01678723,  0.21361685,  0.64057314, -3.5370805 ,  0.56346744,
        2.3472533 ], dtype=float32)
```

```
doc.vector
```

```
array([ 1.0188444 ,  0.03571144,  0.7764361 ,  1.2014427 ,  0.1412878 ,
        0.1682428 ,  1.1233734 ,  0.3487686 ,  0.6835691 ,  1.4795598 ,
        0.20042907,  0.9256677 , -1.5593272 , -0.6143042 , -1.557381  ,
       -0.9958308 , -0.5963076 , -1.6042081 , -0.39841834, -0.3468938 ,
        1.0979723 , -0.07698824,  0.5335008 ,  0.72681975, -1.0884286 ,
        0.7676121 , -1.7708863 ,  0.72250545,  0.20121218, -1.444205  ,
        0.9010805 , -0.15313278,  0.2421783 , -1.3908852 , -0.69058996,
       -0.9723627 ,  0.7619291 , -0.49309894, -1.1831714 , -0.02785109,
        0.76898694, -0.26471007, -0.5232663 , -1.2854791 , -0.9188446 ,
        0.2745516 ,  1.0659344 ,  0.31896058, -0.04838419,  0.8291477 ,
        0.8577619 , -1.0837296 , -0.01163011,  0.04729161, -1.3421645 ,
        1.395626  , -0.2455926 ,  0.6314798 ,  2.260128  , -0.14274798,
        1.7947469 , -0.4407996 ,  1.5280372 , -0.25356895,  1.2945919 ,
       -0.15937866,  0.67561346, -2.3464286 , -0.5416016 ,  1.6168556 ,
        1.8714851 , -1.9004959 ,  1.2033213 , -1.0751767 ,  1.1551958 ,
       -0.16097708,  1.0588483 , -0.35734576, -1.1385773 , -0.8543524 ,
        0.11442439, -1.3668202 , -0.05677014, -0.03700415,  2.2501202 ,
        1.0935678 , -0.8177019 ,  0.7018638 , -0.852125  , -0.05367121,
       -1.1944368 , -1.9318706 ,  0.39264107, -0.14523607,  0.8427848 ,
        1.0945965 ], dtype=float32)
```

Brainstorming: X Embeddings (X2vec)

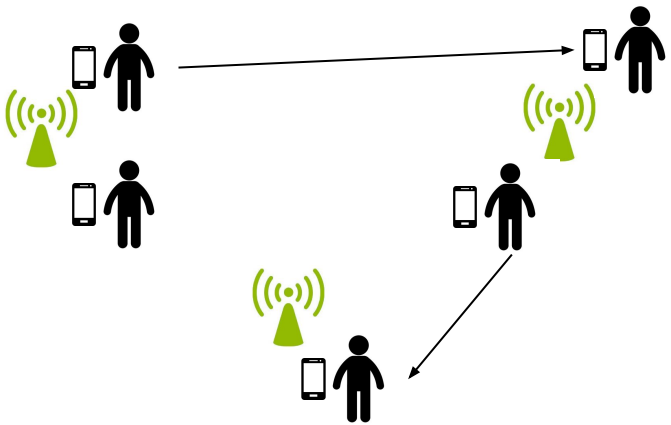
- doc → ?
- token_id → ?



Any ideas?

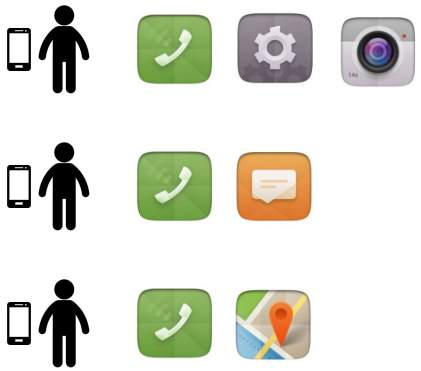
Non-NLP Example 1: Grouping Users by Mobility Patterns

- Document: User
- Word: WiFi spot



Non-NLP Example 2:
Grouping Users by Phone App Usage

- Document: User
- Word: Application



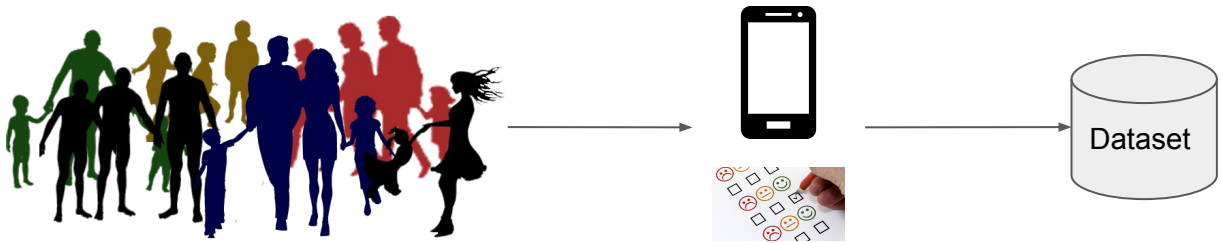
Example: Friends and Family Dataset

http://realitycommons.media.mit.edu/friendsdataset.html

Android OS based mobile phones are used as in-situ social sensors to map users activity features, proximity networks, media consumption, and behavior diffusion patterns.

Subjects complete surveys at regular intervals, combining web-based and on-phone surveys. Monthly surveys include questions about self-perception of relationships, group affiliation, interactions, and also standard scales like the Big-Five personality test.

Daily surveys include questions like mood, sleep, and other activity logging. Information on purchases is collected through receipts and credit card statements submitted at the participants' discretion



Friends and Family description

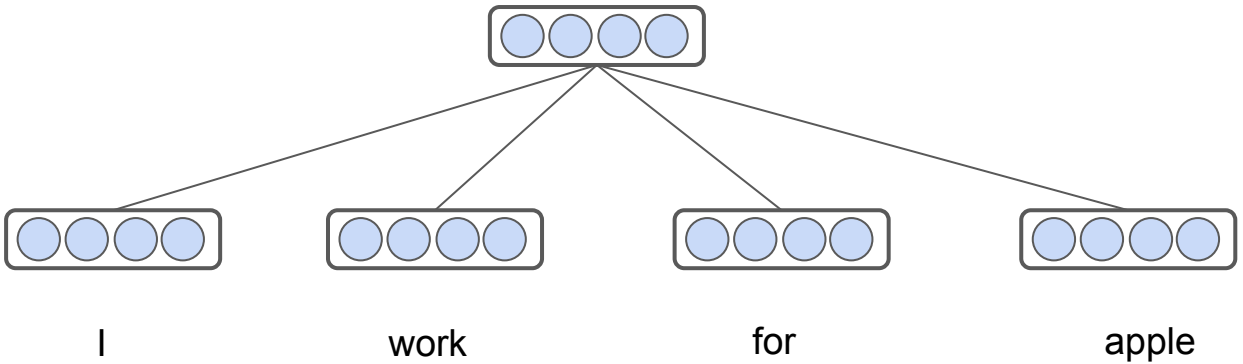
http://realitycommons.media.mit.edu/friendsdataset2.html

- Sensor data (every 6 minutes)
  - Geographical location and proximity with other members
  - Call/SMS log information
  - Activity (Walking, Jogging, In a vehicle etc.)
- Individual demographics survey
  - Income, lifestyle
  - Big-5 personality test
- Weekly survey
  - Times going to restaurant, ordering food, and cooking; dining with whom - alone, with friends, with family etc.
  - Watching movie, TV frequency and with whom and where
  - Expected and enjoyed events, with whom
  - Amount of time spend with kids
  - Involvement with activity groups

Beyond Word Embeddings

Sentence/Document Embeddings

- The simple average of word embeddings is a common choice



Published as a conference paper at ICLR 2017

Smooth Inverse Frequency (SIF)

- Simple Average+

A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS

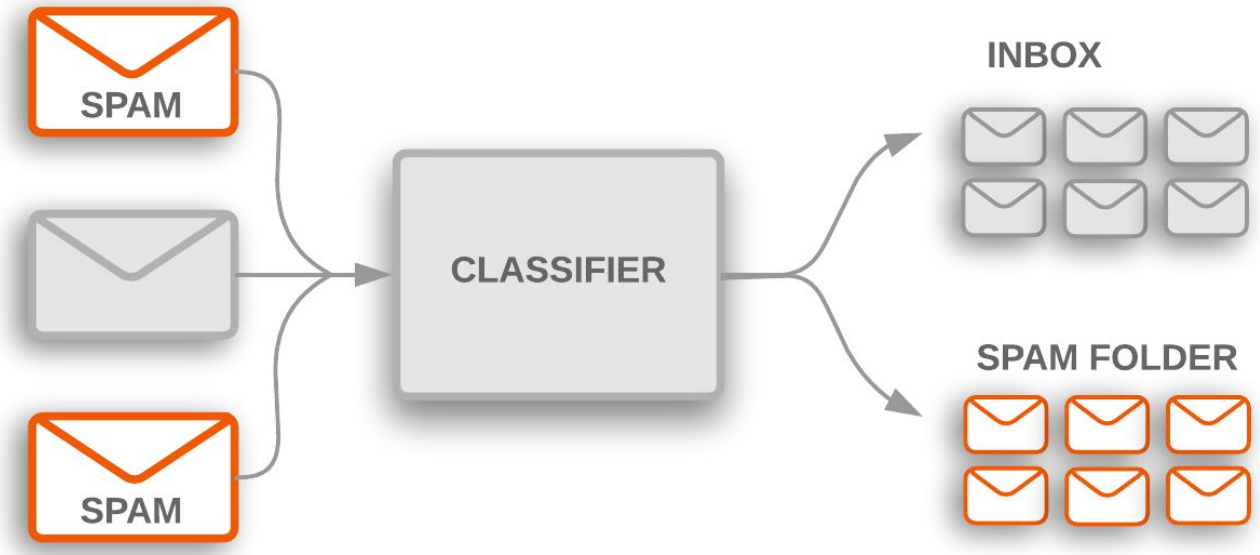
Sanjeev Arora, Yingyu Liang, Tengyu Ma
Princeton University
{arora,yingyul,tengyu}@cs.princeton.edu

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**
2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$ "IDF-like" weighting method
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $v_s \leftarrow v_s - uu^\top v_s$ "common component removal"
7: **end for**

Text Classification: Spam Detection



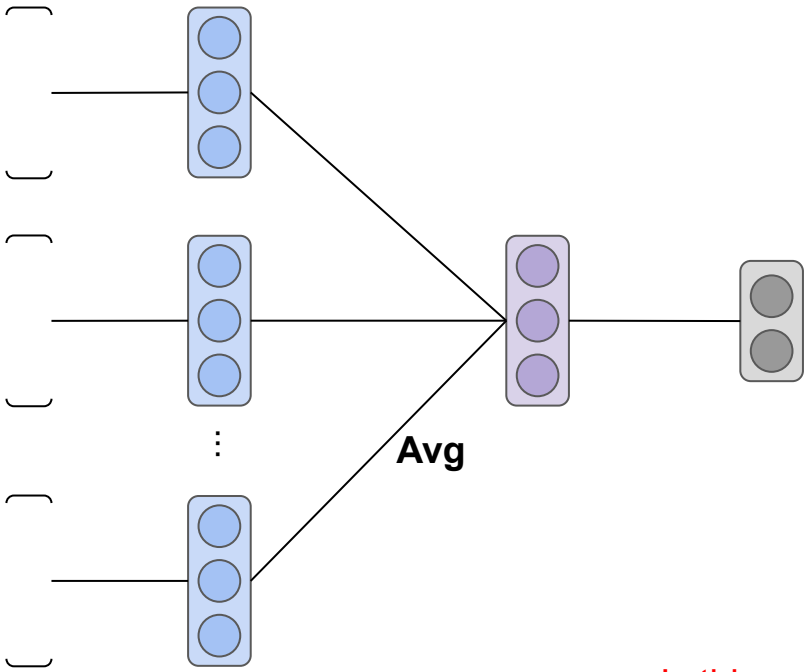
Text Classification: Sentiment Analysis

Any ideas?

Text Classification: Traditional Solution

- BoW representations + classifier
- We have word embeddings. Now what?

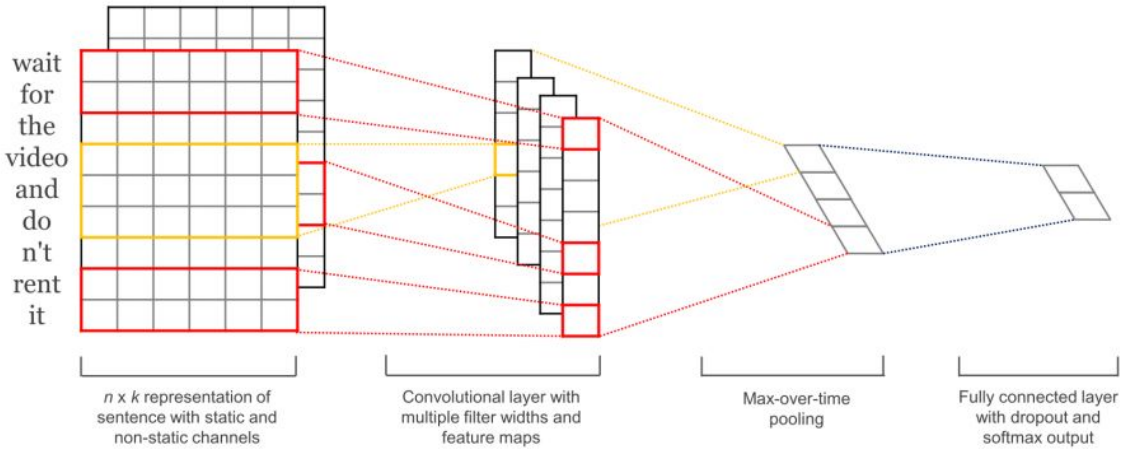
Word Embeddings for Text Classification
Sentence/Document Embeddings + Classifier



Is this model linear or non-linear?

Convolutional Neural Networks for Text

- A solution to incorporate the **sequence order** into the model
  - We will discuss a more intuitive way (RNN) next week!



https://arxiv.org/abs/1408.5882

Summary

- NLP Basics
  - Hands-on session: NLP Pipeline with spaCy
  - Hands-on session: Pandas 101
- Word embeddings
  - Word2vec, GloVe, fastText
- Beyond word embeddings

Key takeaway: NLP techniques can be used for non-NLP problems!