CMPT 733 – Big Data Programming II

Deep Learning II

Instructor Steven Bergner

Course website https://coursys.sfu.ca/2024sp-cmpt-733-g1/pages/

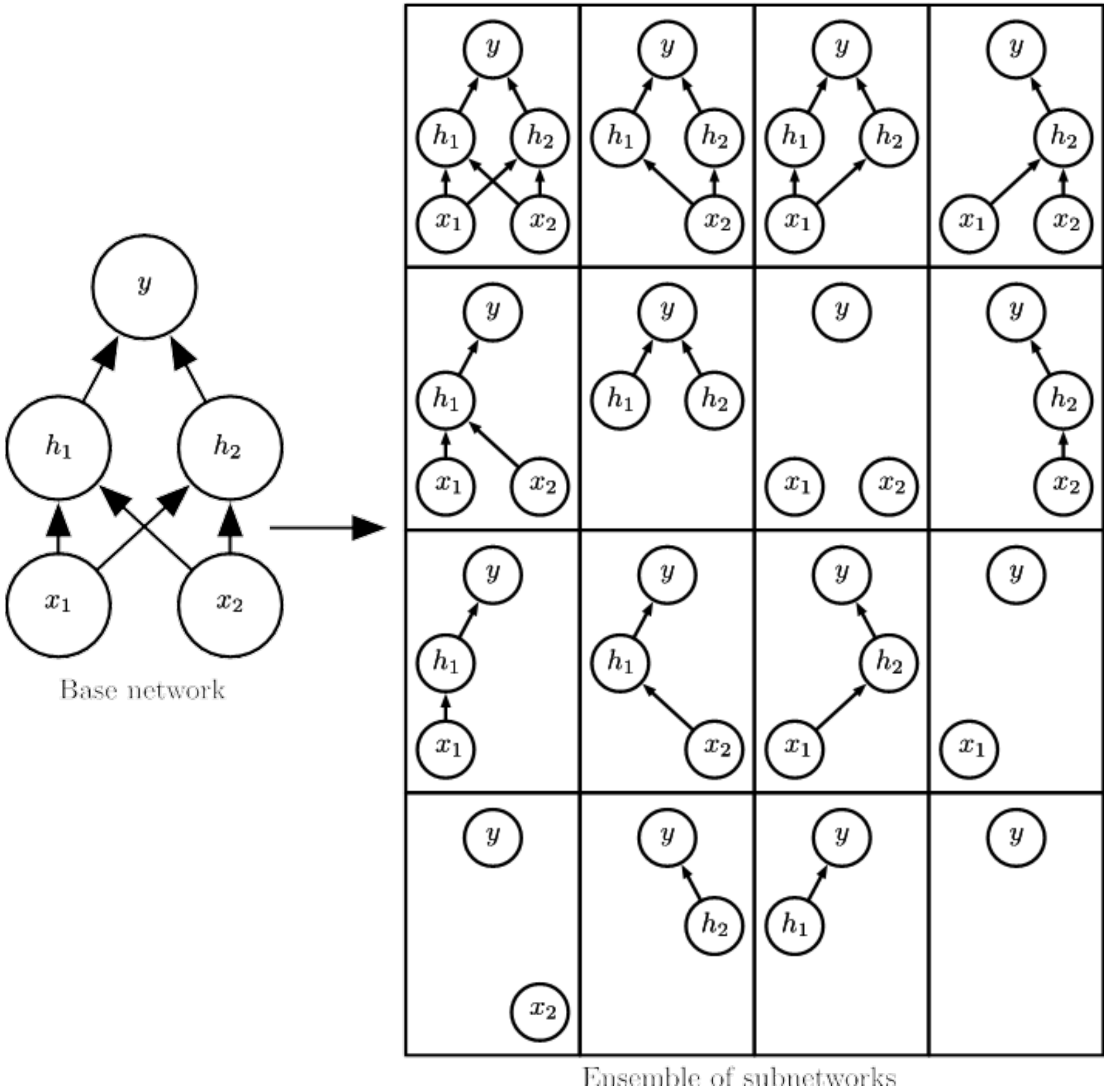
Overview

- Recap: Overfitting remedies
- Deep learning for sequences
- Natural language processing, e.g.
  - Sentiment analysis
  - Word embeddings
- Visualization for Deep Learning

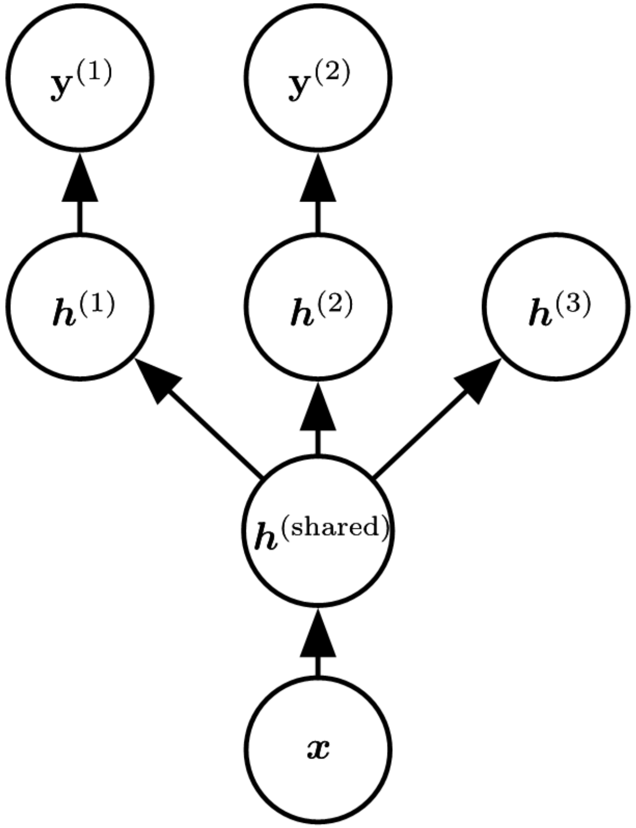
Strategies against Overfitting
(short recap)

Dropout

- Random sample of connection weights is set to zero
- Train different network model each time
- Learn more robust, generalizable features



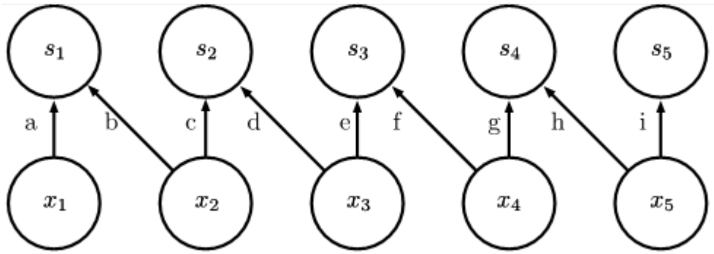
Multitask learning



- Shared parameters are trained with more data
- Improved generalization error due to increased statistical strength
- Missing components of y are masked from the loss function

[Goodfellow, Bengio, Courville 2016]

Types of connectivity



Local connection:
like convolution,
but no sharing

$$\begin{bmatrix} a & b & & & \\ & c & d & & \\ & & e & f & \\ & & & & \ddots \end{bmatrix}$$

$$\begin{bmatrix} a & b & & & \\ & a & b & & \\ & & a & b & \ddots \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d & \cdots \\ h & i & j & k & \cdots \\ o & p & q & r & \cdots \\ & & & & \ddots \end{bmatrix}$$

[Goodfellow, Bengio, Courville 2016]

Convolution calculation illustrated

[1 1 2 4 5 5 5 3 0 0]

* [-1 1]

Choosing architecture family

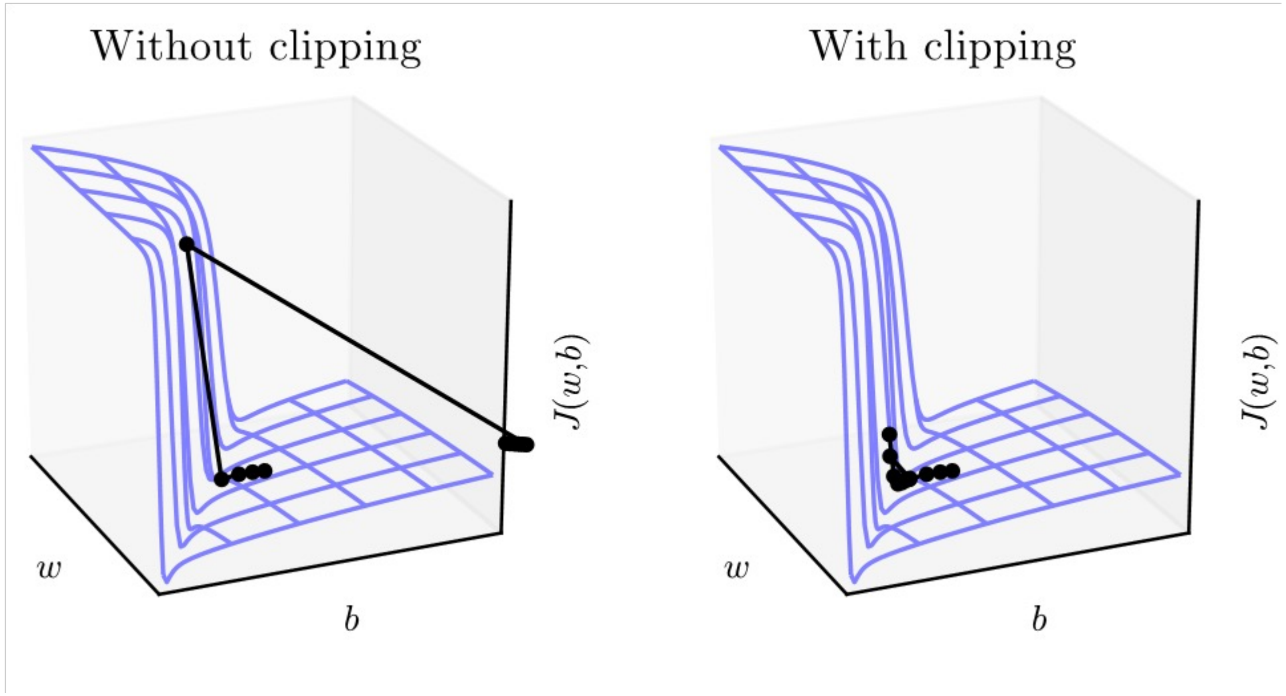
- No structure → fully connected
- Spatial structure → convolutional
- Sequential structure → recurrent

Optimization Algorithm

- Lots of variants address choice of learning rate
- See Visualization of Algorithms
- AdaDelta and RMSprop often work well

Gradient Clipping

- Add learning rate time gradient to update parameters
- Believe direction of gradient, but not its magnitude



Development strategy

- Identify needs: High accuracy or low accuracy?
- Choose metric
  - Accuracy (% of examples correct), Coverage (% examples processed)
  - Precision TP/(TP+FP), Recall TP/(TP+FN)
  - Amount of error in case of regression
- Build end-to-end system
  - Start from baseline, e.g. initialize with pre-trained network
- Refine driven by data

Software for Deep Learning

Current Frameworks

- Tensorflow / Keras
- PyTorch
- DL4J
- Caffe (superseded by Caffe2, which is merged into PyTorch)
- And many more
- Most have CPU-only mode but much faster on NVIDIA GPU

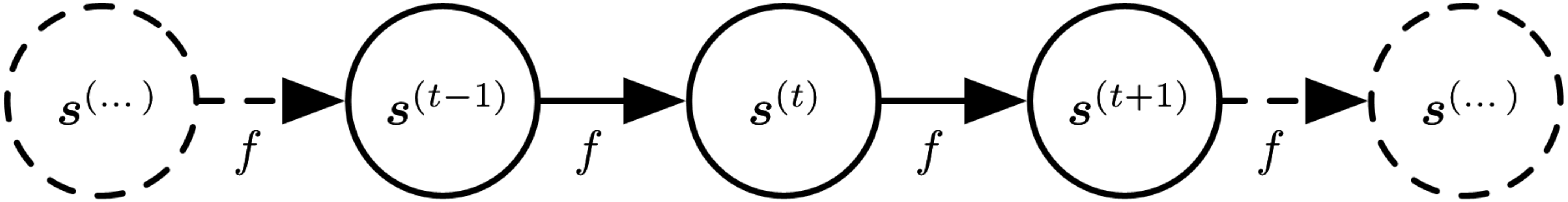
Recap: Choosing architecture family

- No structure → fully connected
- Spatial structure → convolutional
  - Adjacency or order of inputs has meaning
- Sequential structure → recurrent

Sequence Modeling with Recurrent Nets

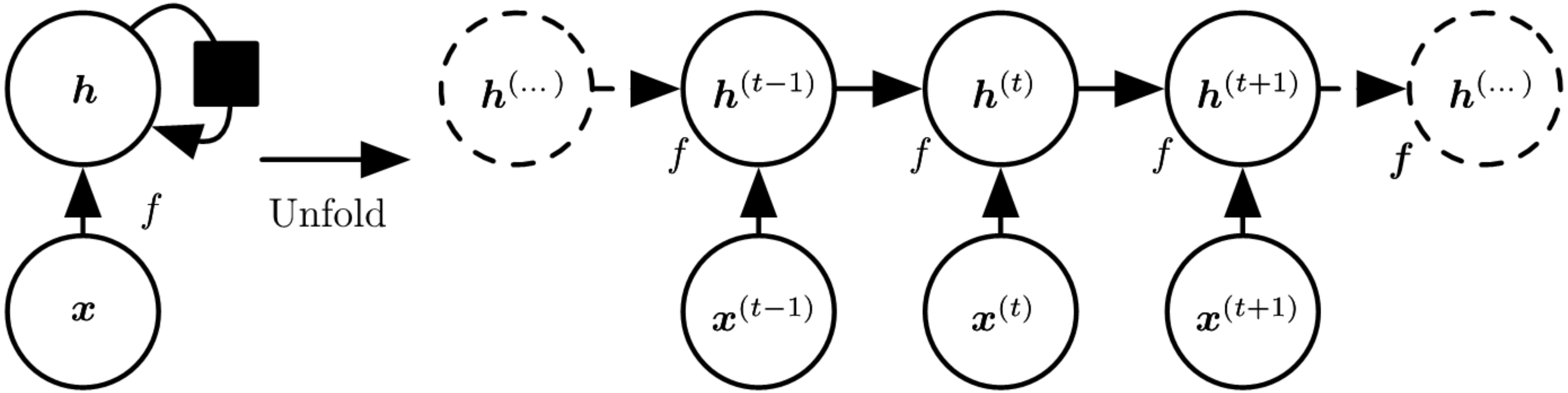
Classical Dynamical Systems

- Recurrent network models a dynamical system that is updated in discrete steps over time
- Function *f* takes input from time *t* to output at time *t+1*
- Rules persist across time



Unfolding Computation Graphs

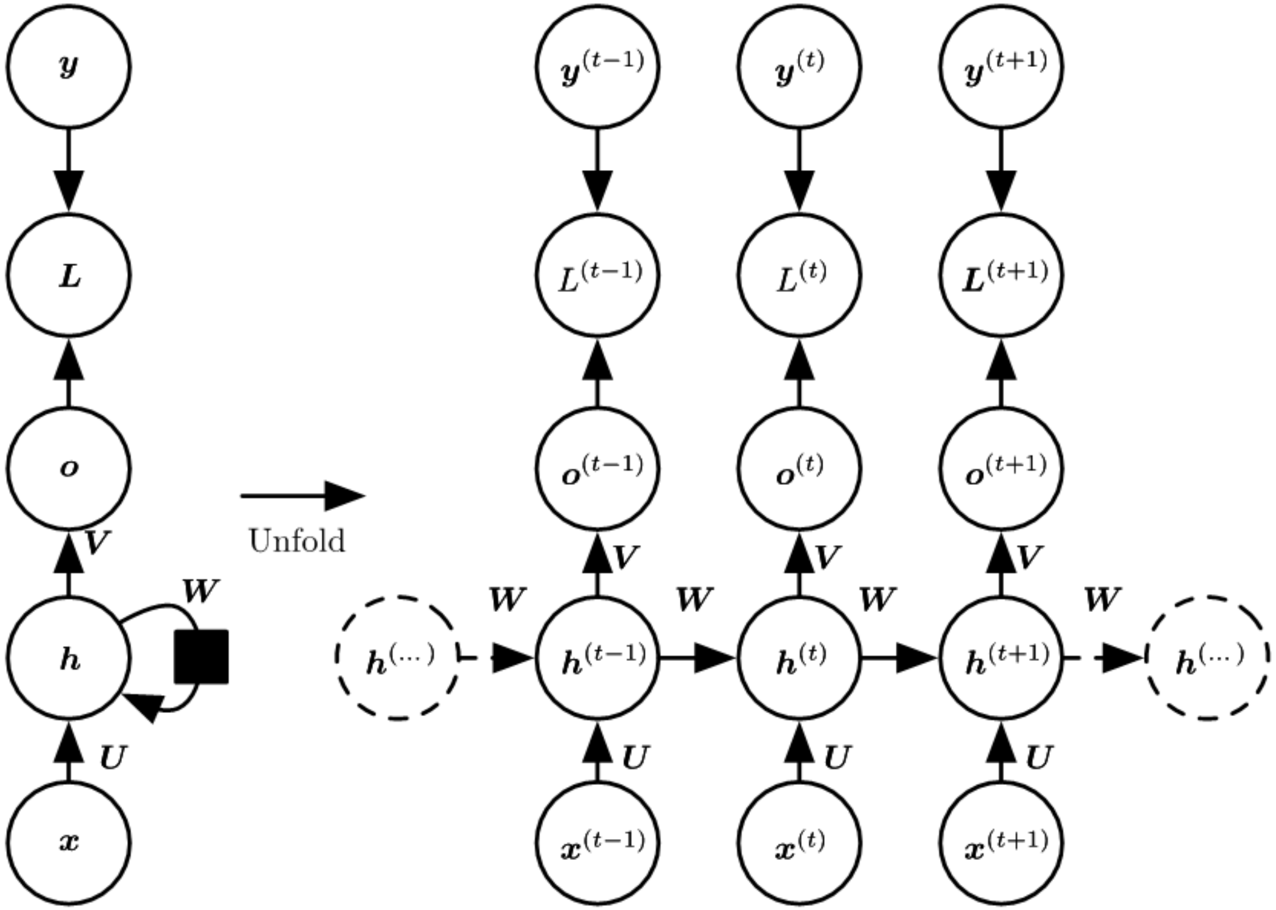
- Recurrent graph can be unfolded, where hidden state h is influencing itself
- Backprop through time is just backprop on unfolded graph



[Goodfellow, Bengio, Courville 2016]

Recurrent Hidden Units

- Can have more than one layer

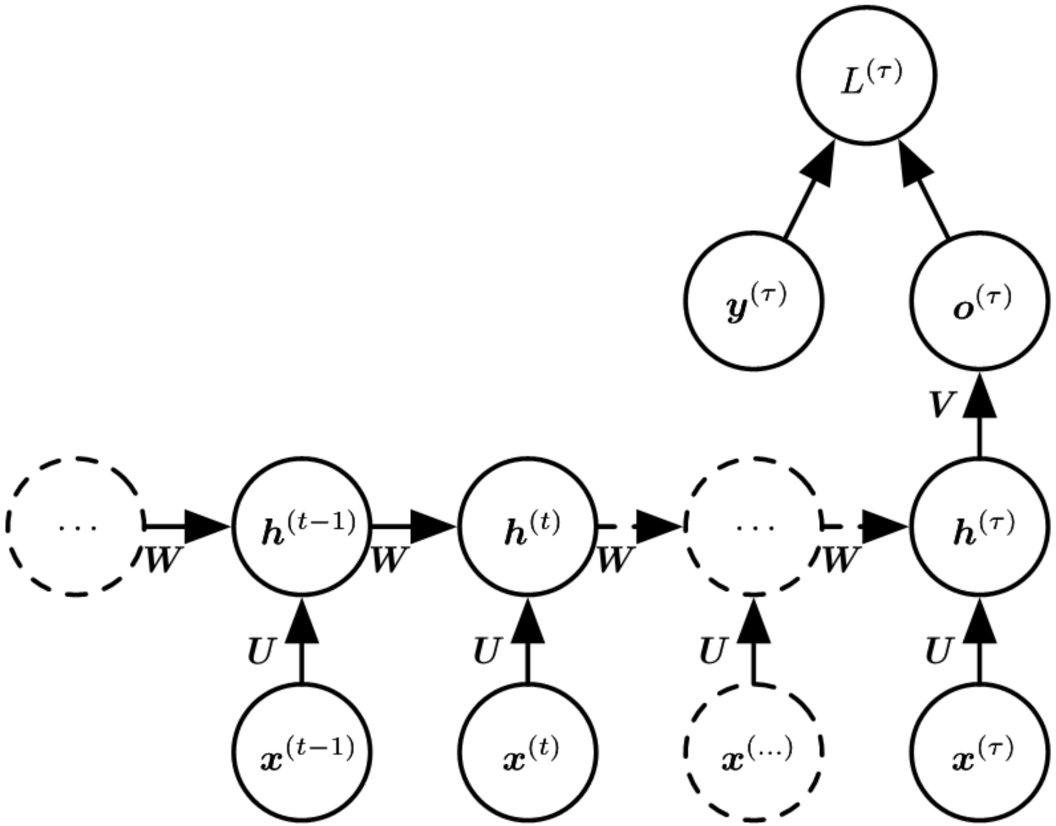


[Goodfellow, Bengio, Courville 2016]

Sequence Input, Single Output

Example

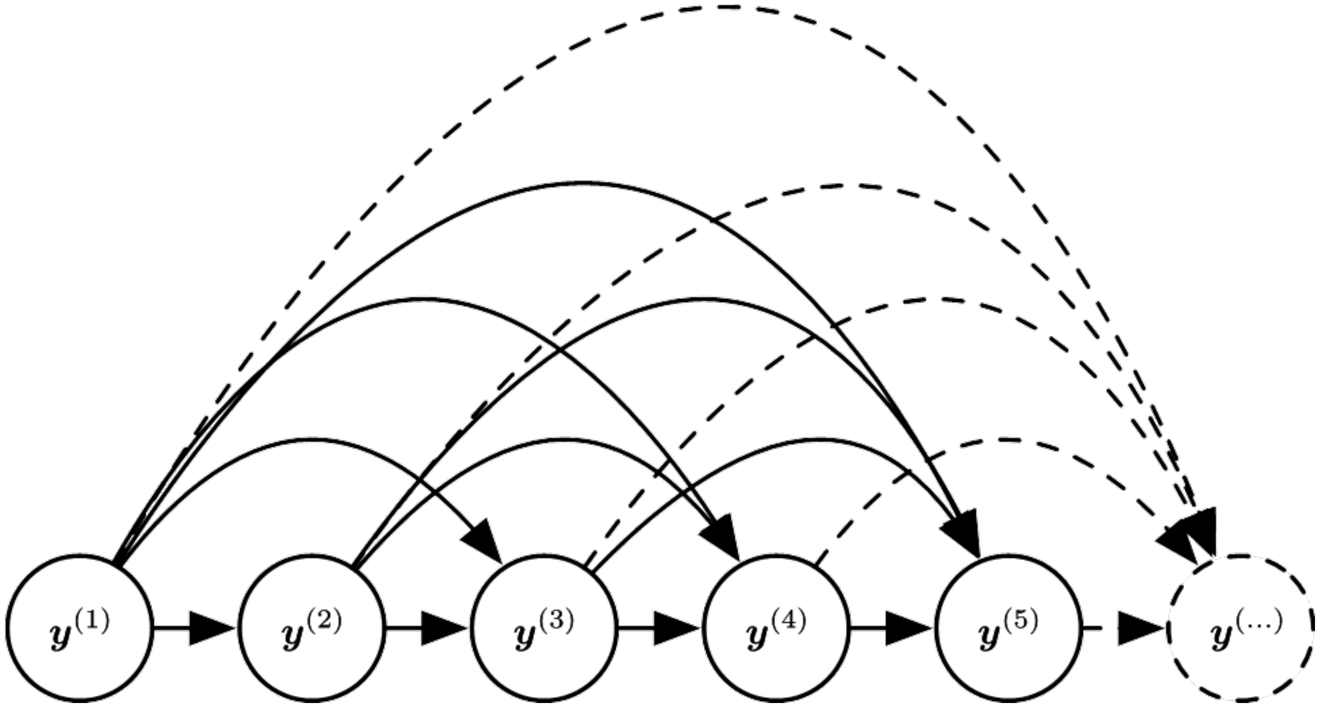
Sentiment analysis of text



[Goodfellow, Bengio, Courville 2016]

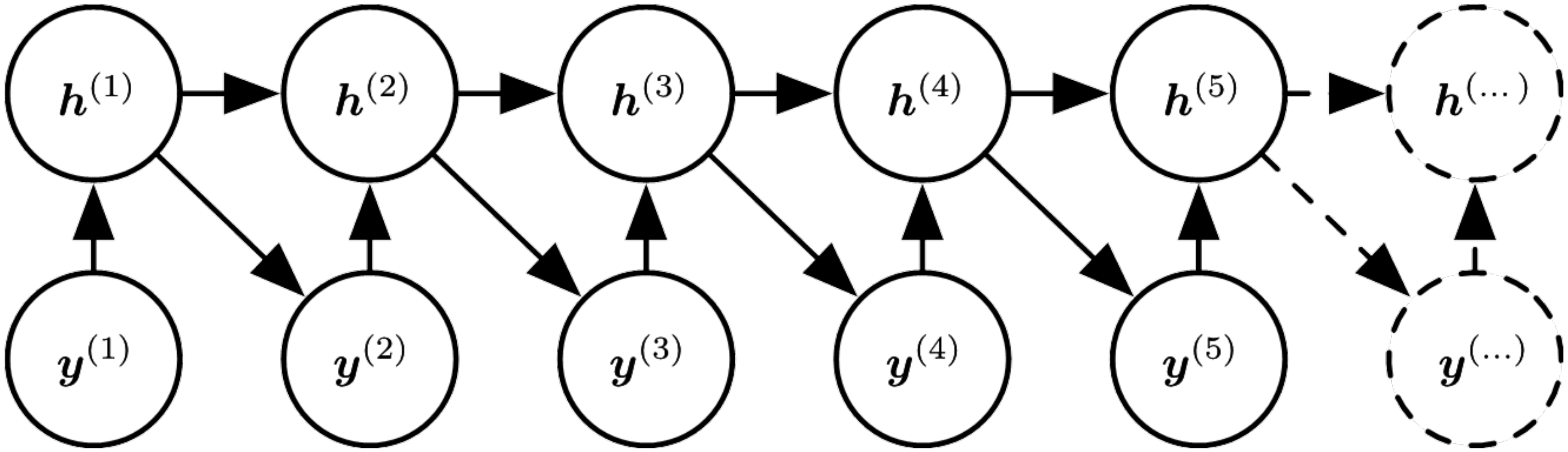
Fully Connected Graphical Model

- Too many dependencies among variables, if each has its own set of parameters



RNN Graphical Model

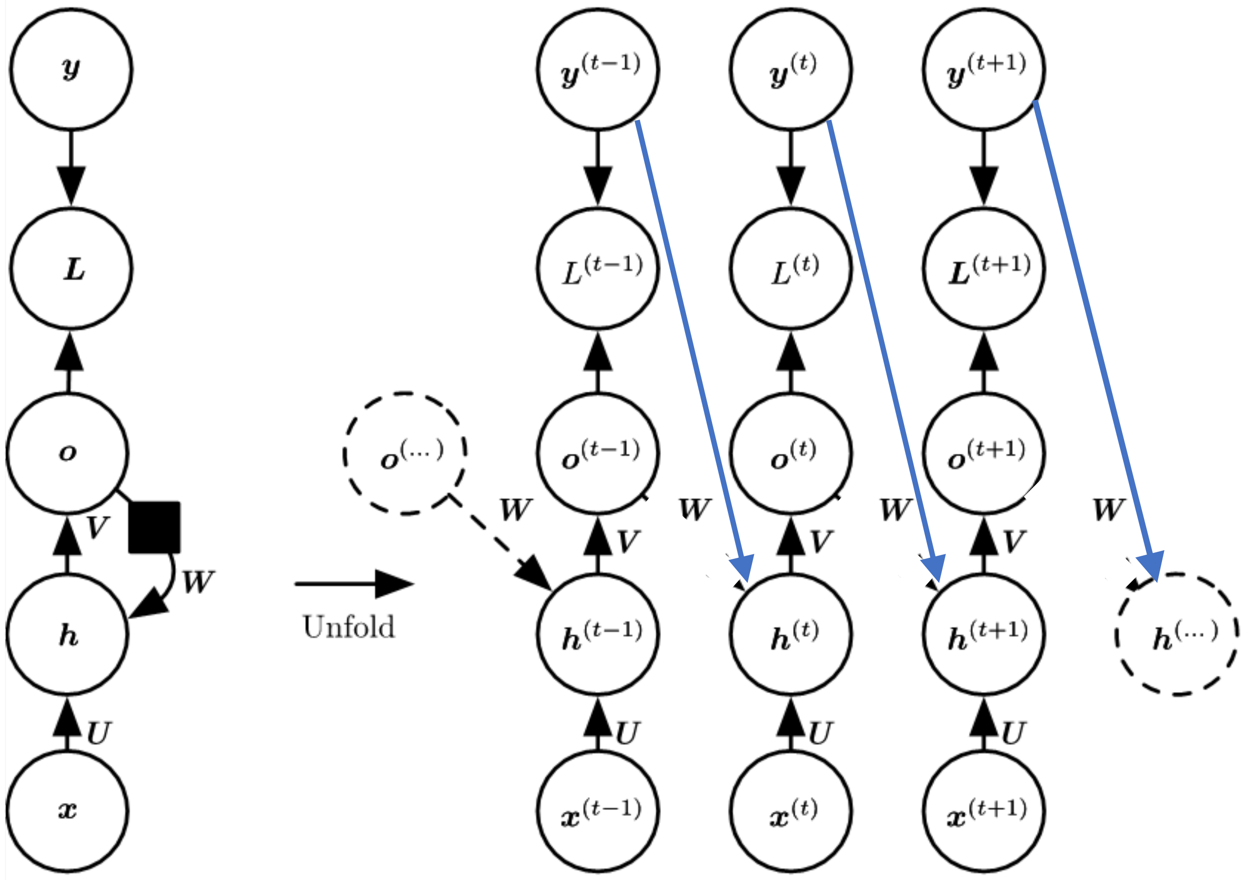
- Organize variables according to time with single update rule
- Finite set of relationships may extend to infinite sequences
- *h* acts as “memory state” summarizing relevant history



[Goodfellow, Bengio, Courville 2016]

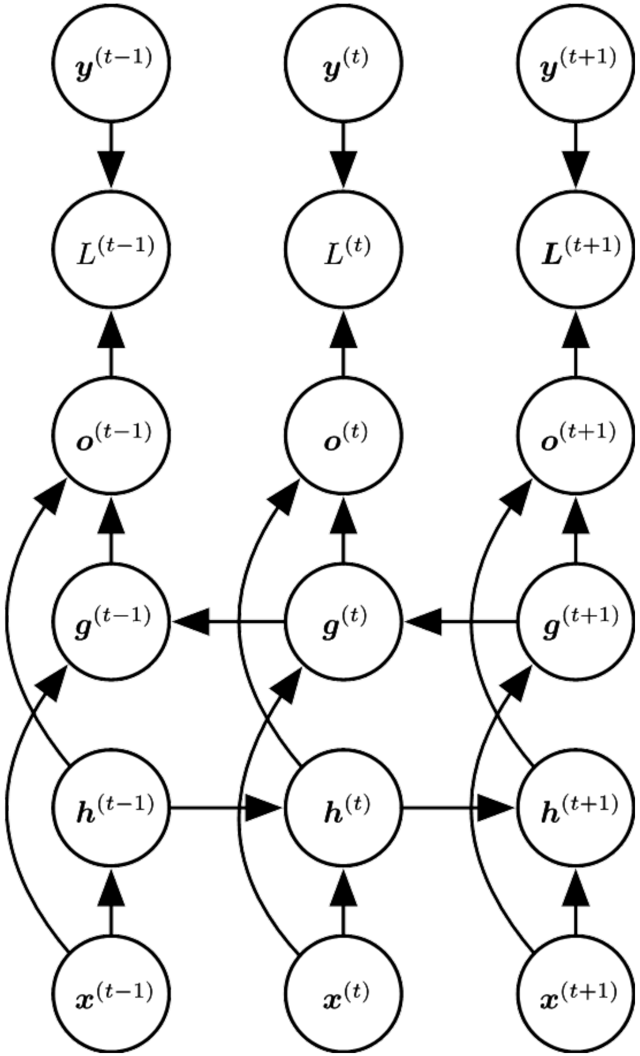
Recurrence only through output

- Avoid backprop through time
- Mitigation: Teacher forcing
  - Use actual or expected output from the training dataset at current time y(t) as input o(t) to the next time step, rather than generated output
  - Backprop stops when it reaches y(t-1) via o(t-1)



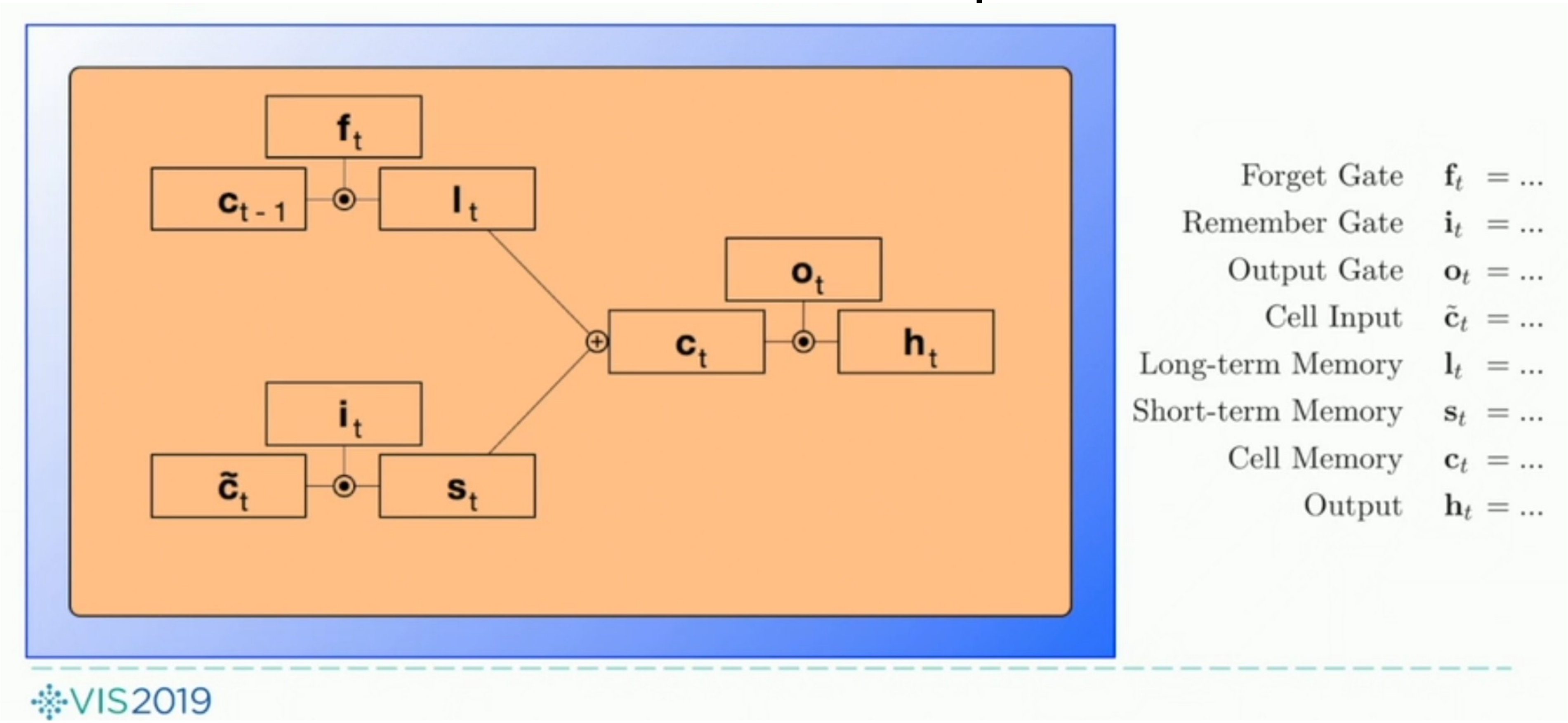
Bidirectional RNN

- Later information may be used to reassess previous observations



LSTMs

- Use addition over time instead of multiplication



[Sawatzky, Bergner, Popowich, 2019]

Further Architectures

- Transformers
- Deep Reinforcement Learning

Excellent explanation of Attention

Karpathy's NanoGPT

https://www.youtube.com/watch?v=kCc8FmEb1nY&t=1s

NanoGPT implementation

https://github.com/karpathy/nanoGPT

Generative language models are passing exams

Exam	GPT-4	GPT-4 (no vision)	GPT-3.5
Uniform Bar Exam (MBE+MEE+MPT)	298 / 400 (~90th)	298 / 400 (~90th)	213 / 400 (~10th)
LSAT	163 (~88th)	161 (~83rd)	149 (~40th)
SAT Evidence-Based Reading & Writing	710 / 800 (~93rd)	710 / 800 (~93rd)	670 / 800 (~87th)
SAT Math	700 / 800 (~89th)	690 / 800 (~89th)	590 / 800 (~70th)
Graduate Record Examination (GRE) Quantitative	163 / 170 (~80th)	157 / 170 (~62nd)	147 / 170 (~25th)
Graduate Record Examination (GRE) Verbal	169 / 170 (~99th)	165 / 170 (~96th)	154 / 170 (~63rd)
Graduate Record Examination (GRE) Writing	4 / 6 (~54th)	4 / 6 (~54th)	4 / 6 (~54th)
USABO Semifinal Exam 2020	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	43 / 150 (31st - 33rd)
USNCO Local Section Exam 2022	36 / 60	38 / 60	24 / 60
Medical Knowledge Self-Assessment Program	75 %	75 %	53 %
Codeforces Rating	392 (below 5th)	392 (below 5th)	260 (below 5th)
AP Art History	5 (86th - 100th)	5 (86th - 100th)	5 (86th - 100th)
AP Biology	5 (85th - 100th)	5 (85th - 100th)	4 (62nd - 85th)

Visualization for DL

- **Tensorboard: Visualizing Learning**
- How to use t-SNE efficiently
- UMap

Model visualization

- **LSTM-Vis**: http://lstm.seas.harvard.edu/client/index.html
- Video demo
- Building blocks of interpretability

Sources

- I. Goodfellow, Y. Bengio, A. Courville “Deep Learning” MIT Press 2016 [link]
- Zhang et al. “Dive into Deep Learning” [link]

NLP tasks for Data Science

CMPT 733
Steven Bergner
Slides in part by: Suraj Swaroop (Summer coop, 2020)

What is NLP?

Natural Language
how humans communicate with each other via **speech and text**

Processing

- branch of AI to read, decipher, and make sense of human language
- Applications: information extraction, translation, personal assistants, word processors, spam detection, ...

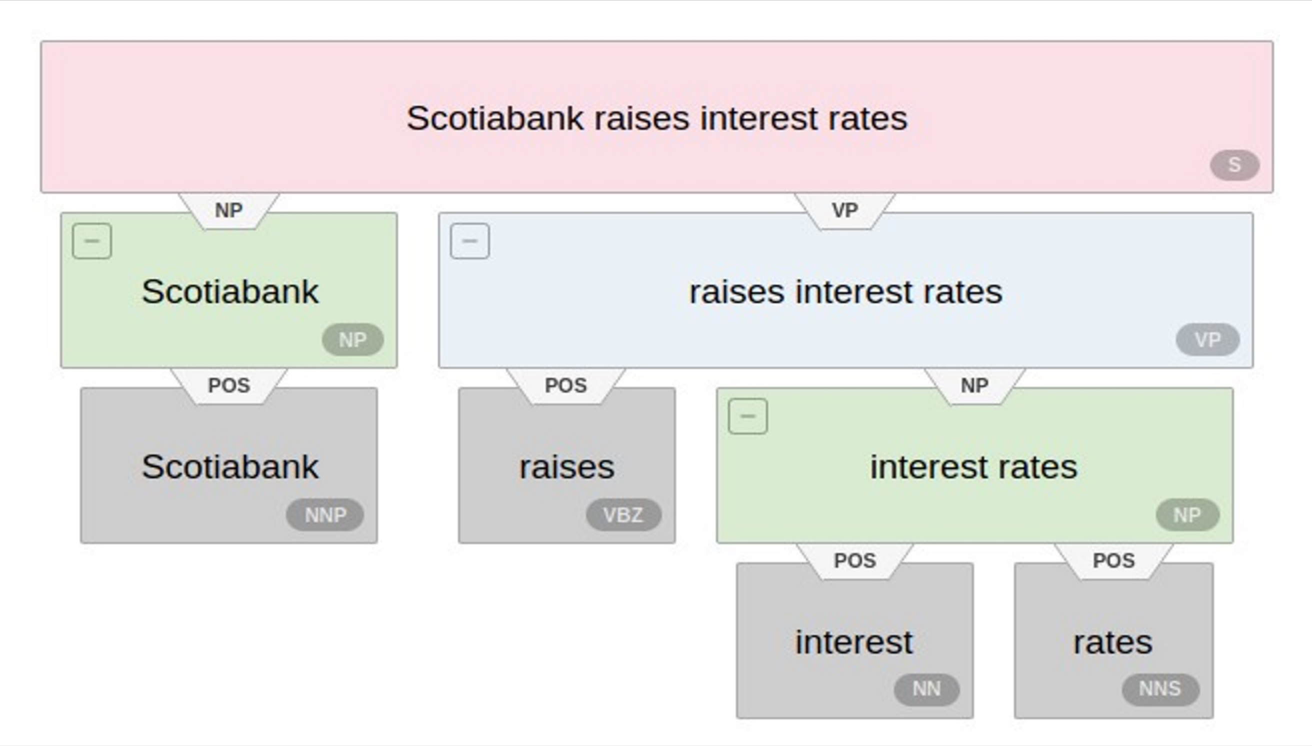
Techniques for NLP

Text Parsing

- Analyzing sentence structure and representing it according to syntactic formalism

- Two views of syntactic structure
    - Constituency
    - Dependency

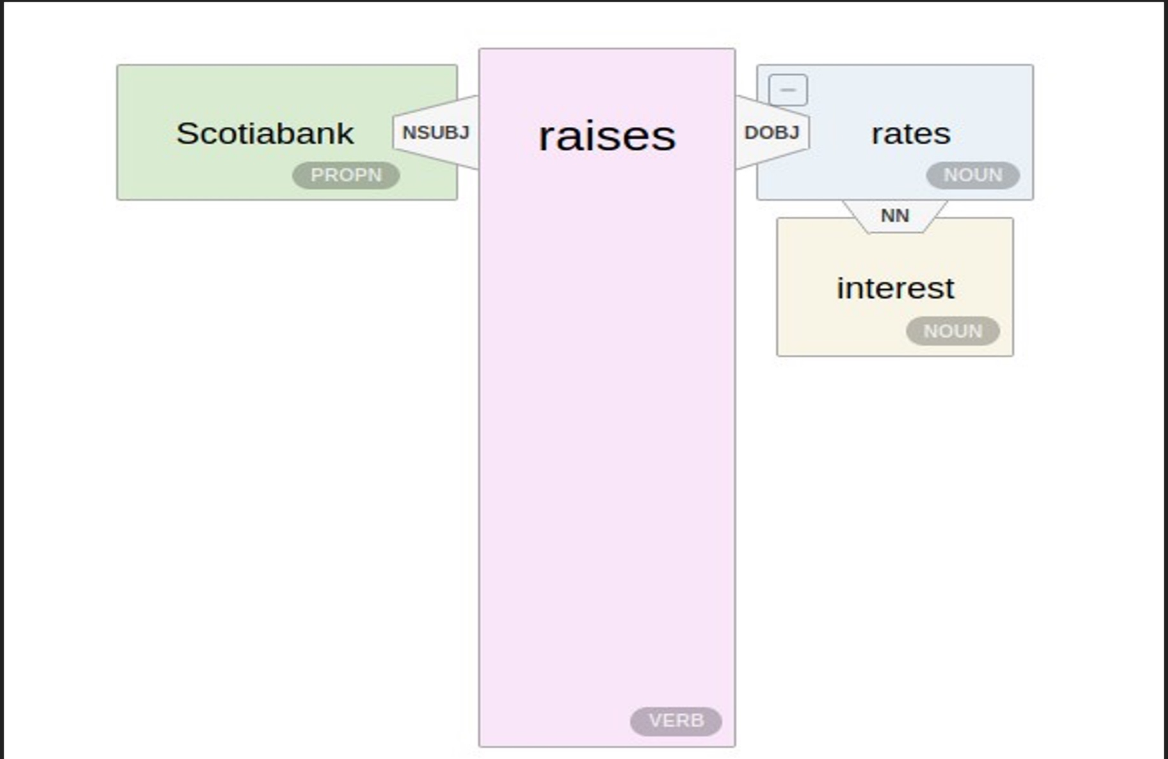
Constituency Structure Example



Constituency Parsing Implementation

```
from constituent_treelib import ConstituentTree, BracketedTree, Language, Structure

# Define the language for the sentence as well as for the spaCy and benepar models
language = Language.English

# Define which specific SpaCy model should be used (default is Medium)
spacy_model_size = ConstituentTree.SpacyModelSize.Medium

# Create the pipeline (note, the required models will be downloaded and installed automatically)
nlp = ConstituentTree.create_pipeline(language, spacy_model_size)

without_token_leaves = ConstituentTree(sentence, nlp, Structure.WithoutTokenLeaves)
```

Dependency Structure Example



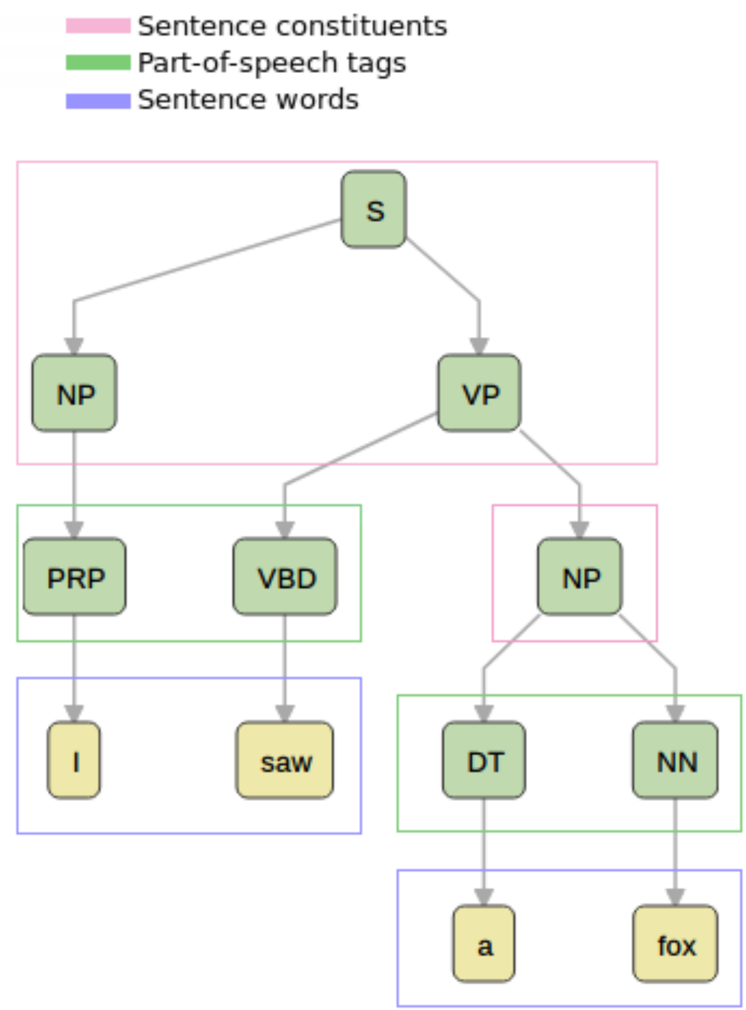
Dependency vs Constituency Tree

Dependency parsing

- only relationships between words and their constitutes

Constituency parsing

- entire sentence structure and relationships between phrases

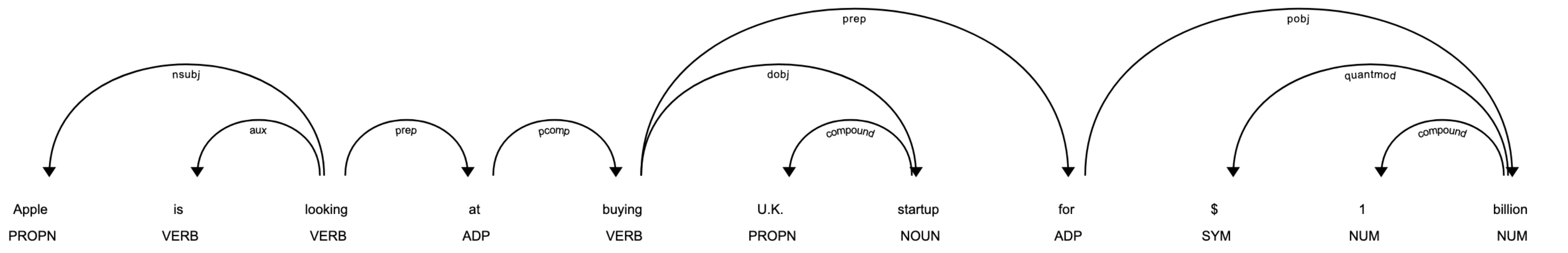


Phrase examples

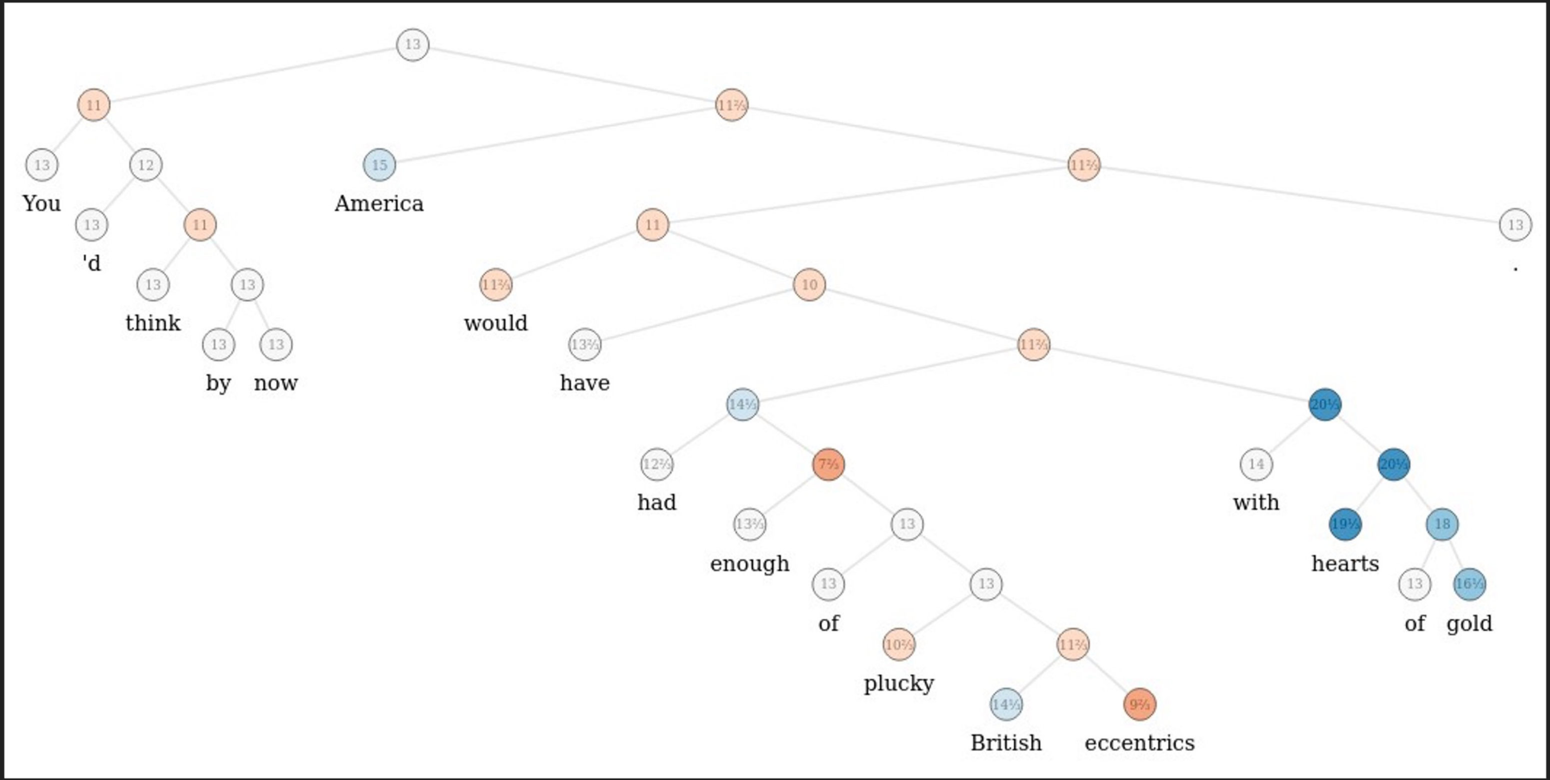
label	long name	example (represented by terminal string)
NP	noun phrase	their public lectures
VP	verb phrase	built the pyramid
PP	preposi- tional phrase	in the five chambers
S	sentence	Khufu built the pyramid
SBAR	sbar	that Khufu built the pyramid

Dependency tree

“Apple is looking at buying U.K. startup for $ 1 billion”



Tree Example [Stanford Sentiment Treebank]



Information Extraction

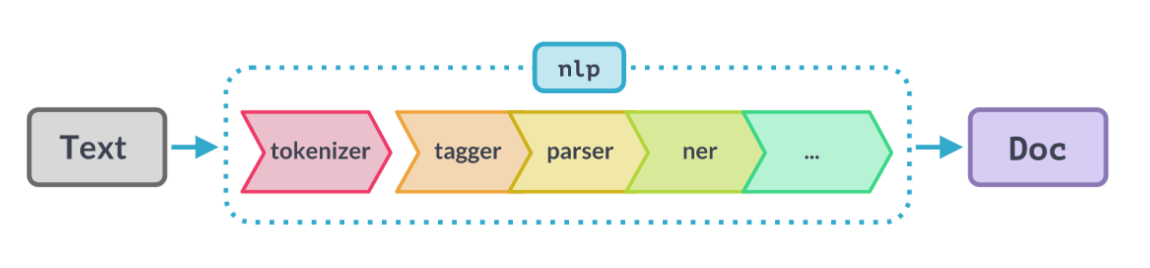
- Automatic extraction of structured and unstructured information

- Various modules
  - POS Tagging
  - Entity Recognition
  - Relation extraction
  - Sentiment Analysis

Named Entity Recognition

- Classify named entities into categories
- NER Techniques
  - Lexicon approach
  - Rule-based systems
  - ML based systems
  - Hybrid approach

NER Implementation




```
nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

pd.DataFrame(((token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
            token.shape_, token.is_alpha, token.is_stop) for token in doc),
             columns="Text  Lemma POS Tag Dep Shape alpha stop".split())
```

- **Text:** The original word text.
- **Lemma:** The base form of the word.
- **POS:** The simple UPOS part-of-speech tag.
- **Tag:** The detailed part-of-speech tag.
- **Dep:** Syntactic dependency, i.e. the relation between tokens.
- **Shape:** The word shape – capitalization, punctuation, digits.
- **is alpha:** Is the token an alpha character?
- **is stop:** Is the token part of a stop list, i.e. the most common words of the language?

Sentiment Analysis

- Determine if an opinion is positive, negative or neutral
- Techniques for Sentiment Analysis
  - Lexical Methods
  - Machine Learning methods

Sentiment Analysis Implementation

```
[8] import nltk
    nltk.download('vader_lexicon')
    from nltk.sentiment.vader import SentimentIntensityAnalyzer
    sid = SentimentIntensityAnalyzer()
    sid.polarity_scores("I am happy today")

{'compound': 0.5719, 'neg': 0.0, 'neu': 0.351, 'pos': 0.649}
```

Part of speech Tagging

- Tags each word with its corresponding part of speech
- Techniques of POS
  - Lexical Based Methods
  - Rule Based Method
  - Probabilistic Method
  - Deep learning models

POS Tagging Implementation

(use spacy instead)

```
import nltk
from nltk import word_tokenize
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
text = word_tokenize("He would not accept anything of value from those he was writing about")
nltk.pos_tag(text)
```

```
[('He', 'PRP'),
 ('would', 'MD'),
 ('not', 'RB'),
 ('accept', 'VB'),
 ('anything', 'NN'),
 ('of', 'IN'),
 ('value', 'NN'),
 ('from', 'IN'),
 ('those', 'DT'),
 ('he', 'PRP'),
 ('was', 'VBD'),
 ('writing', 'VBG'),
 ('about', 'IN')]
```

Semantic Role Labeling (SRL)

- Assigning labels to words or phrases in a sentence to indicate it’s semantic role

- How it works:
    - Predicate identification
    - Predicate disambiguation
    - Argument identification
    - Argument classification

For Example

“He wouldn’t accept anything of value from those he was writing about”

The annotations of semantic roles for this sentence:

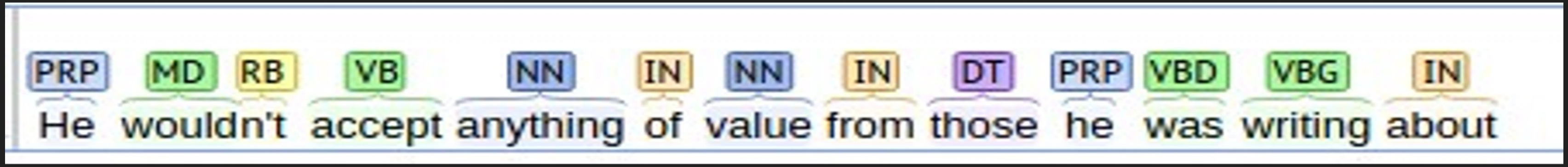
$[_{A0}$ He $]$ $[_{AM\text{-}MOD}$ would $]$ $[_{AM\text{-}NEG}$ n't $]$ $[_{V}$ accept $]$ $[_{A1}$ anything of value $]$ from $[_{A2}$ those he was writing about $]$.

V: verb; A0: acceptor; A1: thing accepted; A2: accepted-from; A3: attribute;
AM-MOD: modal; AM-NEG: negation

Difference between POS and SRL

Sentence: "He wouldn't accept anything of value from those he was writing about"

The annotations of POS Tagging:



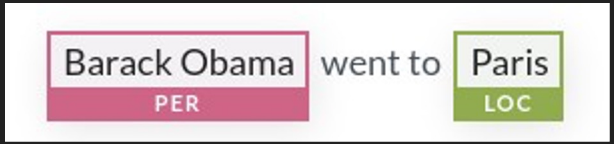
The annotations of semantic roles for this sentence:

[$_{A0}$ He] [$_{AM\text{-}MOD}$ would] [$_{AM\text{-}NEG}$ n't] [$_{V}$ accept] [$_{A1}$ anything of value] from [$_{A2}$ those he was writing about] .

NER and SRL

Sentence: “Barack Obama went to Paris “

The annotations of Entity Recognition Tagging:



The annotations of semantic roles for this sentence:

[$_{\text{ARG0:}}$ Barack Obama] [$_{\text{V:}}$ went] [$_{\text{ARG4:}}$ to Paris]

Combining ER and SRL

SA and NER

- Document-level sentiment analysis
  - Documents may have multiple topics
  - Not enough granularity

- Entity sentiment analysis identifies sentiment of each word
  - know how specific people, organizations, or things are being mentioned

Applications of SRL

- Question Answering system
- Summarization
- Information Extraction

Tools

Tools for NLP

- **NLTK** - legacy baseline, dictionary and rule based methods
- **Spacy**
  - Supports several different languages
- **Huggingface transformers** [github/demos]
  - Many state-of-the-art pre-trained models
- **AllenNLP** - platform for solving natural language processing tasks in PyTorch
- **Torchtext** – text processing support for PyTorch

Thank You