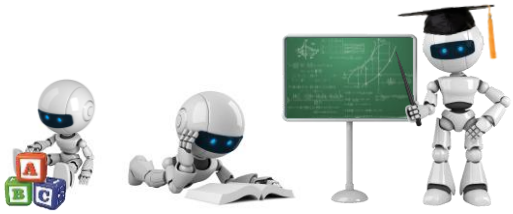
USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Who is Teaching This Course?

Mohammad Rostami

Ron Artstein

TAs

Ehsan Qasemi

Setareh Gilani

Yuliang Cai

Zizhao Hu

Graders

Ruitong Sun

Yongkang Du

Aditya Jain

TBD

Online Oriented

- Course Syllabus: check the website continuously

https://sites.google.com/view/usc-csci544-fall2022

- Course Blackboard: course material, quizzes & exams, announcement, assignments, etc.

https://blackboard.usc.edu/

- Course Zoom ID: 928 3276 9474
- Readings: available on syllabus and publicly available on the internet

- Read them before each lecture. Quizzes may include reading content that was not specifically covered in the class

- Piazza: participate in discussions
- Course Slack Channel

- fall22-csci-544-30249

Grading and Logistics

- Grading Schema

•5% Written application idea (individual; peer-graded; Due: September 15, 2022)

•40% Coding assignments

•10% Quizzes

•5% Presentations of recent research papers (in groups; peer-graded)

•35% Research project (in groups; parts of the project are self- and peer-graded)

•5% Final exam: an on-line, take-home comprehensive exam taken during the final exam period

- Hard deadline: before start of the class
- Regarding requests: one week after announcing the grades (except the final exam)

Grading and Logistics

- Collaboration

- Collaboration on HW is allowed but your code and report should be solely your own work. It is your responsibility to make sure your solution is not shared or copied by others.
- We will check for overlap in HW and treat it harshly

- PyTorch Programing

- Expected to be self-studied
- A short introduction in the class
- Computational resources: Google ($50), Microsoft Azure ($100), AWS ($35), Google Educational Program ($50)

https://gcp.secure.force.com/GCPEDU?cid=RA%2BJNX8uwSjMADyP%2FTDa%2FUXRKYfBsrLyWpiPMFTNJwkCsBW2W46XsQ8LW24TXFf1/

- Plagiarism

Course Timeline

Important dates
HW assignments: 8/25 to 10/25
Project Proposal: 10/4
Project Status Report: 11/3
Project Final Report: 12/01
Final Exam: 12/08

Course Project

- Groups of five:

- Start your groups early: https://docs.google.com/spreadsheets/d/1IvM828-DojX9xeRzpg6GAUMZT894ojFySu16nAaBJXA/edit#gid=0
- Slack channel
- Randomly grouped

- Should have research and exploration nature (has not been done before but doable during the current semester):

- Your own topic, e.g., exploring your native language using existing tools, extending your prior work.
- Implementing and generating results for a good paper along with additional experiments to improve the paper
- Topics will be suggested
- Extra credit for interesting results

Applied Natural Language Processing

- Applied: we work with real language data
- Language: a structured system of communication based on signs, often in written form.
- What is a natural language?

- A means of communication that has evolved naturally in humans through use and repetition without conscious planning, e.g., English, Hindi, Chinese, Tamil, Arabic, etc.
- How about sign languages?

- What is **not** a natural language?

- Programing language: Python, etc.
- Formal languages: first order logic

Applied Natural Language Processing

- Processing: how to program computers to analyze large amounts of natural language data.

- **Natural language understanding**
- Natural language generation
- Speech understanding: https://www.nationalgeographic.com/animals/article/scientists-plan-to-use-ai-to-try-to-decode-the-language-of-whales

- Applications

- Keyword Search
- Spell Checking
- Chatbot
- Machine Translation
- Dialogue Systems
- Grammar Checking

Natural Language Levels of Representation

- Linguistics: the scientific study of natural language

Linguistics	NLP
Phonetics, Phonology	Speech recognition, synthesis
Morphology	Lemmatization, Stemming
Syntax	Part-of-speech tagging, Parsing
Lexical Semantics	Entity recognition
Compositional Semantics	Role labeling, Reference resolution Text classification
Production	Language generation Summarization Machine translation

Evolution of NLP

- Advancement of NLP

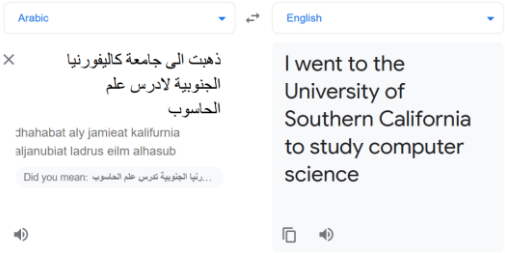
- Rule Based NLP:

  Who is the prime minister of India?

- NLP based on statistical learning: learning from extracted features

I don't like scary movies, but this movie is funny at the same time and that is why I liked it.

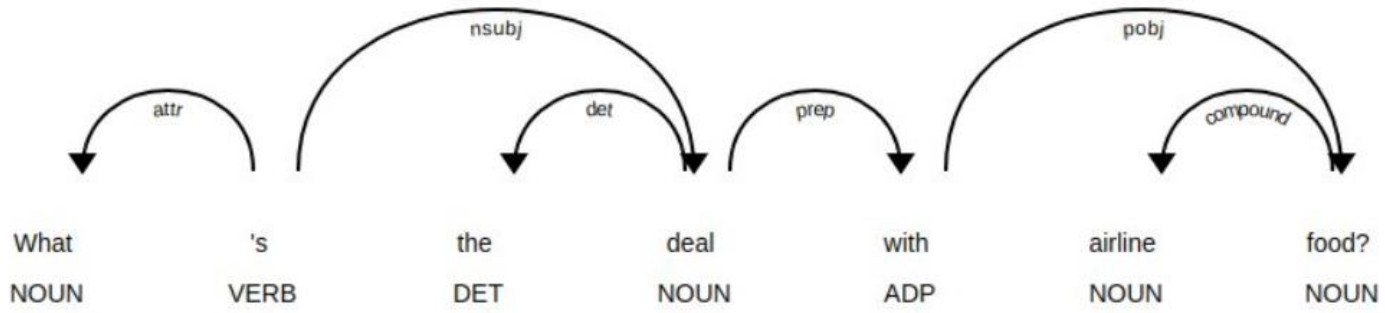
- Deep Learning: learning directly from large textual datasets



Rule-Based NLP

- A hand-crafted system of rules that parse text and match patterns is used to imitate the human way of building language structures.

- EX: Question Answering



- It can have high performance in specific use cases, e.g., question answering, but often suffer performance degradation in generalized settings
- Requires domain knowledge about the language

NLP Using Statistical Learning

- Learning is performed based on probabilistic modeling, likelihood maximization, and training classifiers.

- EX: sentiment analysis (HW 1)

- Requires an annotated training dataset along with a suitable method of feature engineering, potentially based on domain knowledge about the language.

- A parametric model is trained followed by evaluation on a test dataset similar, yet different, to the training dataset

- More generalizable than rule-based NLP and applicable to a broader range of applications, e.g., machine translation.

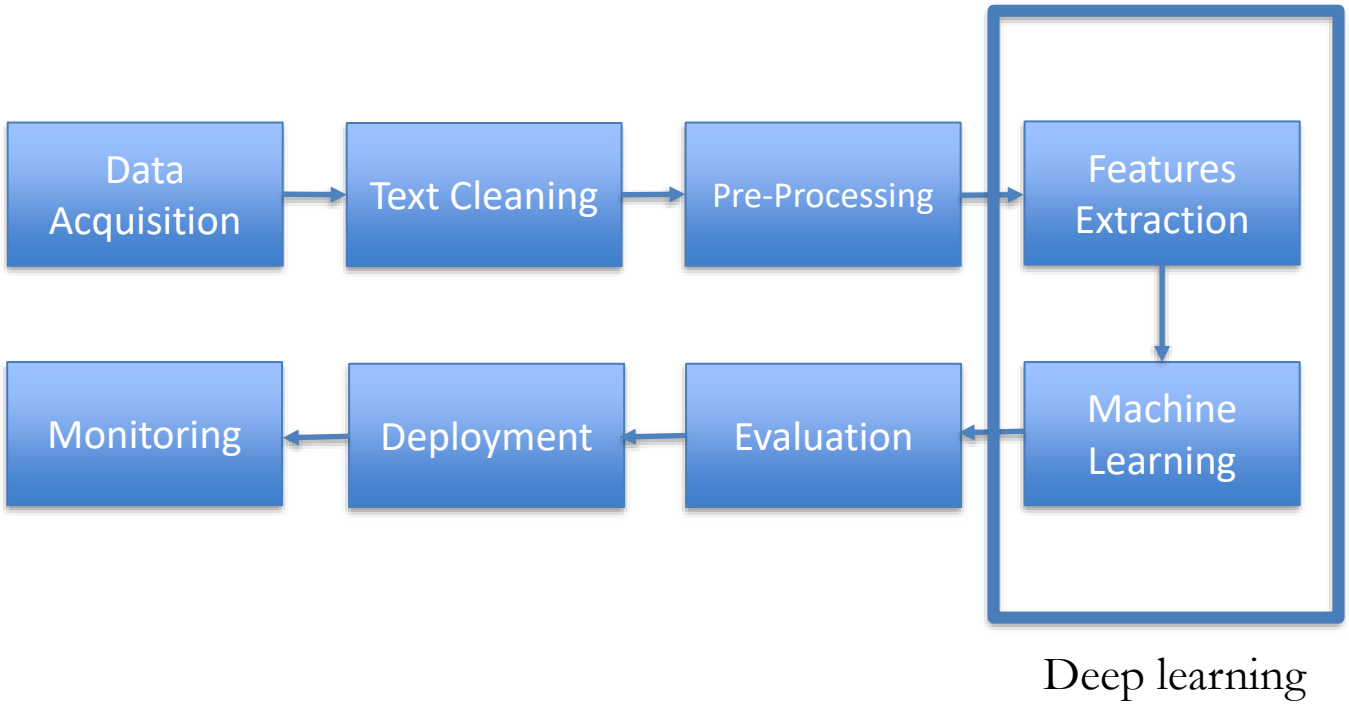
NLP Using Deep Learning

- Similar to machine learning with key differences:
- Feature engineering is automated, as deep networks can learn important features from text -> domain knowledge about langue is minimized
- Raw text with minimal pre-processing is fed into the models
- It requires very large datasets to circumvent lack of using human domain knowledge
- Applicable to more challenging tasks, e.g., dialogue systems.

- Why do we still use rule-based NLP and traditional ML:
- Simple and still good for some tasks, e.g., sequence labeling
- Some ideas in deep learning are extended versions of earlier methods
- Can help to improve deep learning-based algorithms

NLP Pipeline

- We will mostly explore the stages between text cleaning and evaluation



Challenges of Preprocessing: Ambiguity in Natural Language

- Ambiguity
  - The shortstop caught the fly (lexical ambiguity)
  - Flying planes can be dangerous (structural/syntactic ambiguity)

- Local ambiguity vs global ambiguity: context
  - Fat people eat accumulates.
  - The man who hunts ducks out on weekends.

- Psycholinguistics: the study of human sentence processing

- Should NLP be done similar to human processing?

Preprocessing

- What is a word? Two words or one word?

- New York
- Hand-writing -> Handwriting
- He's; It'll -> He is; It will (contraction)
- David's book, David's happy
- Lower case

- **Tokenization**: splitting the text into units for processing

- Anywhere on the scale character to word
- Removing extra spaces
- Removing stop words
- Removing unhelpful words, e.g., external URL links
- Removing unhelpful characters, e.g., non-alphabetical characters.
- Ex: Tokenization is the first step in NLP. (7 tokens)

Preprocessing

- **Stemming:** wordform stripped of some characters
- tokenization -> tokenize
- **Lemmatization**: the base (or citation) form of a word
- Tokenization, tokenize -> token
- Went, gone, goes -> go
- Lemmatization can be lossy
- EX: Where were you born? vs Where is your bear?
- Is it important to be able to reconstruct the original?
- Implemented in the Python NLTK (assignment)
- Example:
- I ordered these so I could fill up my Pandora bracelet right away, I didn't wanna have to wait for 10 more holidays before it was filled up. I liked the fact that they had a lot of pink in the charms, but barely got any in my shipment. I still have some of these charms on my bracelet but I don't wear it often since it turns my wrist green.
- ordered could fill pandora bracelet right away wan na wait holiday filled liked fact lot pink charm barely got shipment still charm bracelet wear often since turn wrist green

Machine Learning

- Preprocessing + feature extraction converts input text into feature vectors

- Sentiment analysis:

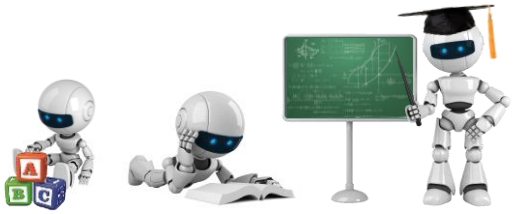
I don't like scary movies, but this movie is funny at the same time and that is why I liked it. -> Vector f with label +1 to denote positive sentiment

- Training a classifier using annotated features is a relatively solved problem in machine learning

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- HW1 is out: start working early
- NLTK
- sklearn
- Quiz1: Blackboard
- Recorded lectures: blackboard -> tools -> usc zoom pro meeting -> cloud recordings.
- Groups: 10 groups are formed. Continue to form your groups soon.
- TA office hours: announcements on Blackboard
- Happy to see some people started to form groups and already learning using NLTK

Classification

- Categorizing instances of data into "classes", where class members share some notion of similarity, e.g., having positive sentiment

Parametric Model:
Learning $\approx$ Choosing and selecting the **best** model

$$y_i = h(\theta) \approx \theta^\top x_i, \theta \in \mathbb{R}^d$$

Class 0

Class 1

$$\mathbf{X} = [x_1, \ldots, x_N] \in \mathbb{R}^{d \times N}, \mathbf{Y} \in \{0, 1\}^N$$

Training Dataset
Testing Dataset

Model Selection:

$$\hat{\theta} = \arg \min_{\theta \in \Omega(\theta)} \underbrace{\sum_i \mathcal{L}(\theta^\top x_i, y_i)}_{\text{Empirical Error}}$$

Features!

Preprocessing

Tokenization: splitting the text into units for processing

- Removing extra spaces: the quality is high
- Removing stop words: article, propositions, etc.

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

- Contractions are standardized: won’t -> will not
- Removing unhelpful words, e.g., external URL links
- Removing unhelpful characters, e.g., non-alphabetical characters.
- Converting capital letter to lowercase

Preprocessing

- **Stemming:** wordform stripped of some characters
- **Lemmatization**: the base (or citation) form of a word
- Example:

- ordered could fill pandora bracelet right away wan na wait holiday filled liked fact lot pink charm barely got shipment still charm bracelet wear often since turn wrist green

```
ps = nltk.stem.porter.PorterStemmer()
print([ps.stem(word) for word in txt])
```

```
['order', 'could', 'fill', 'pandora', 'bracelet', 'right', 'away', 'wan', 'na', 'wait', 'holi
day', 'fill', 'like', 'fact', 'lot', 'pink', 'charm', 'bare', 'got', 'shipment', 'still', 'ch
arm', 'bracelet', 'wear', 'often', 'sinc', 'turn', 'wrist', 'green']
```

```
ps = nltk.stem.WordNetLemmatizer()
print([ps.lemmatize(word) for word in txt])
```

```
['ordered', 'could', 'fill', 'pandora', 'bracelet', 'right', 'away', 'wan', 'na', 'wait', 'ho
liday', 'filled', 'liked', 'fact', 'lot', 'pink', 'charm', 'barely', 'got', 'shipment', 'stil
l', 'charm', 'bracelet', 'wear', 'often', 'since', 'turn', 'wrist', 'green']
```

Feature Extraction

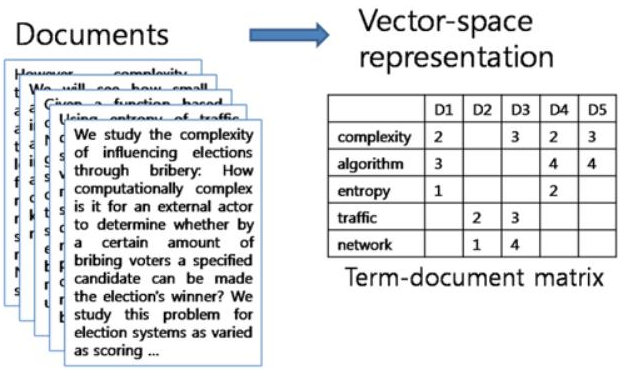
- We start from raw data, e.g., text, and build values, i.e., features, intended to be informative and non-redundant to help the subsequent learning and generalization steps.

- Is a non-trivial task and specific to the application
- The quality is judged by performance
- A tedious task as applications become more abstract, e.g., document categorization vs document summarization

- Example: Are you satisfied with our service?

Bag of Words (BoW)

- Simple example: counting the number of “informative” words in the input text
- Bag of words:
- we pick a dictionary of words and then put an arbitrary order on the words, e.g., alphabetical order
- We convert the text into a feature vector by reporting the frequency of occurrence of the words in the text
- Ex:
- Dic = [good, bad, nice, expensive, love]
- [I love this shirt because it is nice and worm. The color also is nice and matches my skin tone] -> [0, 0, 2, 0, 1]



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Bag of Words (BoW)

- Limitations of bag of ward:

- Insensitive to language structure: “I wanna eat ice cream” vs “wanna eat ice cream?”
- Information in word dependencies is overlooked: “new york” vs “new book”
- The resulting vectors are highly sparse which leads to high computational costs

- Why do we use BoW?

- Simple
- Leads to acceptable performance in some applications

Term Frequency - Inverse Document Frequency

- Core idea: reflect how important a word is to an instance in the dataset.
- Term Frequency: a measure of how frequently a term, t, appears in an instance (document), d:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

- The frequency of occurrence is normalized

- Inverse Document Frequency: a measurement of how distinguishing a term is in the dataset.

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Term Frequency - Inverse Document Frequency

- TF-IDF score: reflects importance of the word for a particular instance and informative to categorize the documents

$$(tf_idf)_{t,d} = tf_{t,d} * idf_t$$

- We use TF-IDF to extract the feature vector
- Ex: What is TF-IDF score for stop words?

Implemented in NLTK

Statistical Learning

- Problem: finding a **predictive** function based on an annotated training dataset using probability theory $(\mathbf{x}_i, \mathbf{y}_i) \sim p(\mathbf{x}, \mathbf{y})$

- Goal: given the value of an input vector *X*, i.e., features, make a good prediction $\hat{Y}$ of the output *Y* (i.e., $\hat{Y} = Y$ with high probability) using the predictive function

- Maximum posterior estimation: $P(Y|X)$

True Error/Risk

Generative vs. Discriminative Models

Generative

- Learn a model of the joint probability P(X, Y)
- Use Bayes' Rule to calculate P(Y|X)
- Return the class that most likely to have generated that instance
- Examples: Naïve Bayes

Discriminative

- Model posterior probability P(Y|X) directly
- Class is a function of feature vector
- Find the exact function that minimizes classification errors on the training dataset and use it for prediction
- Examples: Linear classifier Logistic regression, Neural Networks (NNs), Support Vector Machines (SVMs)

Discriminative vs. Generative Classifiers

- Discriminative classifiers are generally more effective, since they directly optimize the classification accuracy. But
  - They are all sensitive to the choice of features, and in traditional ML these features are extracted heuristically
  - Overfitting can happen if datasets are small
- Generative classifiers directly model the joint probability which is helpful when generating text is necessary but:
  - Modeling the joint probability is a harder problem than classification if only classification is our goal
  - They are more vulnerable with respect to outliers

Bayes Classifier

- Bayes Rule:

$$\overset{\text{Posterior}}{P(Y|X)} = \frac{\overset{\text{Prior}}{P(Y)}\overset{\text{Likelihood}}{P(X|Y)}}{P(X)}$$

- Bayes Optimal Classifier:

$$\hat{Y} = \arg\max_{Y} P(Y)P(X|Y)$$

- We use multiple features

$$\hat{Y} = \arg\max_{Y} P(Y)P(X_1, \ldots, X_n|Y)$$

(Multinomial) Naïve Bayes Classifier

- Challenges for the Bayes optimal classifier
- Computing the joint likelihood probability is practically almost impossible

- Assuming statistical independence between the features:

$$\hat{Y} = \arg\max_{Y} P(Y)P(X_1, \ldots, X_n|Y)$$
$$= \arg\max_{Y} P(Y)\Pi_{i=1}^{n} P(X_i|Y)$$

(Multinomial) Naïve Bayes Classifier

- Challenges for the Naïve Bayes classifier

- Computing the likelihood probability even for single features is practically difficult because many words are not frequently used: superb vs good

$$\hat{Y} = \arg\max_{Y} P(Y)P(X_1, \ldots, X_n|Y)$$
$$= \arg\max_{Y} P(Y)\Pi_{i=1}^{n} P(X_i|Y)$$

- Zero probabilities cannot be conditioned away, no matter the other helpful evidence!
- Smoothing: we add small non-zero probabilities to avoid zero probabilities

Example

- Features/Dic = {I , hate, love, blue, shirt}

  ~ {X1, X2 , X3 , X4 , X5 }
- Training
  - I hate blue shirt (Y=0)
  - Love blue shirt (Y=1)

$$\text{Features}=\begin{bmatrix}1 & 1 & 0 & 1 & 1\\ 0 & 0 & 1 & 1 & 1\end{bmatrix}$$

- What is P(Y|X)?

- P(Xi=k|Y=j) = #(documents with Xi=k and Y=j)/ #(documents with Y=j)

$$P(X|Y)=\begin{bmatrix}P(X1=0|Y=0) & P(X2=0|Y=0) & P(X3=0|Y=0) & P(X4=0|Y=0) & P(X5=0|Y=0)\\ P(X1=0|Y=1) & P(X2=0|Y=1) & P(X3=0|Y=1) & P(X4=0|Y=1) & P(X5=0|Y=1)\\ P(X1=1|Y=0) & P(X2=1|Y=0) & P(X3=1|Y=0) & P(X4=1|Y=0) & P(X5=1|Y=0)\\ P(X1=1|Y=1) & P(X2=1|Y=1) & P(X3=1|Y=1) & P(X4=1|Y=1) & P(X5=1|Y=1)\end{bmatrix}$$

- Prior p(Y) $P(Y)=[P(Y=0) \quad P(Y=1)]$

- P(Y=j) = #(documents with Y=j)/ #(all documents)

- Testing
  - hate shirt {x2,X5}

$$P(Y|X)\propto[P(Y=0)\times P(X2=1|Y=0)\times P(X5=1|Y=0) \quad P(Y=1)\times P(X2=1|Y=1)\times P(X5=1|Y=1)]$$

Example

- Features = {I, hate, love, this, shirt}
- Training
  - I hate this shirt
  - Love this shirt
- What is P(Y|X)?
- Prior p(Y)
- Testing
  - hate shirt

$$P(X|Y) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$P(Y) = [1/2 \quad 1/2]$$

$$P(Y|X) \propto [1/2 \times 1 \times 1 \quad 1/2 \times 0 \times 1] \propto [1\ 0]$$

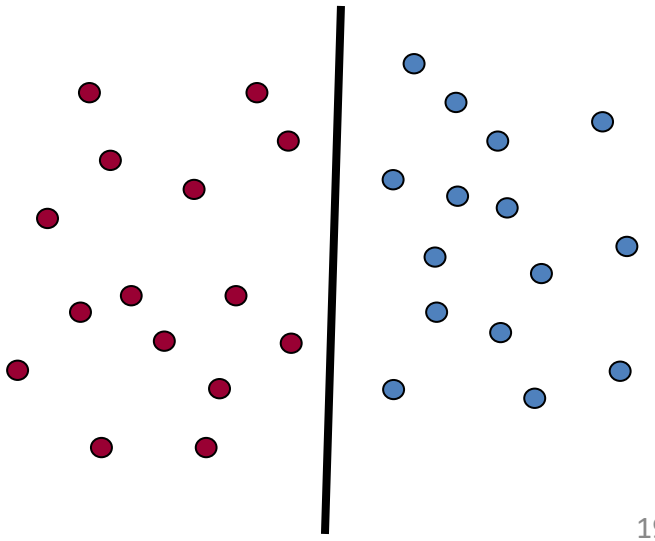
Implemented in sklearn

Linear Classifier

- An interpretation for discriminative models is that we find a boundary between the two classes in the **geometrical** features space

- A linear classifier assumes that the data points are linearly separable in the feature space

- The goal is to find a boundary



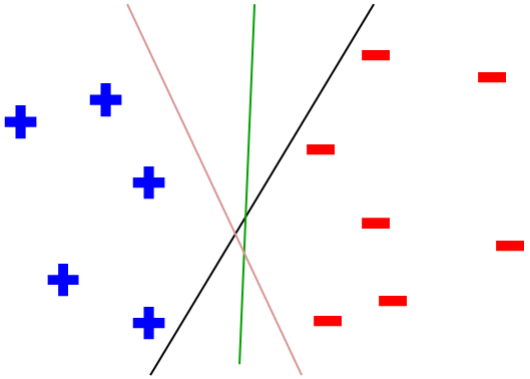
Linear models

- A linear function in *n*-dimensional space (i.e. we have *n* features) is define by *n*+1 weights:

$$Y = \sum_{i=0}^{n} \beta_i X_i$$

- We find the model weights such that the linear function acts as a good predictive model

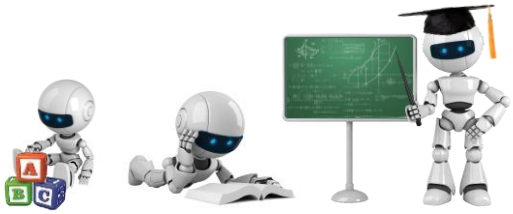
- Is not necessarily unique!



USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- HW1::
- Report: explaining how you solve the problem along with the requested output + Jupyter Notebook with printed output + executable .py fond (different software versions are OK but specify the version clearly in your report)
- Quiz: the quiz will be done on Thursday if Tuesday is a holiday. Unless mentioned, we have quizzes every week.
- Project Group Formation Deadline: 9/13
- We will form random assignment after this date
- Check slack/Piazza and the Excel sheet to find groups
- Meet weekly, helpful for both HW and project
- Proposal Deadline: 10/4
- TA research interests and office hours
- Written assignment
- Last year projects: Youtube
- Read papers from venues such as EMNLP, ACL, NAACL, etc. (often needed in industry as well)

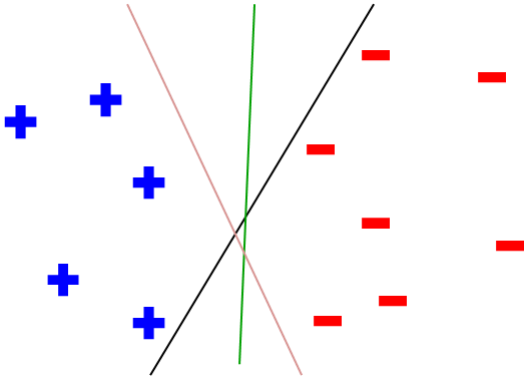
Linear models

- A linear function in *n*-dimensional space (i.e. we have *n* features) is define by *n*+1 weights:

$$Y = \sum_{i=0}^{n} \beta_i X_i$$

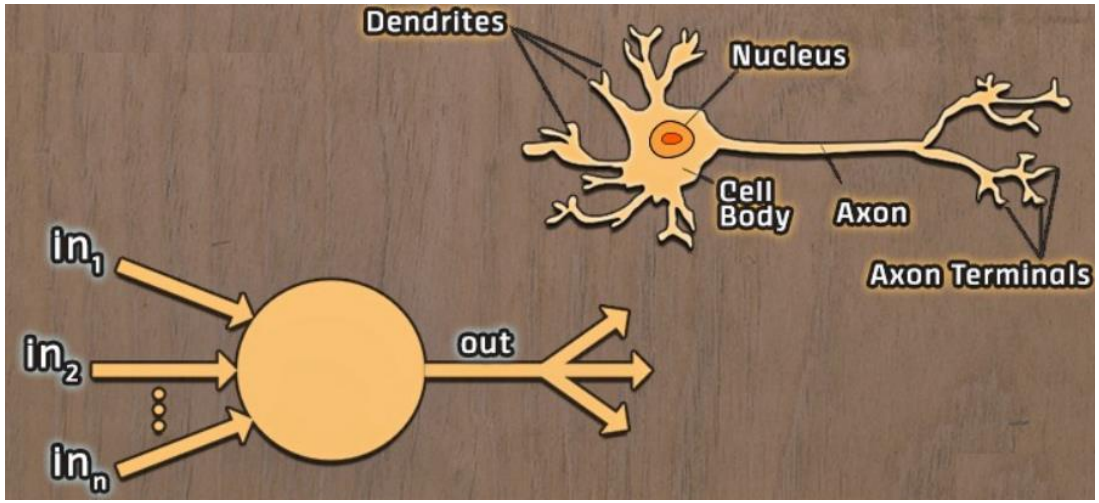
- We find the model weights such that the linear function acts as a good predictive model

- Is not necessarily unique!



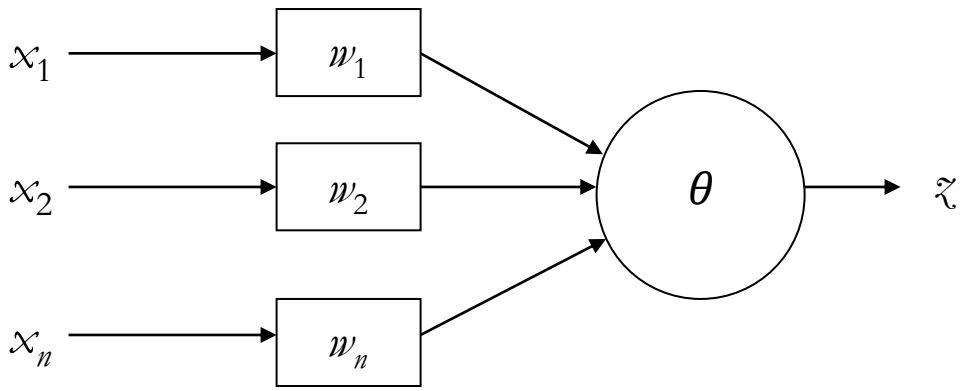
Perceptron

- Invented by Frank Rosenblatt in 1969
- Inspired by the nervous system
- Unit-based: analogous to a neural cell
- Model: neural activity is modeled by mathematical operations



Perceptron

- First neural network learning model in the 1960's
- Simple and limited (single layer models)
- Basic concepts are similar for multi-layer models so this is a good learning tool
- Still used in current applications (modems, etc.)

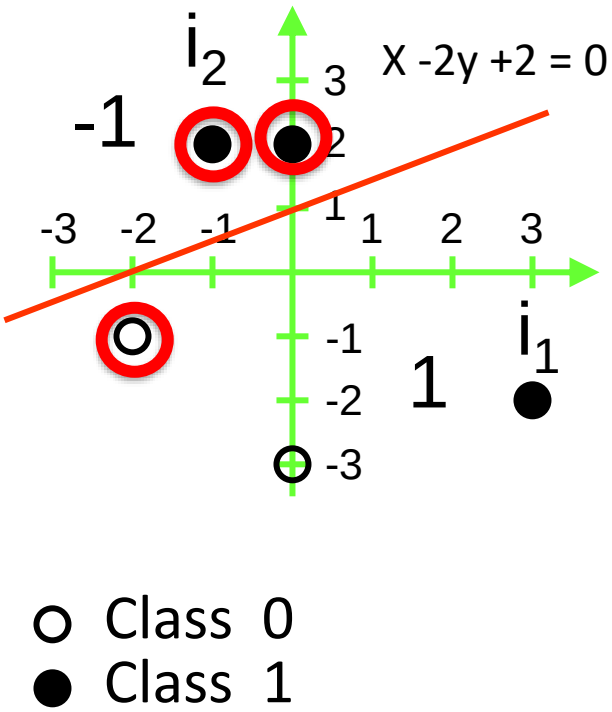


$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i\, w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i\, w_i < \theta \end{cases}$$

Perceptron Learning Algorithm

- An iterative algorithm:
- we initialize weights with random values
- we do several pass on the whole training dataset one by one and update the weights

- Least perturbation principle
  - Only change weights if there is an error
  - Use small *learning rate (l)* to change weights sufficiently to make the current pattern correct
- Each iteration through the training set is an *epoch*



Perceptron Learning Algorithm

- Let w_i be the weights vector at iteration, l be a constant (the learning rate), y be the label for the current iteration, z be the current **thresholded** output, and x be the input

$$\beta_{i+1} = \beta_i + \Delta\beta$$
$$\Delta\beta = l(y - z)x$$

y	z	$y - z$
0	0	0
0	1	−1
1	0	1
1	1	0

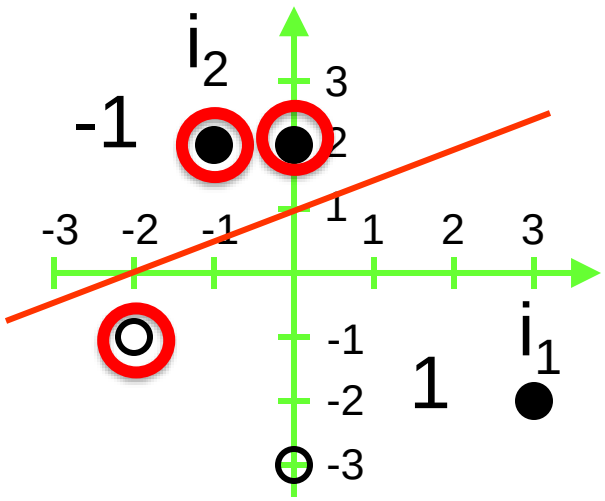
- Continue training until total training set error ceases to improve
- Perceptron Convergence Theorem: Guaranteed to find a solution in finite time if a solution exists

Perceptron Learning Example

We would like a perceptron to correctly classify the five 2-dimensional data points below.

Let the random initial weight vector $\boldsymbol{\beta}_0 = (1, -2, 2)^T$, why? (ax+by+c = 0 or [a b c][x y 1] ^{T} =0)

Then the dividing line crosses at $(0, 1)^T$ and $(-2, 0)^T$.



○ Class 0
● Class 1

Let us pick the misclassified point $(-2, -1)^T$ for learning:

$\mathbf{x}_1 = [-2, -1, 1]^T$ (include offset 1)

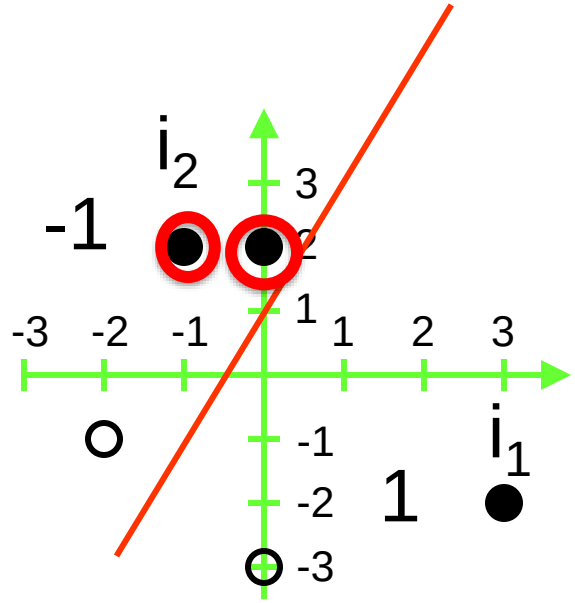
$\mathbf{z}_1 = [1,-2,2]\cdot(-2, -1, 1)^T = 2 \rightarrow 1$ ($\mathbf{x}_1$ is in class 0) $(\mathbf{y}_1 - \mathbf{z}_1 = -1)$

$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 - \mathbf{x}_1$ (let us set $\eta = 1$ for simplicity)

$\boldsymbol{\beta}_1 = (1, -2, 2)^T - (-2, -1, 1)^T = (3, -1, 1)^T$

The new dividing line crosses at $(0, 1)^T$ and $(-1/3, 0)^T$.

Perceptron Learning Example



○ Class 0
● Class 1

Let us pick the misclassified point $(-1, 2)^T$ for learning:

$\mathbf{x}_2 = [-1, 2, 1]^T$ (include offset 1)

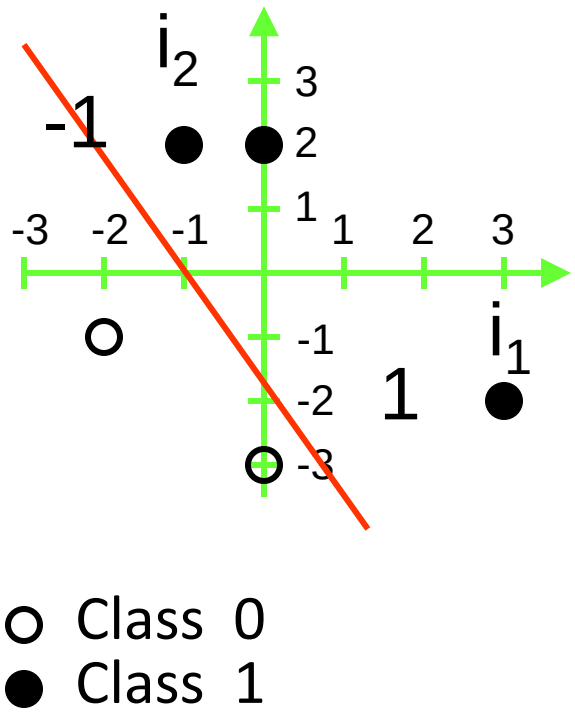
$\mathbf{z}_2 = [3,-1,1]\cdot(-1, 2, 1)^T = -4 \rightarrow 0$ ($\mathbf{x}_2$ is in class 1) ($\mathbf{y}_2 - \mathbf{z}_2 = 1$)

$\boldsymbol{\beta}_2 = \boldsymbol{\beta}_1 + \mathbf{x}_2$ (let us set $\eta = 1$ for simplicity)

$\boldsymbol{\beta}_2 = (3, -1, 1)^T + (-1, 2, 1)^T = (2, 1, 2)^T$

The new dividing line crosses at $(0, -2)^T$ and $(-1, 0)^T$.

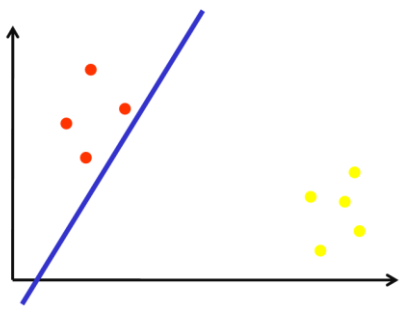
Perceptron Learning Example



- The algorithm has converged in one epoch because all the points are classified correctly
- In general, we may need several epochs
- Learning rate value is important for fast convergence
- At each epoch, we may use a different random order on the data points
- Neural networks, including the state-of-the-art deep networks are extensions of perceptron (potentially using different learning algorithms)

Optimal Boundary

- The perceptron learning algorithm is guaranteed to find a solution if the data points are linearly separable
- But are perceptron solutions optimal?



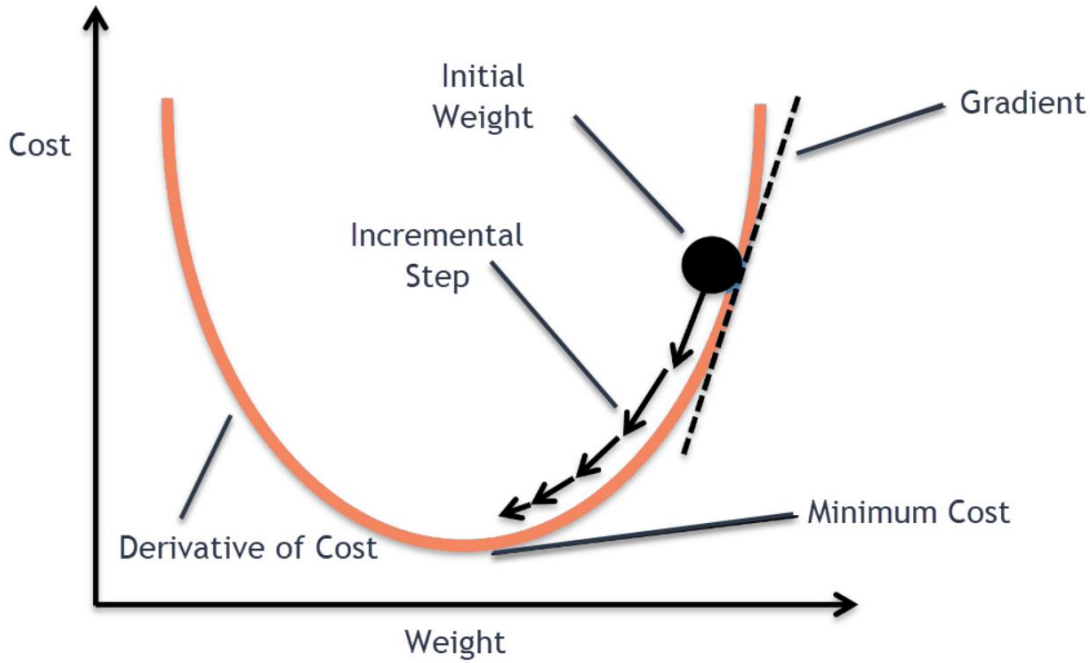
Perfect classification of training samples but may not generalize well to new (untrained) samples.

- Idea: compute a continuous, differentiable error function between input and desired output

- Define an objective function to measure quality of a model
- Find the weights for which the objective function is minimal.
- With a differential function, we can use the gradient descent techniques to find a good solution (at least locally optimal)

Gradient decent

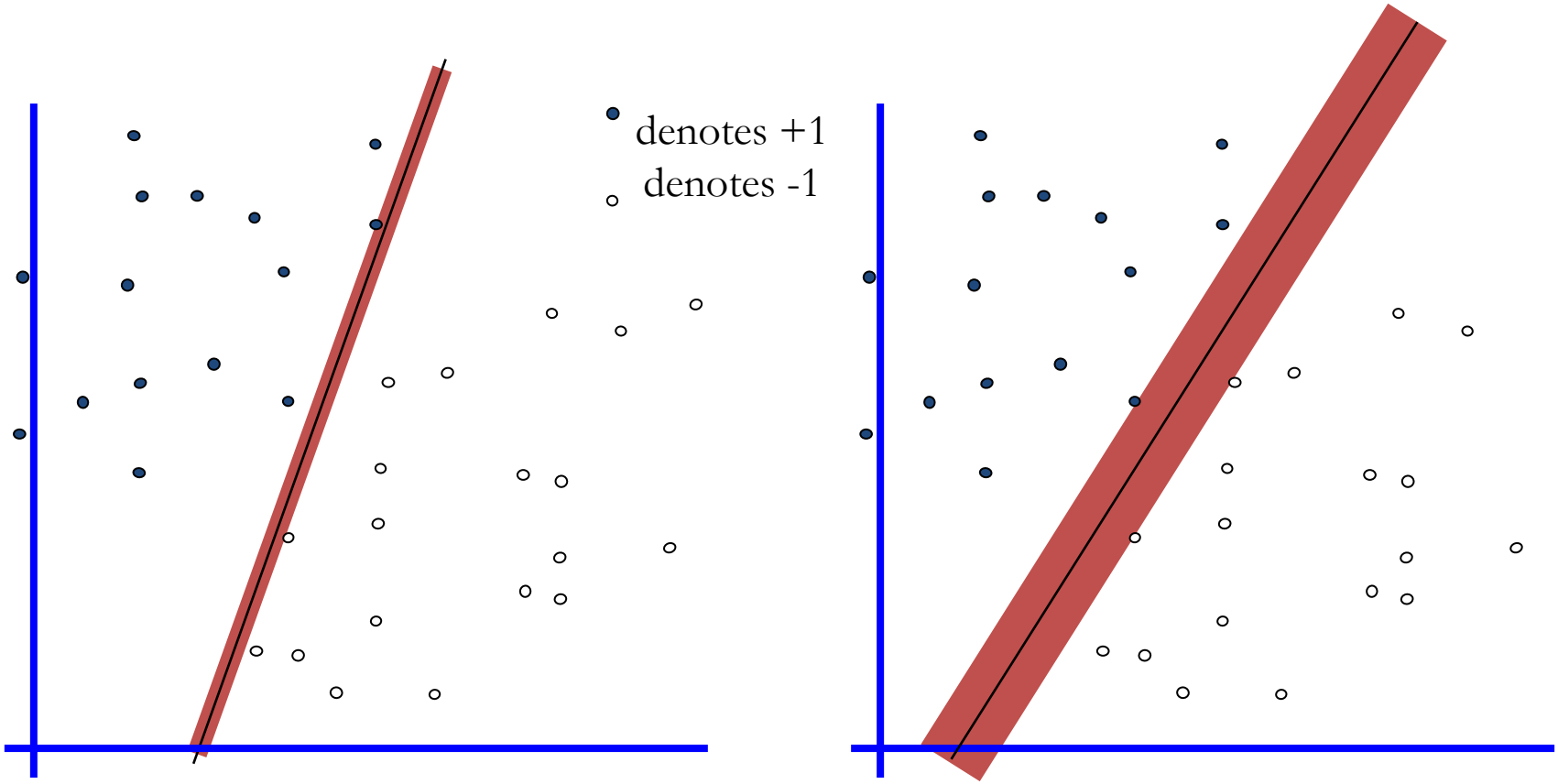
- Suitable for finding minimums of convex differentiable functions (cost functions)



$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Support Vector Machine

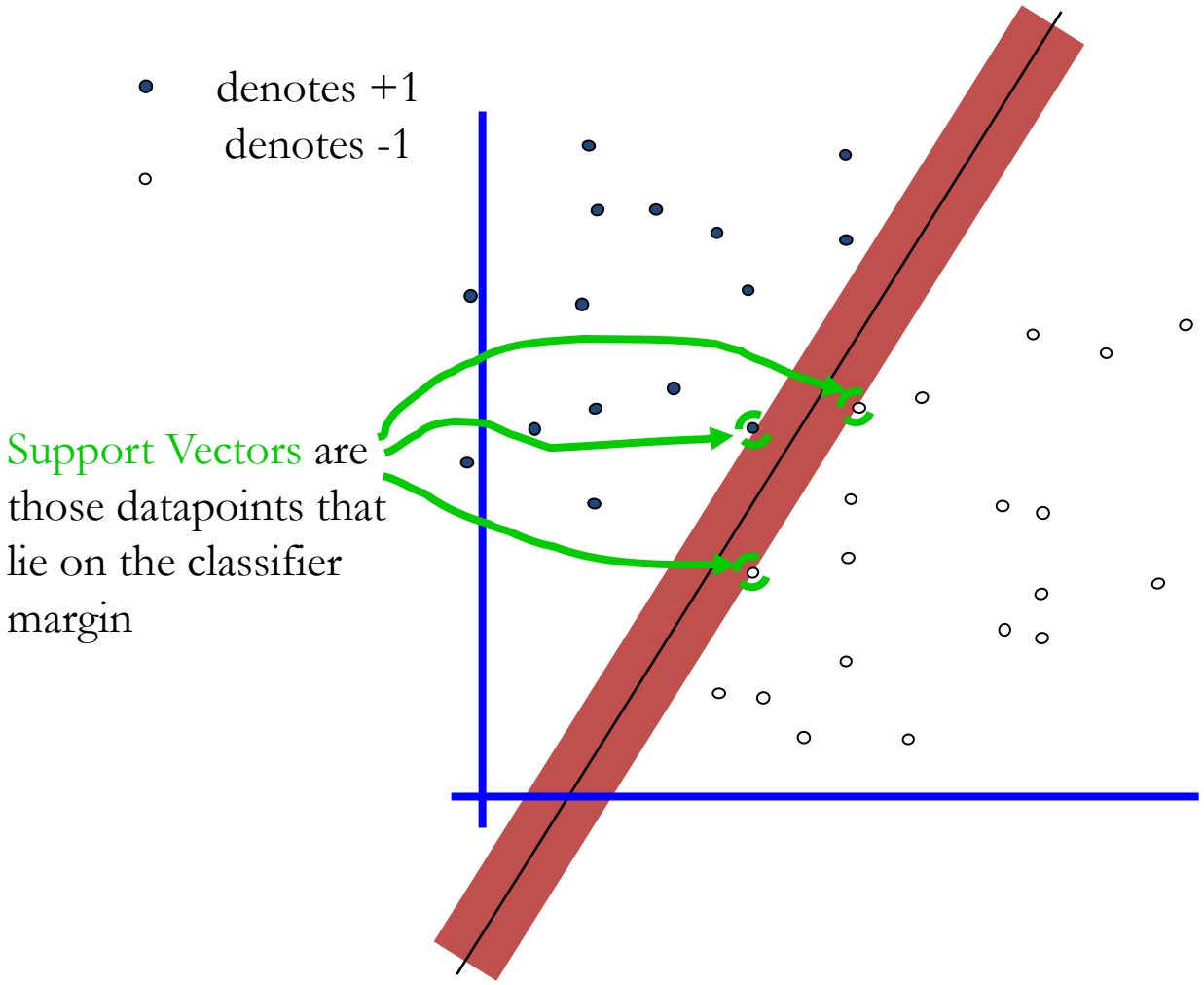
The margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.



Support Vector Machine (SVM): the maximum margin linear classifier

Adopted from slides by Andrew Moore

Support Vector Machine



Adopted from slides by Andrew Moore

Computing the Margin Width

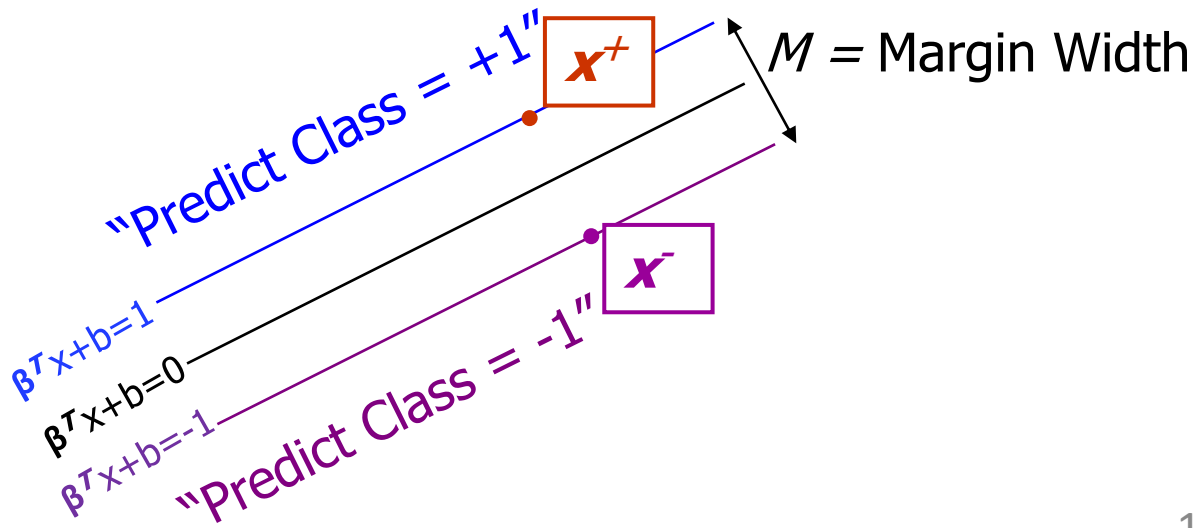
- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{\beta}^T . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{\beta}^T . \boldsymbol{x} + b = -1 \}$
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.
- We can deduce:

$$M = |\boldsymbol{x}^+ - \boldsymbol{x}^-| = |\lambda \boldsymbol{\beta}|$$

$$\begin{array}{l} \boldsymbol{\beta}^T.\boldsymbol{x}^+ + b = +1 \\ \boldsymbol{\beta}^T.\boldsymbol{x}^- + b = -1 \end{array} \longrightarrow \boldsymbol{\beta}^T.(\boldsymbol{x}^+ - \boldsymbol{x}^-)=2 \longrightarrow \boldsymbol{\beta}^T.(\lambda \boldsymbol{\beta})=2 \longrightarrow \lambda = \frac{2}{\boldsymbol{\beta}^T.\boldsymbol{\beta}}$$

$$M = |\boldsymbol{x}^+ - \boldsymbol{x}^-|$$

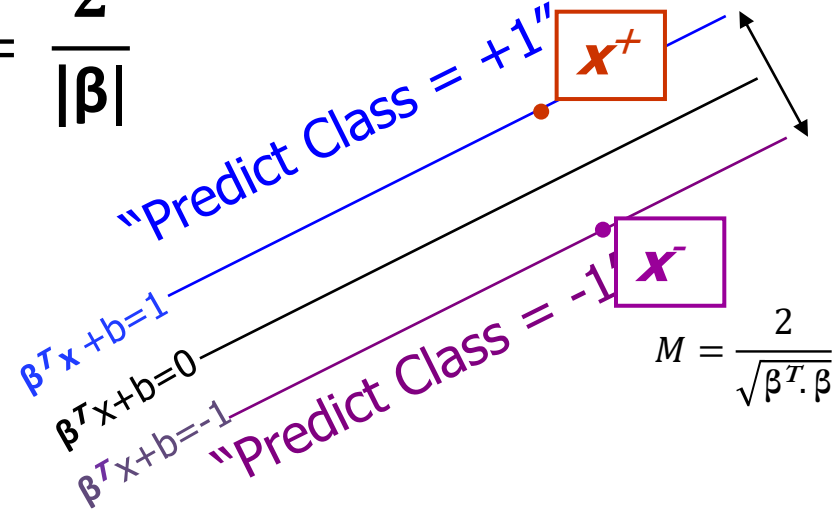
$$\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{\beta}$$



Adopted from slides by Andrew Moore

Computing the margin width

$$M = |\boldsymbol{x}^{+} - \boldsymbol{x}^{-}| = |\lambda \boldsymbol{\beta}| = \left|\frac{2}{\boldsymbol{\beta}^{T}.\boldsymbol{\beta}}\boldsymbol{\beta}\right| = \frac{2}{|\boldsymbol{\beta}|}$$



Given **β** and *b* (a boundary), we can:

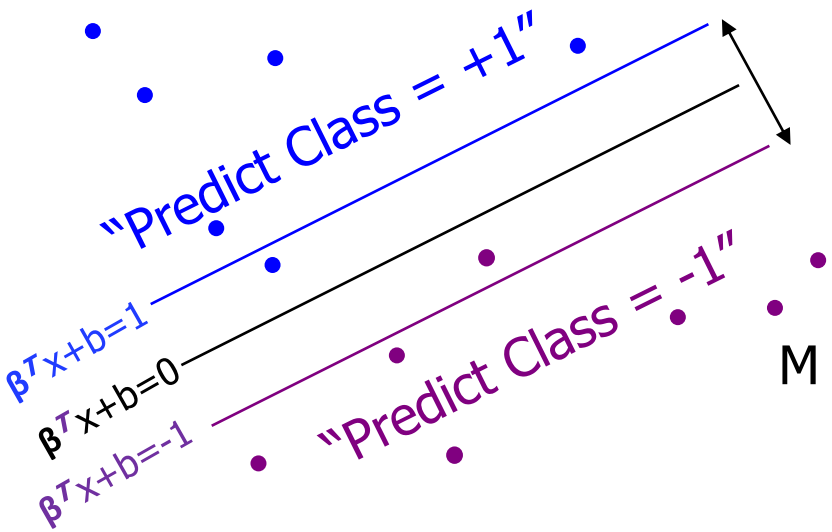
- Classify all data points (only one boundary is optimal)
- Compute the width of the margin

Now we need to write an optimization problem to search for the optimal **β** and *b* with the widest margin that classifies all the training data points correctly

Minimize $|\boldsymbol{\beta}|$ Subject to $\boldsymbol{\beta}^{T}.\boldsymbol{x} + b > 1$; when in class +1
$\boldsymbol{\beta}^{T}.\boldsymbol{x} + b < -1$; when in class -1

Adopted from slides by Andrew Moore

Learning the Maximum Margin Classifier



Minimize $\boldsymbol{\beta}^T.\boldsymbol{\beta}$ Subject to

$(\boldsymbol{\beta}^T.\ \mathbf{x}_k + b)\ y_k >= 1$ for all $(\mathbf{x}_k, y_k)$

$$M = \frac{2}{|\beta|}$$

Incorporating soft constraints

Min $\frac{1}{2}\beta^T.\beta + C\sum_{k=1}^{R}\varepsilon_k$ s.t $(\boldsymbol{\beta}^T.\ \mathbf{x}_k + b)\ y_k >= 1-\varepsilon_k$ for all $(\mathbf{x}_k, y_k)$

$0 <= \varepsilon_k <=1$

Solving the above problem using quadratic programming is a straightforward task

Adopted from slides by Andrew Moore

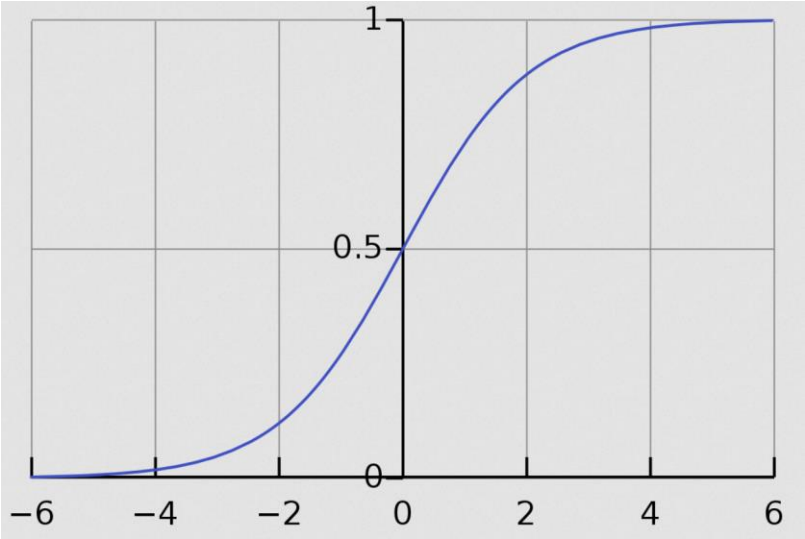
Logistic Regression

- Maximum Likelihood Estimation:

$$\hat{Y} = \arg\max P(Y|X)$$

- We assume that the likelihood function is a logistic function of a linear relationship

$$P(Y=1|X) = \frac{1}{1+\bar{e}^{\beta^T X}}$$



Logistic Regression

- Log-likelihood Inference:

Analogous to using Entropy loss

$$\hat{\beta} = \arg\max_{\beta} \log(\Pi_i P(Y_i|X_i)) =$$
$$\arg\max_{\beta} \Sigma_i \log(P(Y_i|X_i)) =$$
$$\arg\max_{\beta} \Sigma_i Y_i \log(P(Y_i = 1|X_i)) + (1 - Y_i)\log(P(Y_i = 0|X_i)) =$$
$$\arg\max_{\beta} \Sigma_i Y_i(-\log(1+e^{\beta^T X_i})+(1-Y_i)(-\beta^T X_i)-\log(1+e^{\beta^T X_i}) =$$
$$\arg\min_{\beta} \Sigma_i (1 - Y_i)\beta^T X_i + \log(1 + e^{\beta^T X_i})$$

- There is no closed form solution:

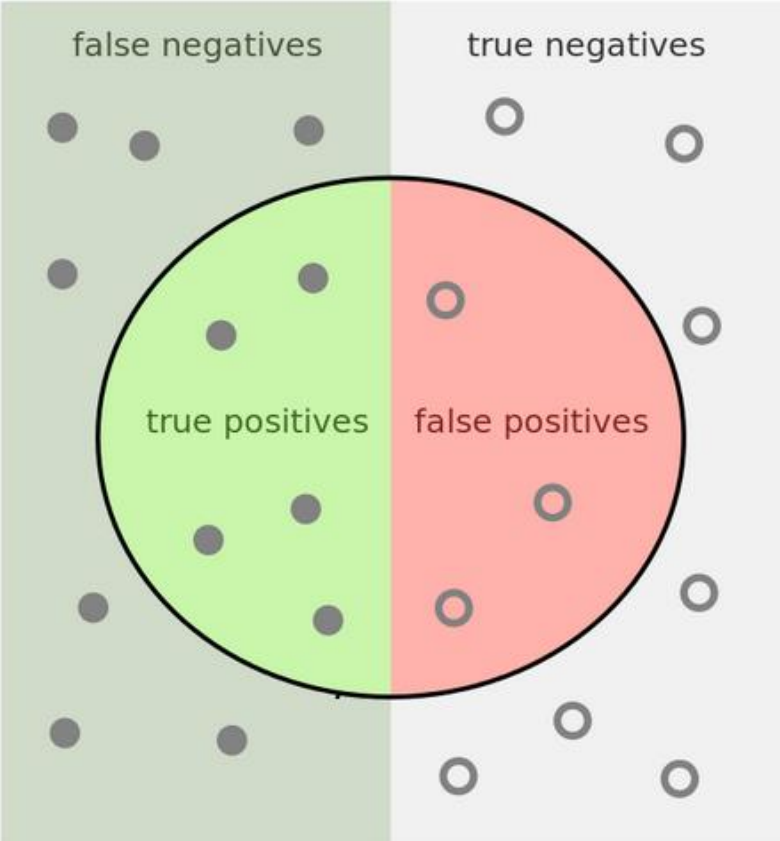
- We can use numerical optimization, e.g., Newton method
- We can approximate the logit term

Model Evaluation Process

- We use a training dataset for model selection
- A **good** parametric model along with a **suitable** training algorithm guarantees training a model that works well on the training data
- We need to validate that trained models **generalize** well on unseen data instances
- We need a second testing dataset which is fully independent of the training dataset
- We randomly split the annotated dataset into testing and training splits (sometimes, a validation set is generated as well)

Evaluation Metrics

- **Accuracy**: proportion of correctly classified items
- Accuracy can be dominated by **true negatives** (items correctly classified as not in a class).
- Sensitive with respect to imbalance
- Precision: $\frac{\text{True Positives}}{\text{True Positives+False Positive}}$
- Also called positive predictive value
- Recall: $\frac{\text{True Positives}}{\text{True Positives+False negative}}$
- Also called sensitivity
- Precision and recall are not useful metrics when used in isolation?
- We want our model to have good performance with respect to both metrics
- Implemented in sklearn



Evaluation Metrics

- Why having one measure is helpful?
- $\text{F1} = \dfrac{2\ \text{Precision Recall}}{\text{Precision} + \text{Recall}}$
- F1 is biased towards the lower of precision and recall:

- harmonic mean < geometric mean < arithmetic mean
- F1=0 when Precision=0 or Recall=0

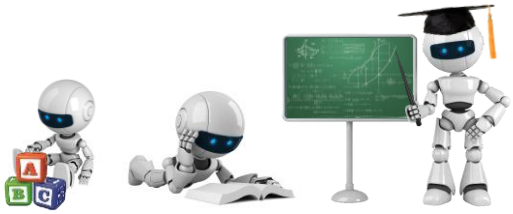
- Generalized F score:

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

USC Viterbi School of Engineering
Department of Computer Science

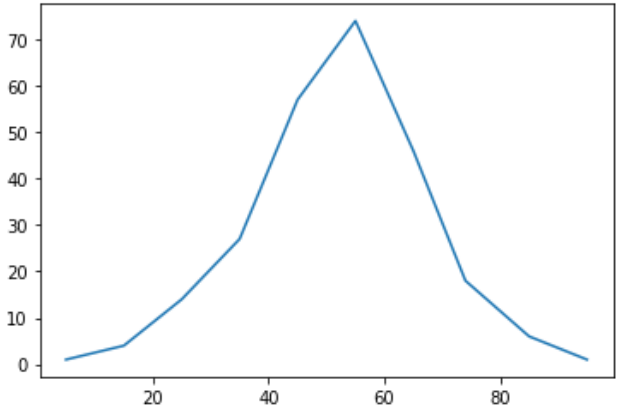
CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- Quizzes:
- Quiz Period: 2:05-2:20 and still 10 minutes
- Quiz Goal and Difficulty
- We will have Quiz 2 next Tuesday



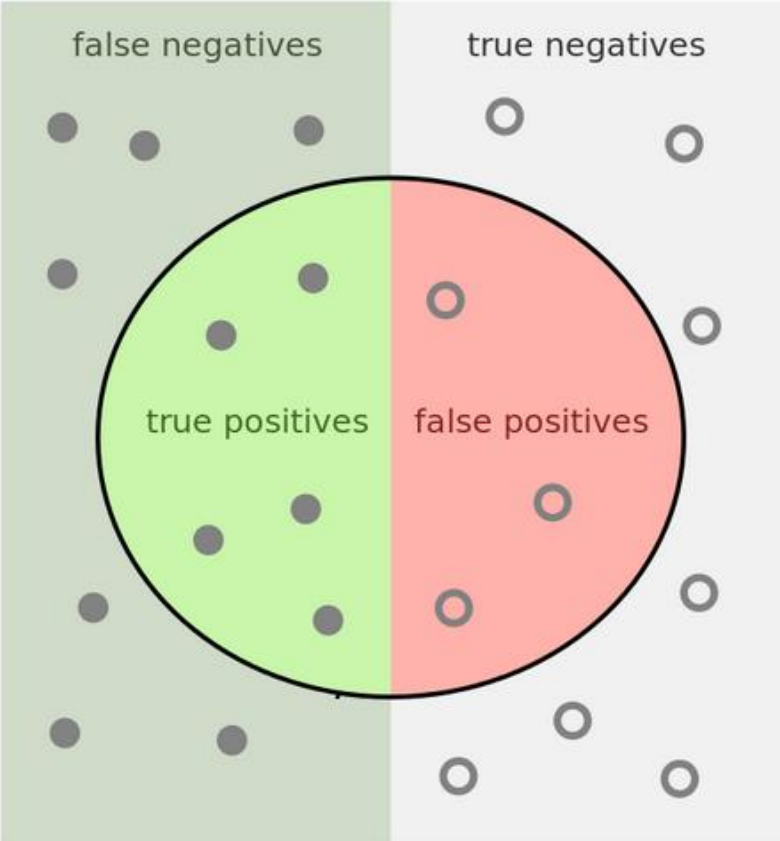
- Project:
- 42 Groups < 52 groups
- Groups of 6?
- HW1:
- Please check the homework description for edits done based on feedback from the Slack channel and prepare your report accordingly

Model Evaluation Process

- We use a training dataset for model selection
- A **good** parametric model along with a **suitable** training algorithm guarantees training a model that works well on the training data
- We need to validate that trained models **generalize** well on unseen data instances
- We need a second testing dataset which is fully independent of the training dataset
- We randomly split the annotated dataset into testing and training splits (sometimes, a validation set is generated as well)

Evaluation Metrics

- **Accuracy**: proportion of correctly classified items
- Accuracy can be dominated by **true negatives** (items correctly classified as not in a class).
- Sensitive with respect to imbalance
- Precision: $\frac{\text{True Positives}}{\text{True Positives+False Positive}}$
- Also called positive predictive value
- Recall: $\frac{\text{True Positives}}{\text{True Positives+False negative}}$
- Also called sensitivity
- Precision and recall are not useful metrics when used in isolation?
- We want our model to have good performance with respect to both metrics
- Implemented in sklearn



Evaluation Metrics

- Why having one measure is helpful? Optimization!

$$\text{F1} = \frac{2\ \text{Precision Recall}}{\text{Precision} + \text{Recall}} = \frac{2}{1/\text{Precision} + 1/\text{Recall}}$$

- F1 is biased towards the lower of precision and recall:

- harmonic mean < geometric mean < arithmetic mean
- F1=0 when Precision=0 or Recall=0

- Generalized F score:

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

- Multiclass case?

Natural Language Representation

- Language processing hierarchy levels:
- **Documents**
- Sentences
- Phrase
- Words
- Sparsity in the NLP training datasets: natural language has a very huge space:
- Ex: Average Wikipedia page size is 580 words and English has ~1M words, yet the actual number of possibilities is far more.
- We need **interpretable** representations or **embeddings** to represent natural language data for model training
- One-hot representation: two large (15M words) and meaningless

  Hotel: [0,0,0,0,1,0,0,0,0,0,0,…,0,0,0]

  Motel:[0,0,0,0,0,0,0,0,0,1,0,…,0,0,0]

Similarity of Vectors

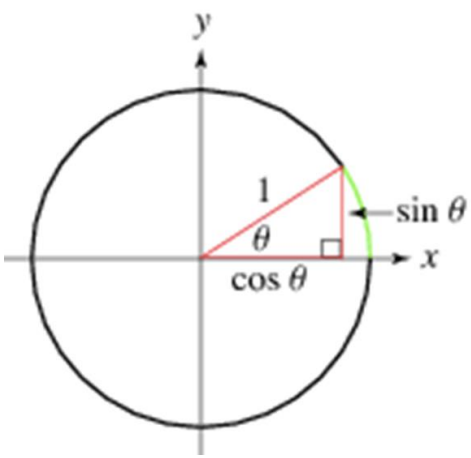
- Euclidean distance, i.e., geometric closeness :

- Curse of dimensionality

- Dot product:

$a \bullet b = ||a||||b||\cos(\theta_{ab})$

$= a_1b_1+a_2b_2+ \ldots +a_nb_n$



- Cosine similarity (scale invariant)

$\cos \theta_{ab} = a \bullet b \ / \ ||a||||b||$ -> $1- \cos \theta_{ab}$ is a metric

- Invariant with respect to the vector starting point

- EX: Hotel: [0,0,0,0,1,0,0,0,0,0,0,…,0,0,0], Motel:[0,0,0,0,0,0,0,0,0,1,0,…,0,0,0]

  Hotel'*Motel = 0

The Distributional Hypothesis

- Words that occur in the **same contexts** tend to have similar meanings (Zellig Harris, 1954)
- Example: nice, good
- Word relatedness association (Budanitsky and Hirst, 2006): related words **co-occur** in different contexts
- Example: cup, coffee
- If semantic similarity and association of words can be encoded into their representations, we may be able to address the challenge of sparsity
- In the absence of a particular word during training, we can rely on its synonyms that exist in the training dataset: Motel vs Hotel
- We can draw conclusions:

Lecturers teach in the university-> Professors ___ in the university.

Vector Embedding of Words

- Represent words using dense vectors:
  - Latent Semantic Analysis/Indexing (SC Deerwester et al, 1988)
    - Term weighting-based model
    - Consider occurrences of terms at **document level**.
  - Word2Vec (Mikolov et al, 2013)
    - Prediction-based model.
    - Consider occurrences of terms at **context level**.
  - GloVe (Pennington et al, 2014)
    - Count-based model.
    - Consider occurrences of terms at **context level.**

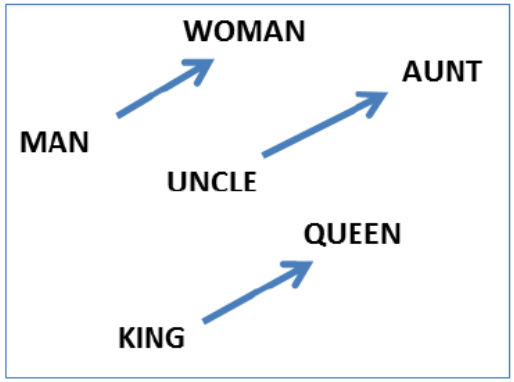
Word Embedding

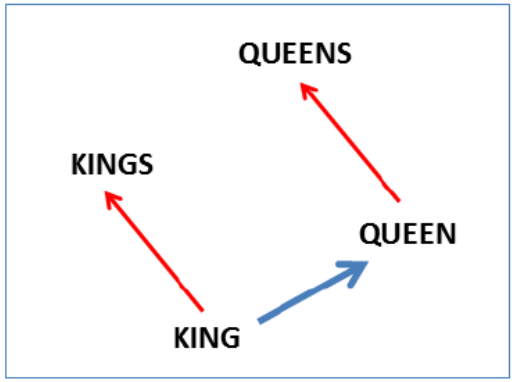
- Each word is represented by a vector:

- The same size is used for all words
- Relatively low dimensional (~300)
- Vectors for similar words are similar (measured in dot product)
- Vector operations can be used for

semantic and syntactic deductions, e.g.,

Queen – Woman + Man = King





- The key idea is to derive the embeddings from the distributions of word context as they appear in a large corpus.

Matrix Factorization

- We can form a matrix of M using the idea of Bag of Words: the word representations are highly sparse

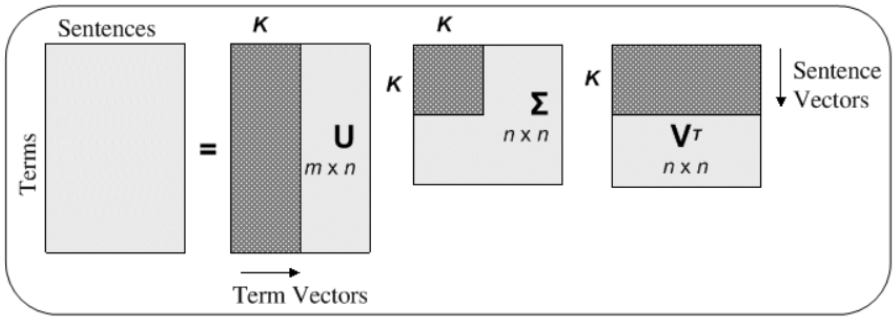
Words

Contexts

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

- Singular value decomposition (U, V are orthonormal)

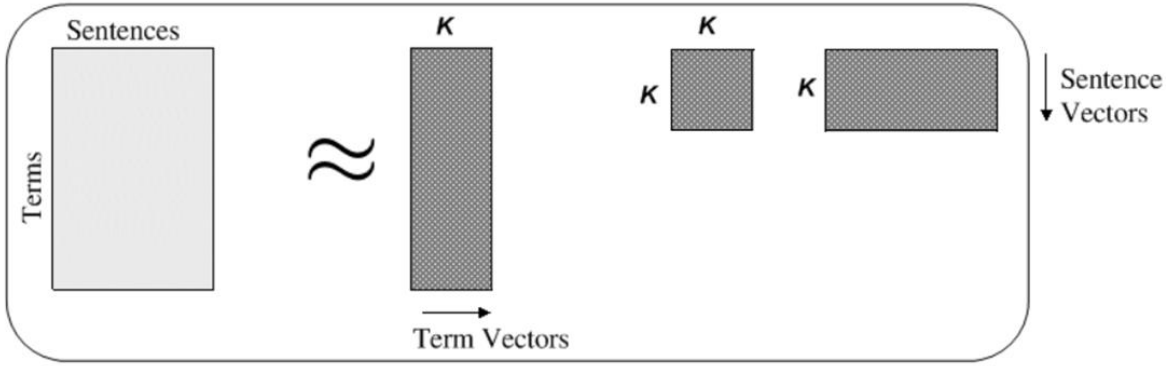
$$M_{m\times n} = U_{m\times n}\Sigma_{n\times n}V^T_{n\times n}$$



Matrix Factorization

- Many singular values are going to be zero or negligible

$$M_{m\times n} \approx U'_{m\times k}\Sigma'_{k\times k}V'^{T}_{k\times n}$$



$$m_l = u_l \Sigma' V'^{T}$$
$$u_l = m_l V' \Sigma'^{-1}$$

- We can use rows of U as word embeddings
- An old idea for dimensionality reduction (it is possible to use other matrix factorization methods, e.g., non-negative matrix factorization)
- Determining context is heuristic
- computational expensive with O(mn^2) cost for an n*m matrix
- Hard to incorporate new words

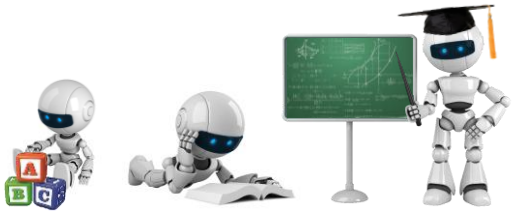
Word2Vec

- Core idea: find embeddings using a prediction task involving **neighboring words** in a huge real-world corpus.
- Input data can be sets of **successive word-patterns** from meaningful sentences in the corpus, e.g., "one of the most important".
- Try to build **synthetic** prediction tasks using these patterns, e.g., "(one of __ most important, the)"
- Train a model to solve the prediction task
- Embeddings are found as a **byproduct** of this process
- More specifically:
- **We consider a window with the center word w_t and "context words" $w_{t'}$ with a window fixed size, e.g., (t'=t-5, … t-1, t+1, … , t+5).**
- The model is assumed to be a two-layer neural network
- We train the network to predict all w_t given $w_{t'}$ such that $p(w_t|w_{t'})$ is maximized
- We learn embeddings such that the prediction loss is minimized, i.e., if two words occur in close proximity, their representations become similar.

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- Project Teams: ~50 groups are formed

- Topics & written assignment

- Quiz 3
- HW1 deadline:

- Zipped folder: name-family

- HW2: 3 weeks

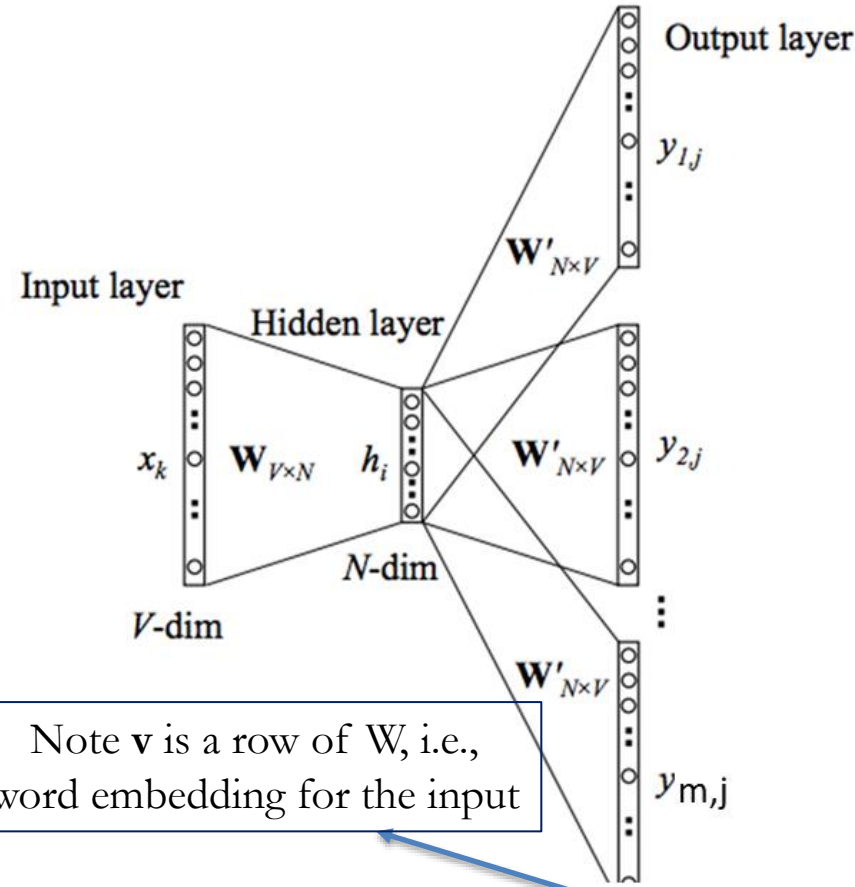
Word2Vec

- Core idea: find embeddings using a prediction task involving **neighboring words** in a huge real-world corpus.
  - Input data can be sets of **successive word-patterns** from meaningful sentences in the corpus, e.g., "one of the most important".
  - Try to build **synthetic** prediction tasks using these patterns, e.g., "(one of __ most important, the)"
  - Train a model to solve the prediction task
  - Embeddings are found as a **byproduct** of this process
- More specifically:
  - **We consider a window with the center word w_t and "context words" $w_{t'}$ with a window fixed size, e.g., (t'=t-5, … t-1, t+1, … , t+5).**
  - The model is assumed to be a two-layer neural network
  - We train the network to predict all w_t given $w_{t'}$ such that $p(w_t|w_{t'})$ is maximized
  - We learn embeddings such that the prediction loss is minimized, i.e., if two words occur in close proximity, their representations become similar.

Skip-Gram

- Given a center word, we predict the context words:

- Vocabulary size: V
- Input layer: center word in 1-hot form.
- The row k in W_{VxN} is the vector embedding of k-th center word.
- The column k of W'_{NXV} is context vector of the k-th word.
- At output layer y_{ij}, i=1..M is computed:
  1. We use the context word 1-hot vector to choose its column in W'_{NxV}
  2. dot product with h_i for the center word
  3. compute the softmax
  4. **Match** the output one-hot vector
- After optimization, we will have two vectors for each word. We can set the eventual embedding to e the average of these two vectors



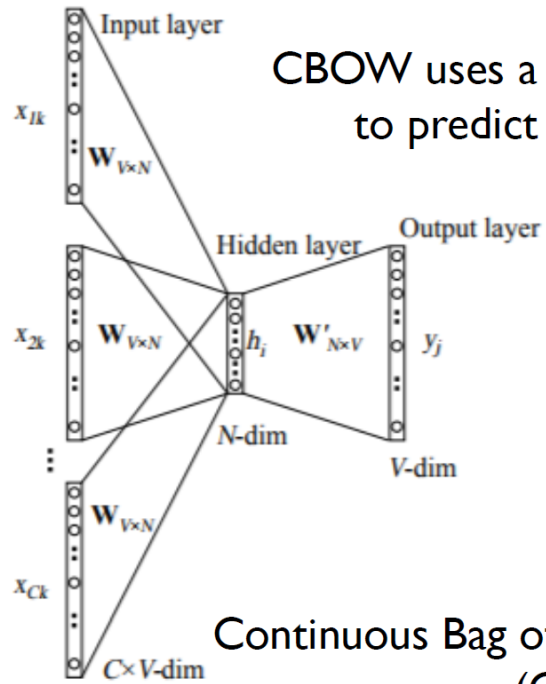
Note **v** is a row of W, i.e., word embedding for the input

$$h_{N\times 1} = W^T_{V\times N} x_{V\times 1} = v$$
$$\hat{z}_j = h^T_{N\times 1} W'[:, j]_{N\times 1} = v^T u$$
$$\hat{y} = \sigma(W'^T_{N\times V} h_{N\times 1})$$

Note **u** is a row of W', i.e., word embedding for the output

Continuous Bag of Words

- Given a context word, we predict the center word:
- Vocabulary size: V
- Input layer: context words in 1-hot form.
- The row k in W_{VxN} is the vector embedding of k-th context word.
- The column k of W'_{NXV} is vector of the k-th center word.
- The output column y_{ij}, i=1..M is computed:
  1. We use the center word 1-hot vector to choose its column in W'_{NxV}
  2. dot product with h_i for the contex word
  3. compute the softmax
- We can set the eventual embedding to the average of these two vectors
- Skip-gram incorporates non-frequent words better than CBOW



CBOW uses a window of word to predict the middle word

Continuous Bag of words (CBOW)

- Ex: Today is really __ day

Skip-gram: delightful+context and nice+context

Word2Vec Optimization Problem

- Consider an arbitrary order on vocabulary and let v_t be the word vector for center word t and u_t be the word vector for the context word t:

Ex: [...like to **eat** lunch and...]

- We generate binary tasks: (eat, like), (eat, to),...

- We solve for the word vector by maximizing the following likelihood function

$$v_t, u_t = arg\max \log(\Pi_{t=1}^{T} P(c|w_t)) =$$

$$arg\max \log(\Pi_{t=1}^{T} \Pi_{j=-M}^{M} P(w_{t-j}|w_t)) =$$

$$arg\max \frac{1}{T} \sum_{t=1}^{T} \sum_{j=-M, j\neq 0}^{M} \log(P(w_{t-j}|w_t)) =$$

Important Step

Word2Vec Optimization Problem

- Conditional Probability modeling

Note **v** denotes word embeddings for center words

$$P(w_{t-j}|w_t) = \frac{e^{u_{t-j}^T v_t}}{\sum_{t=1}^{T} e^{u_{t-j}^T v_t}}$$

Is it an extension of the logistic function?

Computationally expensive!

Note **u** denotes word embeddings for the context words

- The log-likelihood optimization problem: one sum!

$$u_o, v_c = \arg\min \frac{1}{V} \sum_{w=1}^{V} -u_o^T v_c + \log(\sum_{w'=1}^{V} e^{u_o^T v_c})$$

- Can be solved similar to logistic regression objective using numerical optimization techniques, e.g., gradient descent

Word2Vec Optimization Problem

- gradient descent step

$$u_o, v_c = \arg\min \frac{1}{V} \sum_{w=1}^{V} -u_o^T v_c + \log(\sum_{w'=1}^{V} e^{u_o^T v_c}) =$$

$$v_c^{i+1} = v_c^i - \eta \nabla f(v_c^i) =$$

$$\nabla f(v_c^i) = -u_o + \sum_{w=1}^{v} \frac{e^{u_o^T v_c}}{\sum_{w'=1}^{V} e^{u_o^T v_c}} u_o =$$

$$\nabla f(v_c^i) = -u_o + \sum_{w=1}^{v} p(u_o|v_c) u_o = -u_o + E(u_o|v_c)$$

- Tutorial:

https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial

Negative Sampling

- Word2Vec optimization is a highly computationally intensive problem: the **shallow** network has a large number of weights and we will have billions of pairs
- Because we use one-hot vectors, each training pair (c,o) contributes minimally to updating the weights
- Negative sampling: for each positive pair, we randomly generate negative pairs, for which the network output should be 0.

$$u_o, v_c = \arg\min \sigma(u_o^T v_c) + \sum_{k=1}^{K} E_{v_{w_i} \sim P(w_c)} \log(-\sigma(v_{w_i}^T v_c))$$

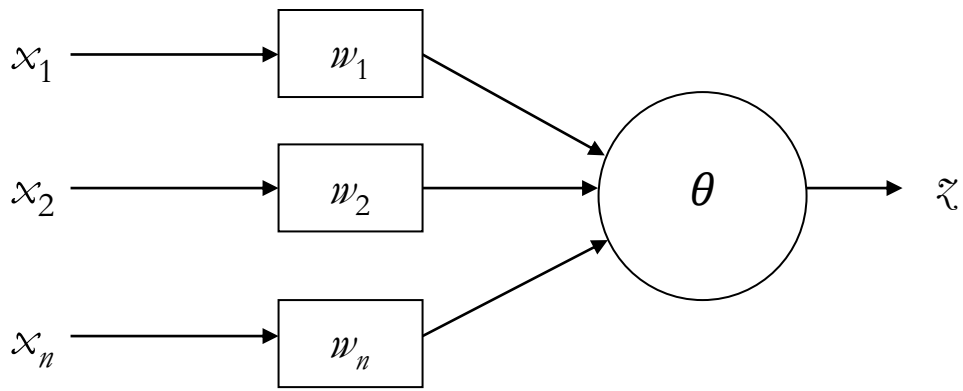
- This is an instance of naïve data augmentation

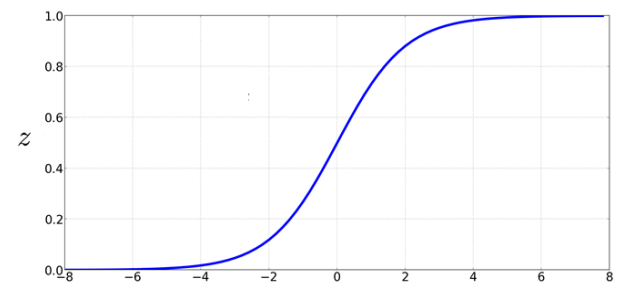
Neural vs Factorization-Based Embeddings

- Comparison is challenging (Levy and Goldberg, NeurIPS 2014):
  - **Hyperparamters**
  - Factorization algorithm
  - Amount of data
- A particular word embedding approach is unlikely to be state-of-the-art for all applications
- Hyperparameters appear to have the largest impact in performance.
- Neural models are less sensitive with respect to hyperparameters, and training data preparation is more straightforward and systematic

Neural Networks

- Perceptron: the neural network learning model in the 1960's
- Simple and limited (single layer models)
- Basic concepts are similar for multi-layer models





$$z = 1/(1 + \exp(-x))$$

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i\, w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i\, w_i < \theta \end{cases}$$

XOR Problem

- Can perceptron be used to learn XOR?

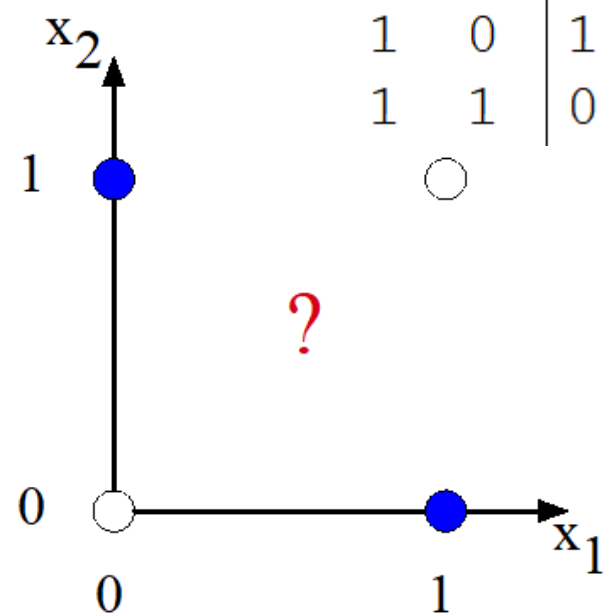
$$ax_1 + bx_2 + c = 0$$

$$\begin{aligned} a\,0 + b\,0 + c &< 0 \\ a\,0 + b\,1 + c &> 0 \\ a\,1 + b\,0 + c &> 0 \\ a\,1 + b\,1 + c &< 0 \end{aligned} \quad \rightarrow \quad \begin{aligned} c &< 0 \\ b + c &> 0 \\ a + c &> 0 \\ a + b + c &< 0 \end{aligned} \quad \rightarrow$$

$$\begin{aligned} c &< 0 \\ a + b + 2c &> 0 \\ -a - b - c &> 0 \end{aligned} \quad \rightarrow \quad \begin{aligned} c &< 0 \\ c &> 0 \end{aligned} \quad !$$

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



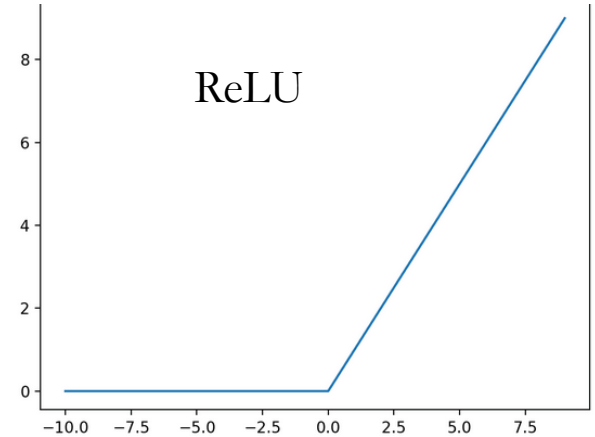
c) x_1 XOR x_2

Figures from Jurafsky and Martin Slides

XOR Problem

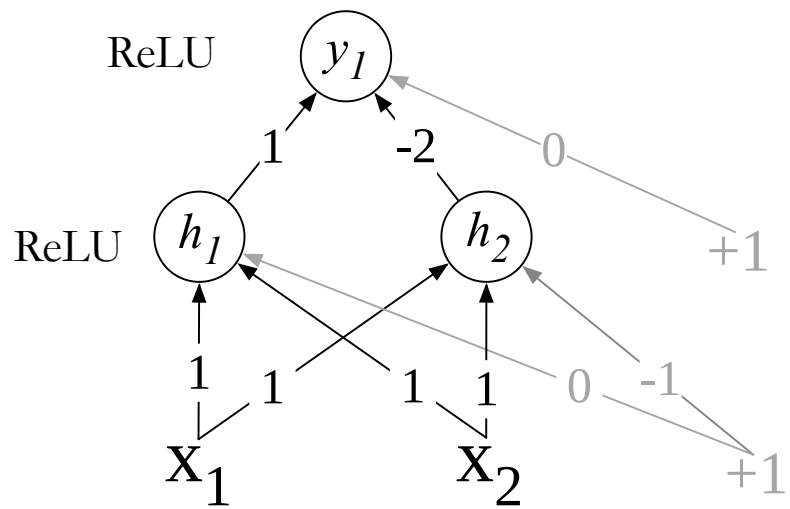
- Idea: XOR function can be computed using two layers perceptron units.

$$y_1 = Relu(Relu(x_2 + x_1) - 2Relu(x_2 + x_1 - 1))$$



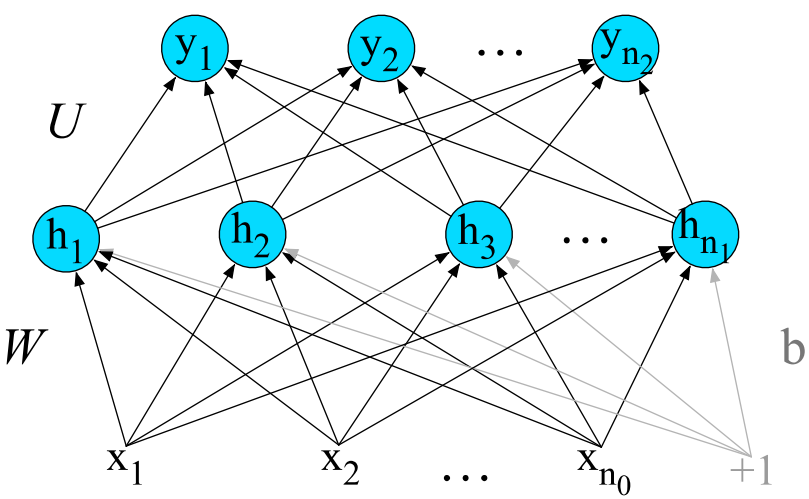
XOR

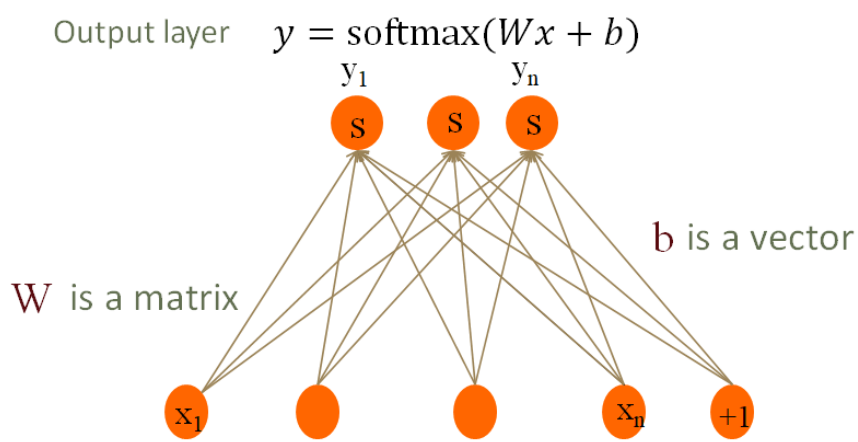
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Feedforward Neural Networks

- An extension to perceptron by forming series connection between layers of parallel perceptrons: input, output, and hidden layers
- The number of layers, the nodes at each layer, and non-linear functions are design parameters

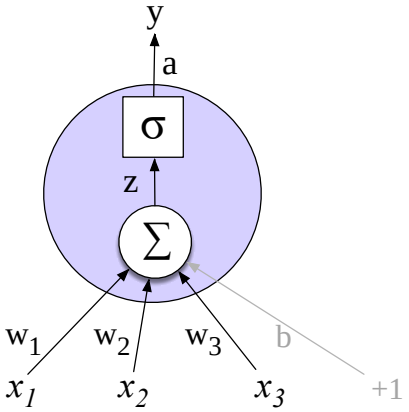




Figures adopted from slides by Jurafski and Martin

Multilayer Perceptron

- In the default setting, a node in hidden layers receives inputs from all nodes in the previous layer and its output is fed to all nodes in the next layer



$$
\begin{aligned}
z^{[1]} &= W^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= g^{[1]}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= g^{[2]}(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
$$

for i **in** 1..n

$$
\begin{aligned}
z^{[i]} &= W^{[i]}\, a^{[i-1]} + b^{[i]} \\
a^{[i]} &= g^{[i]}(z^{[i]}) \\
\hat{y} &= a^{[n]}
\end{aligned}
$$

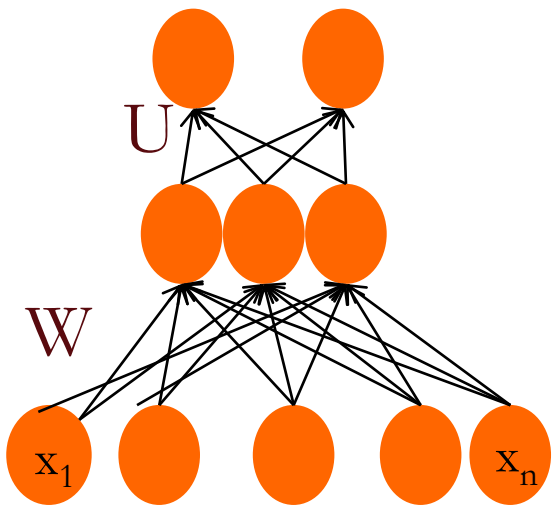
Figures adopted from slides by Jurafski and Martin

Multiclass Outputs

- For a multiclass classification problem:
  - We add one output for each class
  - We use a "softmax layer" at the output to generate a probability distribution
  - We use a proper loss function at the output

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \le i \le D$$



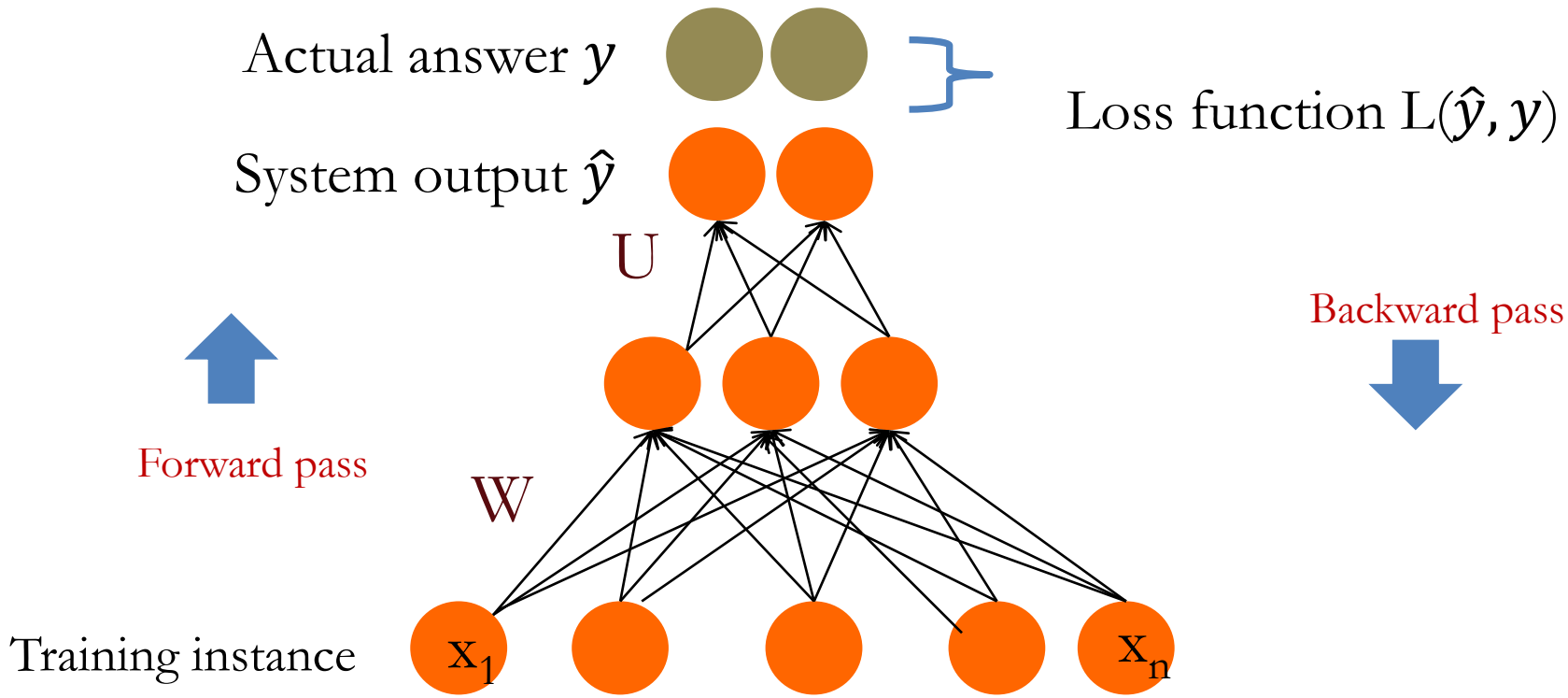
Figures adopted from slides by Jurafski and Martin

Universal Approximation Theorem

- MLPs can represent a wide range of functions given appropriate values for the weights. (George Cybenko in 1989)
- Given sufficient layers, i.e., deep nets
- Given sufficient nodes, i.e., wide nets
- Only existential result: it merely states that approximating most given functions is possible but does not provide the solution.

Training Feedforward Networks

- Backpropagation algorithm: an iterative algorithm to learn network weights using annotated data



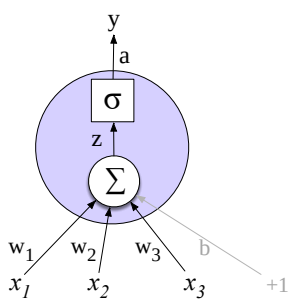
Figures adopted from slides by Jurafski and Martin

Training Feedforward Networks

- For every training data point (x, y)
  - Run *forward* computation to find model estimate $\hat{y}$
  - Run *backward* computation to update weights:
    - For every output node
  - Compute loss L between true y and the estimated $\hat{y}$
  - For every weight w from hidden layer to the output layer

Update the weight using gradient descent $\frac{d}{dw} L(f(x; w), y)$

    - For all other nodes
  - Assess how much blame it deserves for the current answer



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_i}$$

Adopted from slides by Jurafski and Martin

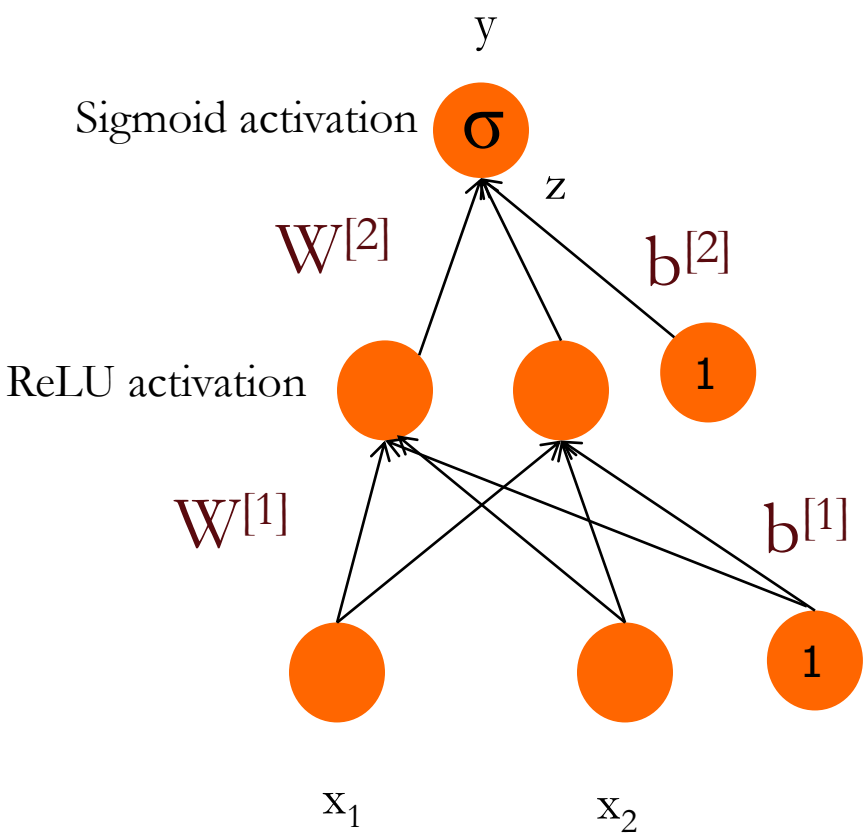
Backpropagation for Two Layer Network Output

$$
\begin{aligned}
z^{[1]} &= W^{[1]}\mathbf{x} + b^{[1]} \\
a^{[1]} &= \mathrm{ReLU}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i}$$

$$\frac{d\,\mathrm{ReLU}(z)}{dz} = \begin{cases} 0 & for \quad z < 0 \\ 1 & for \quad z \geq 0 \end{cases}$$

$$\frac{\partial y}{\partial z} = \frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$



Backpropagation for Two Layer Network Output

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i}$$

$$\begin{aligned} z^{[1]} &= W^{[1]}\mathbf{x} + b^{[1]} \\ a^{[1]} &= \mathrm{ReLU}(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} &= \sigma(z^{[2]}) \\ \hat{y} &= a^{[2]} \end{aligned}$$

$$L(\hat{y}, y) = -(\mathrm{y}\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$y = \sigma(z)$$

$$z = \sum w_i\, h_i + \mathrm{b}$$

$$\frac{\partial L}{\partial y} = -\left(\left(y\frac{\partial \log(a)}{\partial a}\right) + (1-\mathrm{y})\,\frac{\partial \log(1-a)}{\partial a}\right)$$

Notational Simplicity

$$= -\left(\left(y\frac{1}{a}\right) + (1-\mathrm{y})\,\frac{1}{1-a}(-1)\right) = -\left(\frac{y}{a} + \frac{y-1}{1-a}\right)$$

$$\frac{\partial a}{\partial z} = a(1-a)$$

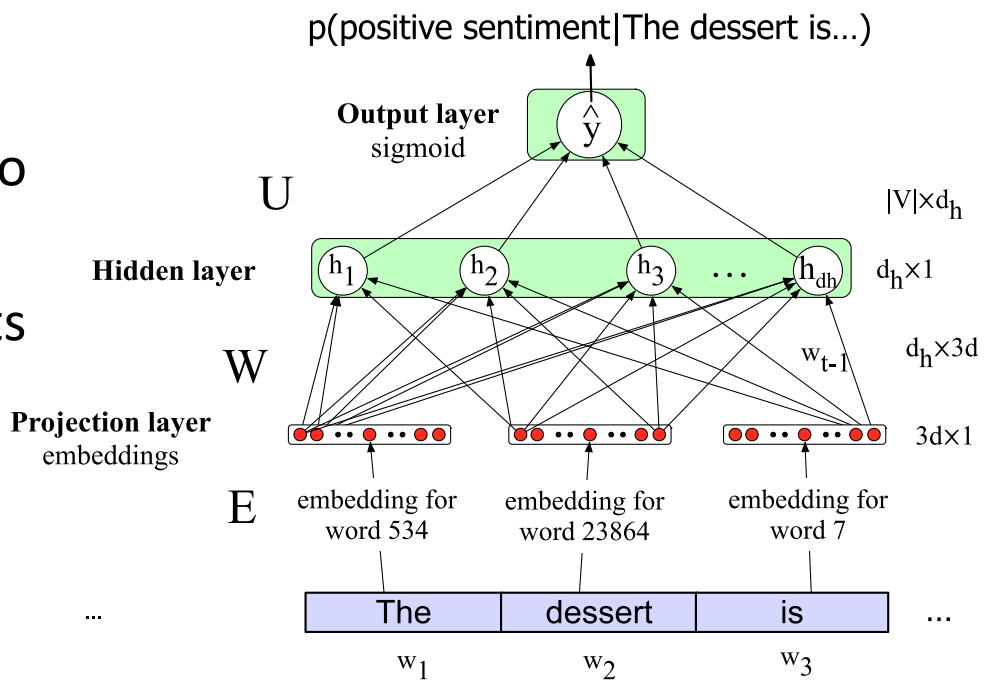
$$\frac{\partial z}{\partial w_i} = h_i$$

$$\frac{\partial L}{\partial w_i} = -\left(\frac{y}{a} + \frac{y-1}{1-a}\right)a(1-a)h_i = (\mathrm{a}-\mathrm{y})h_i$$

MLP for NLP Tasks

- Assume a fixed size length

1. Make the input the length of the longest input
   - If shorter then pad with zero embeddings
   - Truncate if you longer inputs are observed at test time
2. Create a single "sentence embedding"
   - Take the mean of all the word embeddings
   - Take the element-wise max of all the word embeddings



Figures adopted from slides by Jurafski and Martin

Neural Networks vs SVM Rivalry

- Back Propagation: 1986
- SVM: 1992
- Deep Learning: 2012

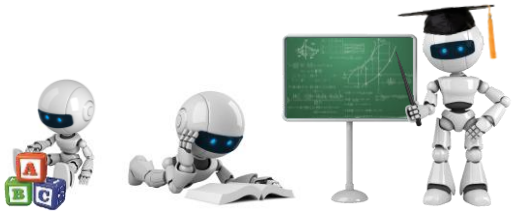
- Three key reasons for reemergence of deep learning:

- Computational power: NVIDIA CUDA (2007)
- Annotated datasets: ImageNet (2009)
- ReLU !!!

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Logistical Notes

- Project Groups: Sept 13
- HW2: computational resource
- PyTorch lecture

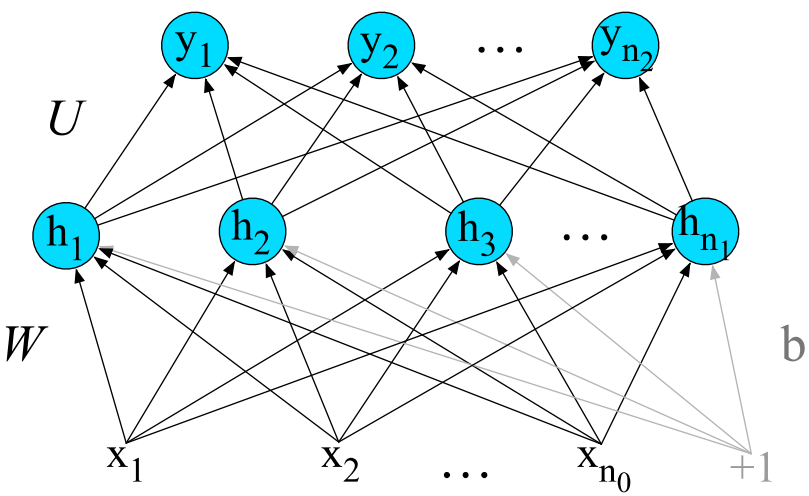
- Coursera: Deep Neural Networks with PyTorch

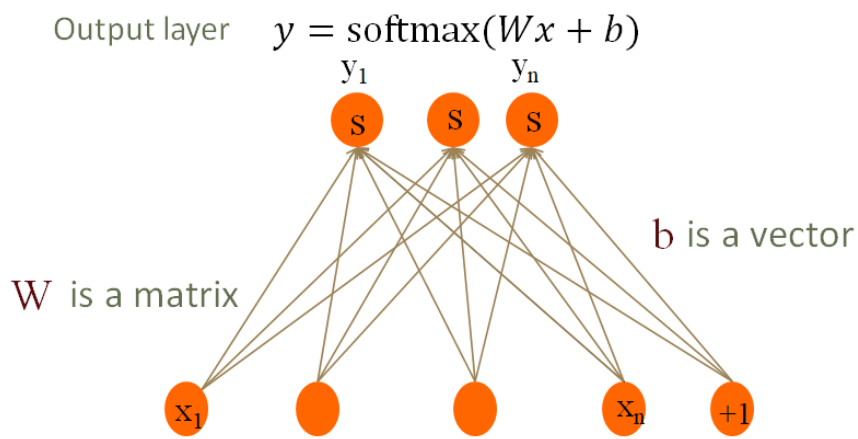
Feedforward Neural Networks

- Is a function approximator where the output depends on a single input

$$y = f(x)$$

- The inputs are assumed to be independent from each other



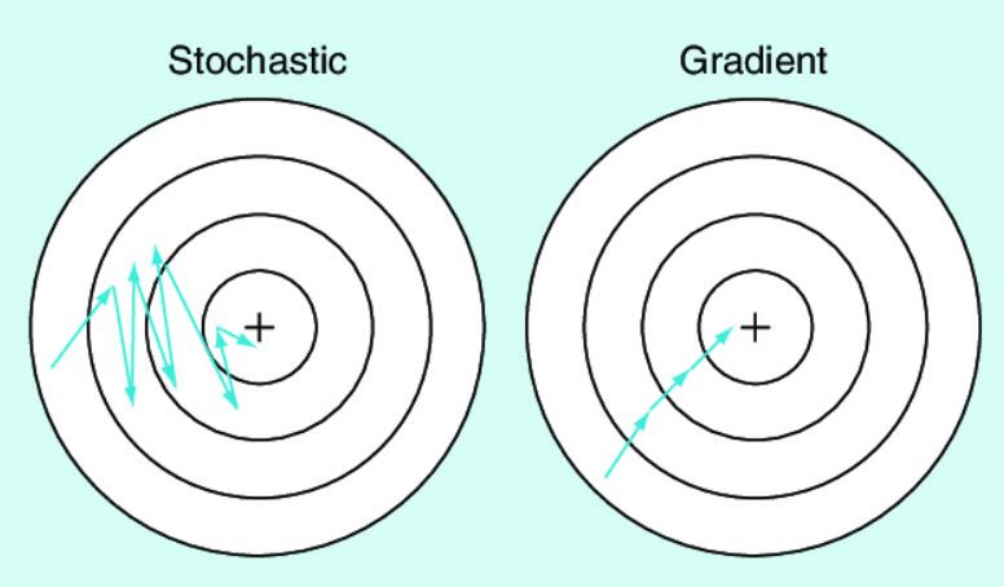


Figures adopted from slides by Jurafski and Martin

Training Feedforward Neural Networks

- Backpropagation is performed using stochastic gradient descent

- We create random batches from the training dataset
- We perform the normal gradient descent using a batch
- Epoch: optimize the cost function using all batches
- Optimize through epochs until convergence



Feedforward Neural Networks

- Limitations of feedforward neural networks
- Input size should be fixed
- All the input instances should have the same length
- Solutions for this problem are not perfect!
- Language properties:
- Contextual: “river bank” vs “bank branch”
- Long-term Dependency: I was born in the US but moved to Italy when I was 2 and grew up there. I moved back to the US when I was 18 for college, so I can speak __ and English.
- The order of words is important: “this is an informative book, but I do not like it” vs “this is not an informative book, but I like it”

Figures adopted from slides by Jurafski and Martin

Recurrent Neural Networks

- We can consider NL data as sequential data points, where the current word depends on the previous words in the sequence:

  1 2 3 4 5 6 7 8 9 10 11

- Ex: Today, I want to play football and then watch a movie.
- Learning representations by back-propagating errors, 1986
- Core Idea: the function approximator can receive the input word by word such that its output depend on the history, i.e., relying on a notion of memory

FNN $$y = f(x)$$

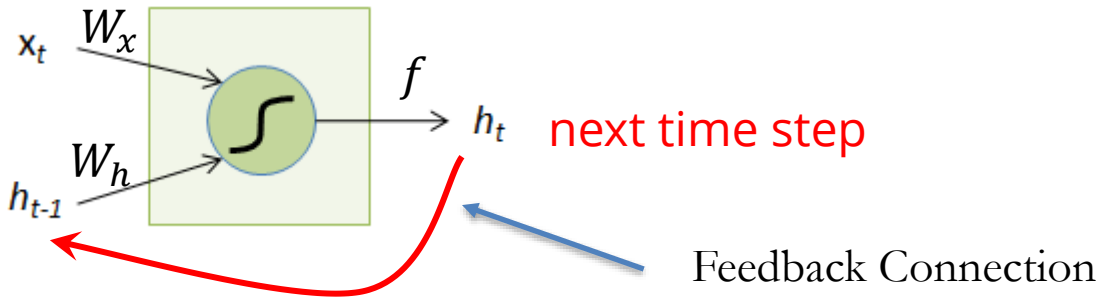
RNN $$y_t = f(x_t, h_{t-1})$$

output input memory

Recurrent Neural Networks

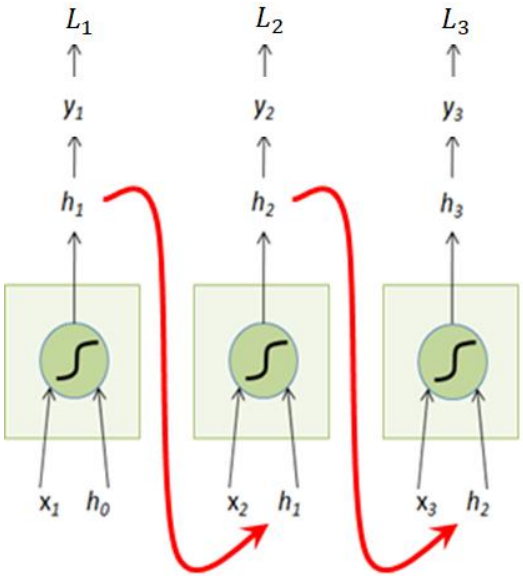
- Equipping perceptron with memory

- x_t: Input at time t
- h_{t-1}: State at time t-1



$$h_t = f(W_x x_t + W_h h_{t-1}), W_x \in \mathbb{R}^{M*N}, W_h \in \mathbb{R}^{H*H}$$

- Unfolding RNN
- We can make the unit multi-layer



Figures adopted from slides by Arun Mallya

Recurrent Neural Networks

- The weight matrices are shared across time

- multi-output

$$L = L_1 + L_2 + L_3$$

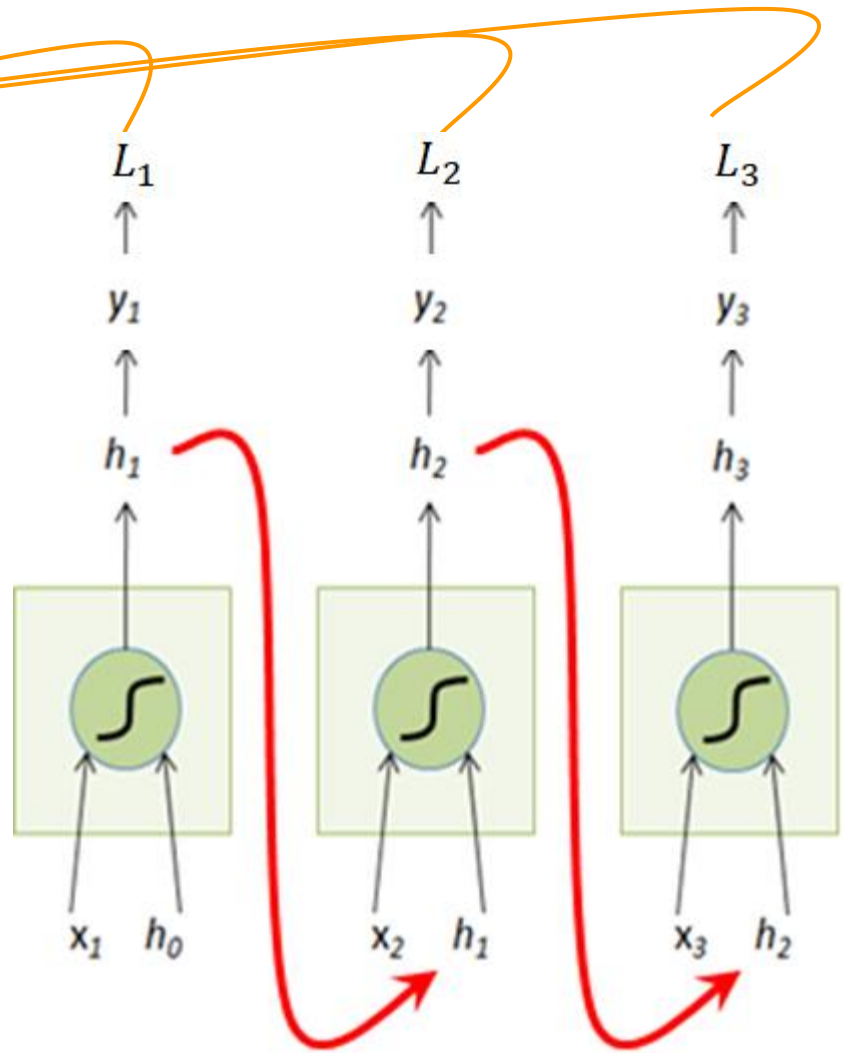
$$L = \sum_{t=1}^{T} L_t$$

$$y_t = f(x_t, h_{t-1})$$

$$L_t = l(y, \hat{y}_t)$$

- Single Output

$$L = L_3$$



Figures adopted from slides by Arun Mallya

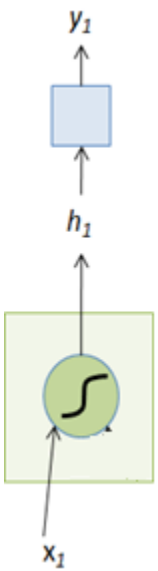
Training Feedforward Networks

- For every training data point (x, y)
  - Run *forward* computation to find model estimate $\hat{y}$
  - Run *backward* computation to update weights:
    - For every output node
  - Compute loss L between true y and the estimated $\hat{y}$
  - For every weight w from hidden layer to the output layer

Update the weight using gradient descent $\frac{d}{dw} L(f(x; w), y)$

- For all other nodes
  - Assess how much blame it deserves for the current answer

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial h} \frac{\partial h}{\partial W} \quad , \quad L = L(\mathrm{y}(\mathrm{h}(\mathrm{W})))$$



Figures adopted from slides by Arun Mallya

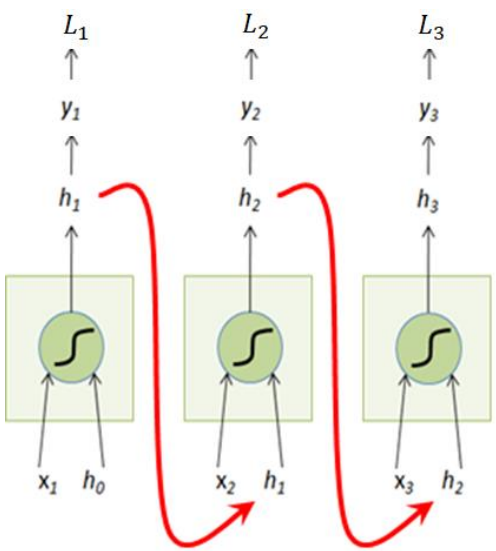
Training RNNs

- For every training data point (x, y)
  - Run *forward* computation to find model estimates $\widehat{y_t}$
  - Run *backward* computation to update weights:
    - For every output node
  - Compute loss L between true y and the estimated $\widehat{y_t}$
  - For every weight w from hidden layer to the output layer

Update the weight using gradient descent $\frac{d}{dw}\sum_{t=1}^{T} L_t$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial W} \quad ???$$

$$L_2 = L_2(y_2(h_2(W, h_1(W))))$$

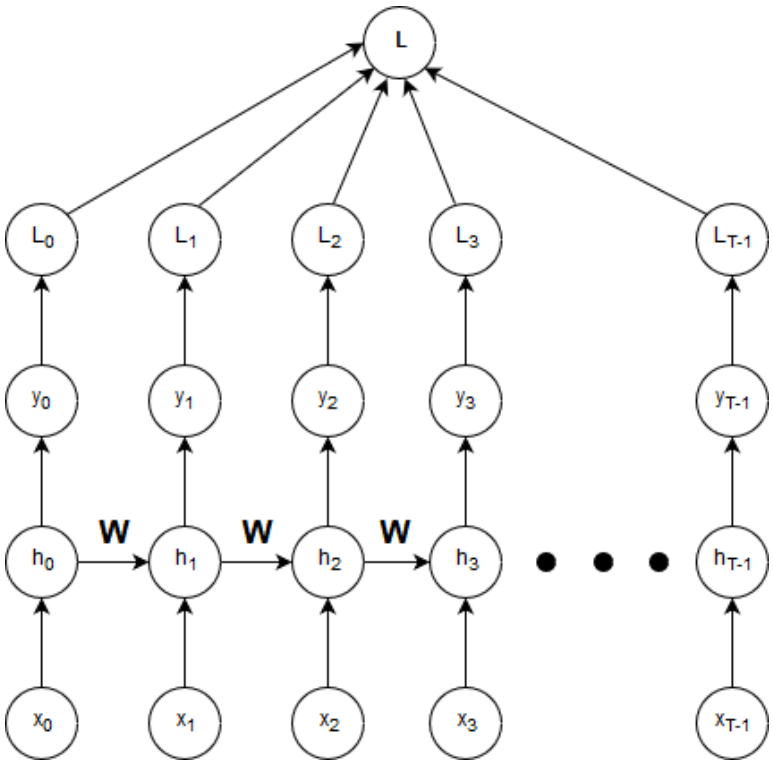


Figures adopted from slides by Arun Mallya

Backpropagation Through Time

- Gradient descent step to update the weights:

$$\mathbf{W} \rightarrow \mathbf{W} - \alpha \frac{\partial L}{\partial \mathbf{W}}$$



- Issue: **W** occurs each timestep
- **Every** path from **W** to L is one dependency for differentiation
- We need to find all paths from **W** to L
- There is one depenency through L_1
- There are two depenencies through L_2

Figures adopted from slides by Arun Mallya

Backpropagation Through Time

$$L = \sum_{t=1}^{T} L_t$$

2D Chain Rule

$$\frac{\partial L_t}{\partial W} = \sum_{k=1}^{t} \boxed{\frac{\partial L_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$$

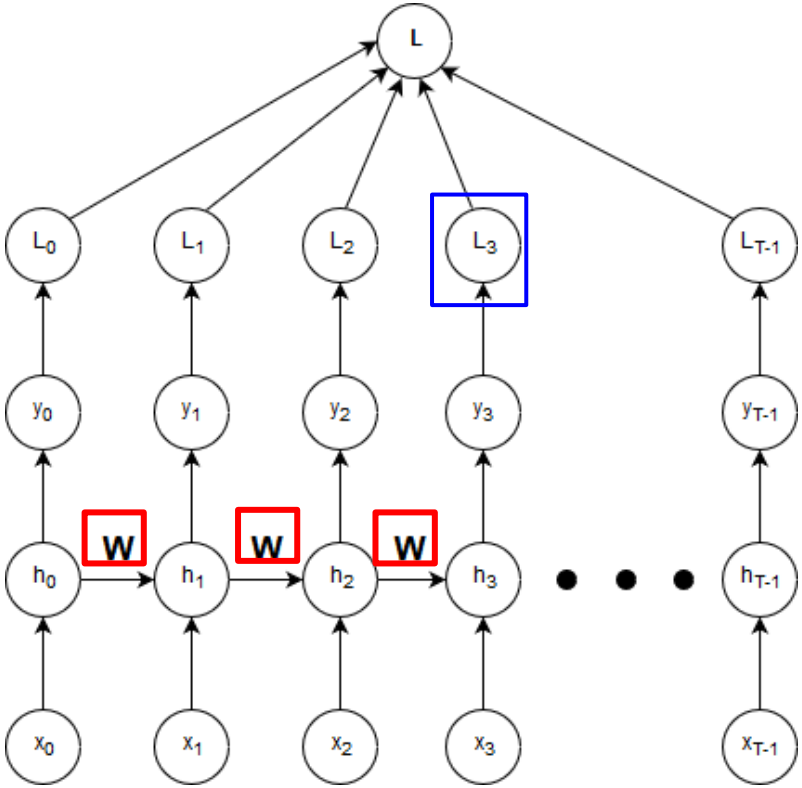
1D Chain Rule

$$\frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k}$$

$$\frac{\partial L_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

1D Chain Rule

$$\frac{\partial L_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \prod_{m=k+1}^{t} \frac{\partial h_m}{\partial h_{m-1}} \frac{\partial h_k}{\partial W}$$



Backpropagation Through Time

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \prod_{m=k+1}^{t} \frac{\partial h_m}{\partial h_{m-1}} \frac{\partial h_k}{\partial W}$$

- Computationally expensive
- Vanishing/Exploding gradients challenge

- Truncated Backpropagation

- Tutorial

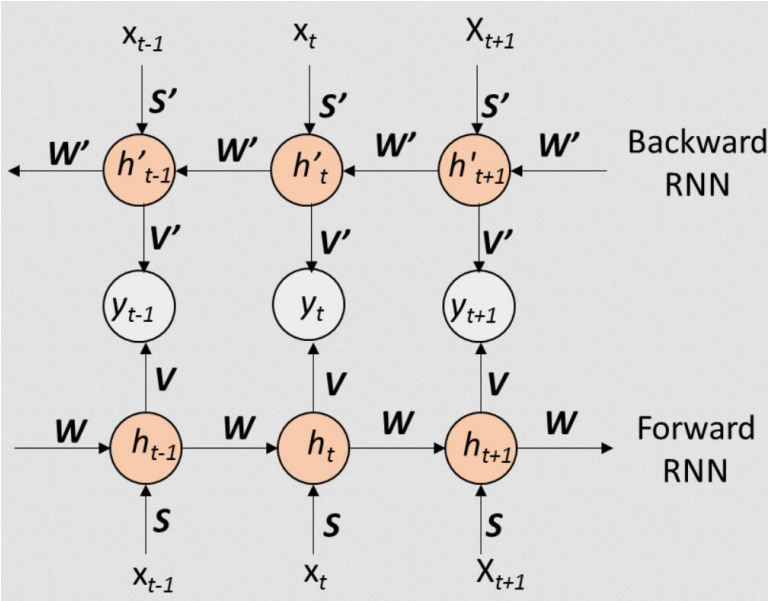
https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html

Bidirectional RNN

- RNN is developed for temporal data (unidirectional)
- Effects in NL are bidirectional:

- Ex: I lived in __ for ten years so I can speak French

- Solution: we can use two RNNs that process the input in opposite directions

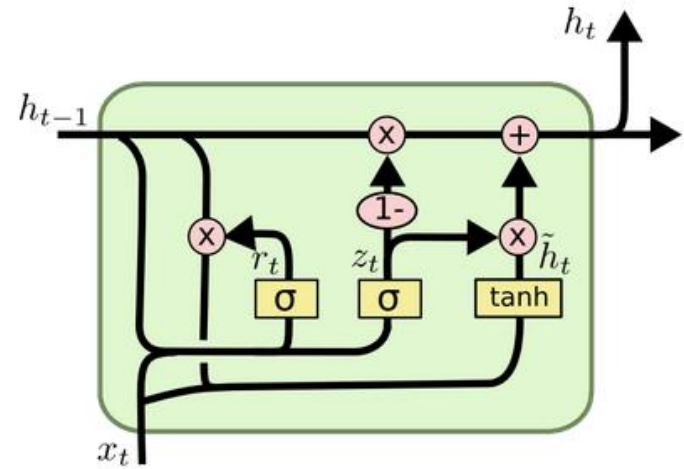


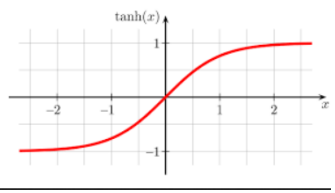
$$h_t = f(W_x x_t + W_h h_{t-1})$$
$$h'_{t'} = f(W'_x x_{t'} + W'_h h'_{t'-1})$$
$$t' = T - t$$
$$y_t = g(h_t, h'_{t'})$$

Gated RNN

- Gated recurrent unit: can learn long-range dependencies

- Control mechanism on information flow
- Gates control information flow
- Resent and Update gates are often close to either 0 or 1 due to using sigmoid
- New gate is used as a preliminary candidate to update the state variable





Reset Gate $$r_t = \sigma(W_{ir}x_t + W_{hr}h_{(t-1)})$$

Update Gate $$z_t = \sigma(W_{iz}x_t + W_{hz}h_{(t-1)})$$

New Gate $$\tilde{h}_t = \tanh(W_{in}x_t + r_t \odot (W_{hn}h_{(t-1)}))$$

$$h_t = (1 - z_t) \odot h_{(t-1)} + z_t \odot \tilde{h}_t$$

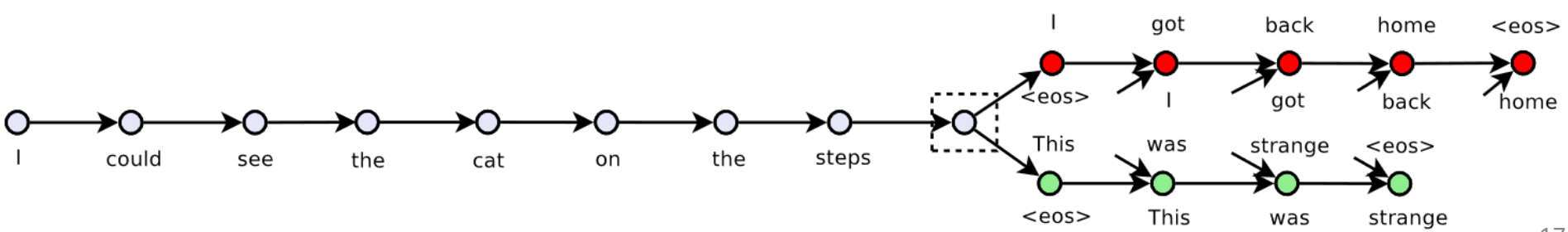
Hadamard Product

Language Processing Hierarchy Levels

- Hierarchy levels:
- Documents
- **Sentences**
- Phrase
- Words
- The hierarchy is determined based on application
- Ex: Document-level representation is more natural for document classification, but word representation makes more sense to do semantic inference tasks
- The success of Word2Vec led exploring possibility of extending a similar approach to structures in other hierarchies that have similar relationships
- Information Retrieval

Skip-Thought Vectors

- An encoder-decoder model:
- Encoder: maps a sentence into a vector
- Decoder: conditions on this vector to generate surrounding sentences
- Architecture: RNN encoder with GRU activations, RNN decoder with conditioned GRU
- Benefit: Skip-thoughts sentence representations lead to robust performance across sentence-level tasks



Skip-Thought Vectors

- Encoder Structure:

- Let w_i^j be the word j in sentence I, where N is the number of words in that sentence
- At each time step, the encoder produces a hidden state h_i^t as the representation of the sequence $w_i^1, \ldots, w_i^t$
- The hidden state h_i^N represents the full sentence

$$
\begin{aligned}
\mathbf{r}^t &= \sigma(\mathbf{W}_r \mathbf{x}^t + \mathbf{U}_r \mathbf{h}^{t-1}) \\
\mathbf{z}^t &= \sigma(\mathbf{W}_z \mathbf{x}^t + \mathbf{U}_z \mathbf{h}^{t-1}) \\
\bar{\mathbf{h}}^t &= \tanh(\mathbf{W}\mathbf{x}^t + \mathbf{U}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \\
\mathbf{h}^t &= (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \bar{\mathbf{h}}^t
\end{aligned}
$$

Skip-Thought Vectors

- Decoder Structure:

- The matrices C_z, C_r, and C that are used to bias the update gate, reset gate and hidden state computation by the sentence vector
- Separate decoders are used for previous and next sentences
- Parameters for each decoder are separated

$$
\begin{aligned}
\mathbf{r}^t &= \sigma(\mathbf{W}_r^d \mathbf{x}^{t-1} + \mathbf{U}_r^d \mathbf{h}^{t-1} + \mathbf{C}_r \mathbf{h}_i) \\
\mathbf{z}^t &= \sigma(\mathbf{W}_z^d \mathbf{x}^{t-1} + \mathbf{U}_z^d \mathbf{h}^{t-1} + \mathbf{C}_z \mathbf{h}_i) \\
\bar{\mathbf{h}}^t &= \tanh(\mathbf{W}^d \mathbf{x}^{t-1} + \mathbf{U}^d (\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{C}\mathbf{h}_i) \\
\mathbf{h}_{i+1}^t &= (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \bar{\mathbf{h}}^t
\end{aligned}
$$

Given $\mathbf{h}_{i+1}^t$, the probability of word w_{i+1}^t given the previous $t-1$ words and the encoder vector is

$$P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) \propto \exp(\mathbf{v}_{w_{i+1}^t} \mathbf{h}_{i+1}^t)$$

Skip-Thought Vectors

- Objective function: Given the sentence tuple (s_{i-1}, s_i, s_{i+1}) the objective function is the sum of the log-probabilities for the forward and backward sentences conditioned on the encoder representation:

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)$$

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

Primarily based on slides prepared by I-Hung Hsu

Logistical Notes

- Project:
  - 259 students: 52 groups with one group with 4 members
  - 14 unassigned students
  - Proposal deadline: 3 weeks
- Written Assignment:
  - Start peer grading soon after the deadline

PYTORCH

- It’s a python-based scientific computing package
- A library that helps using the power of GPUs
- Why popular?
- PyTorch vs TensorFlow

Installation

- Follow instructions in https://pytorch.org/get-started/locally/

PyTorch Build	Stable (1.9.0)	Preview (Nightly)	LTS (1.8.2)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python	C++ / Java		
Compute Platform	CUDA 10.2	CUDA 11.1	ROCm 4.2 (beta)	CPU
Run this Command:	`conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch`			

PyTorch

PyTorch data structure format

- Tensor: similar to Numpy array but runs on GPU or other hardware accelerators.
- (Trainable) Tensor: stores data and gradient and can be considered similar to a variable

$$y = \beta^T x$$

- Module: A neural network layer; may store state or learnable weights

Tensors vs Numpy Arrays

PyTorch Tensors are just like numpy arrays, but they can run on GPU.

Can have high dimensions

Tensor operations: x.mm(), torch.add(x,y), etc

Numpy Conversion: x.numpy()

Example

Here a two-layer net is trained using PyTorch Tensors.

- Forward Pass

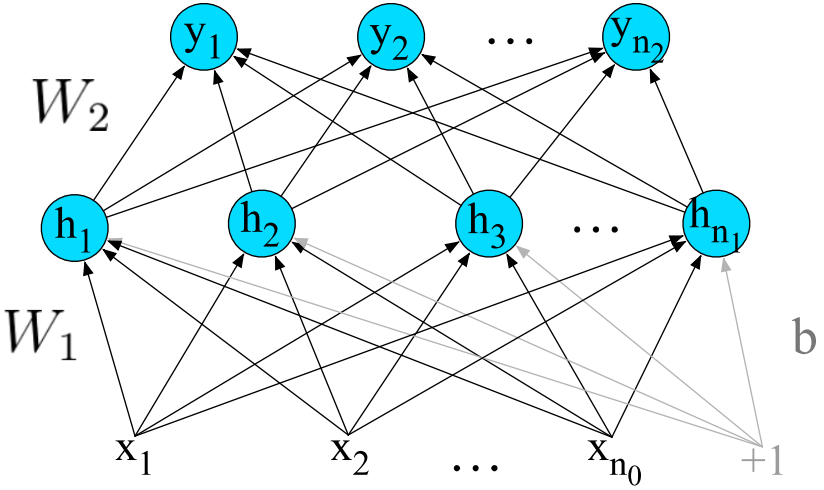
$$\mathcal{L} = \|y - \hat{y}\|_2^2$$

$$y = W_2^T ReLu(W_1^T x)$$

- Backword Pass

$$W^{i+1} = W^i - \eta \frac{d\mathcal{L}}{dW^i}$$

$$\frac{d\mathcal{L}}{dW_2} = 2(y - \hat{y}) \frac{d\hat{y}}{dW_2}$$



Optimization with Tensors

Create random tensor for data and weight

Tensor can also be loaded by:

1. Load from data (list)

```
data = [[1, 2],[3, 4]]
x_data = torch.tensor(data)
```

2. From Numpy array

```
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
```

3. From another tensor

```
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
```

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

lr = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred-y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred-y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= lr*grad_w1
    w2 -= lr*grad_w2
```

Optimization with Tensors

Forward pass: compute predictions and loss

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

lr = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred-y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred-y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= lr*grad_w1
    w2 -= lr*grad_w2
```

Optimization with Tensors

Backward pass: manually computed gradients ***if you don't have autograd***.

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

lr = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred-y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred-y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= lr*grad_w1
    w2 -= lr*grad_w2
```

Optimization with Tensors

Optimization: Gradient descent step on weights

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

lr = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred-y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred-y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= lr*grad_w1
    w2 -= lr*grad_w2
```

Optimization with Tensors

To run on GPU, just cast tensors to a cuda datatype

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

lr = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred-y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred-y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= lr*grad_w1
    w2 -= lr*grad_w2
```

Autograd

The previous process:

- Slow
- Gradient is hard to compute when model becomes more complex

=> **This is a primary reason for popularity of PyTorch**

Trainable Tensors:

Set “required_grad” to True to enable torch.autograd

Optimization with Autograd

We set the weights' "requires_grad" to be True

```
dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

w3 = w1.detach().clone()
w3.requires_grad=True
# To initialize, you can use:
# w3 = torch.randn(D_in, H, requires_grad=True).type(dtype)

w4 = w2.detach().clone()
w4.requires_grad=True
```

Optimization with Autograd

Forward pass looks exactly the same as the Tensor/Numpy version.

```
lr = 1e-6
for t in range(500):
    y_pred2 = x.mm(w3).clamp(min=0).mm(w4)
    loss = (y_pred2-y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w3 -= lr * w3.grad
        w4 -= lr * w4.grad

    w3.grad.zero_()
    w4.grad.zero_()
```

Optimization with Autograd

But the gradient of loss with respect to w3 and w4 can be done by a simple one-line code.

```
lr = 1e-6
for t in range(500):
    y_pred2 = x.mm(w3).clamp(min=0).mm(w4)
    loss = (y_pred2-y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w3 -= lr * w3.grad
        w4 -= lr * w4.grad

    w3.grad.zero_()
    w4.grad.zero_()
```

```
grad_y_pred = 2.0 * (y_pred-y)
grad_w2 = h_relu.t().mm(grad_y_pred)
grad_h_relu = grad_y_pred.mm(w2.t())
grad_h = grad_h_relu.clone()
grad_h[h < 0] = 0
grad_w1 = x.t().mm(grad_h)
```

Optimization with Autograd

Make gradient step on weights.

What's torch.no_grad()?

- We need to use NO_GRAD to keep the update out of the gradient computation
- Why is that? It boils down to the DYNAMIC GRAPH that PyTorch uses.

```
lr = 1e-6
for t in range(500):
    y_pred2 = x.mm(w3).clamp(min=0).mm(w4)
    loss = (y_pred2-y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w3 -= lr * w3.grad
        w4 -= lr * w4.grad

    w3.grad.zero_()
    w4.grad.zero_()
```

Optimization with Autograd

grad.zero()?

- After each gradient descent step, we should restart from zero
- Why is that? Gradients are accumulated

```
lr = 1e-6
for t in range(500):
    y_pred2 = x.mm(w3).clamp(min=0).mm(w4)
    loss = (y_pred2-y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w3 -= lr * w3.grad
        w4 -= lr * w4.grad

    w3.grad.zero_()
    w4.grad.zero_()
```

New Autograd Functions

We can define our own autograd functions by writing forward and backward for Tensors

```
class ReLU(torch.autograd.Function):

    @staticmethod
    def forward(ctx, x):
        ctx.save_for_backward(x)
        return x.clamp(min=0)

    @staticmethod
    def backward(ctx, grad_y):
        x, = ctx.saved_tensors
        grad_input = grad_y.clone()
        grad_input[x < 0] = 0
        return grad_input
```

New Autograd Functions

```python
class ReLU(torch.autograd.Function):

    @staticmethod
    def forward(ctx, x):
        ctx.save_for_backward(x)
        return x.clamp(min=0)

    @staticmethod
    def backward(ctx, grad_y):
        x, = ctx.saved_tensors
        grad_input = grad_y.clone()
        grad_input[x < 0] = 0
        return grad_input
```

```python
lr = 1e-6
for t in range(500):
    y_pred3 = ReLU.apply(x.mm(w5)).mm(w6)
    loss = (y_pred3-y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w5 -= lr * w5.grad
        w6 -= lr * w6.grad

    w5.grad.zero_()
    w6.grad.zero_()
```

We then apply our new function in the forward pass.

Networks: torch.nn

- High-level wrapper for creating neural networkss
- This is another reason behind popularity of PyTorch and TensorFlow
- Various classes of neural networks can be built using buil-in modules, e.g., CNN, RNN, LSTM, transformers, etc
- A diverse set of hyperparamters are buil-in implemnted, e.g., various activation function, loss functions, etc.

torch.nn

Define our model as a sequence of layers

NN also can be used for common loss functions

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)

model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out)
)
model.cuda()

loss_fn = torch.nn.MSELoss(reduction='sum')

def weights_init(m):
    if isinstance(m, torch.nn.Linear):
        torch.nn.init.zeros_(m.weight)
        torch.nn.init.ones_(m.bias)

model.apply(weights_init)
print(model)

lr = 1e-6

for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    model.zero_grad()
    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param.data -= lr * param.grad.data
```

torch.nn

Forward pass:

- Feed data to model
- Use prediction to and ground truth to get loss function

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)

model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out)
)
model.cuda()

loss_fn = torch.nn.MSELoss(reduction='sum')

def weights_init(m):
    if isinstance(m, torch.nn.Linear):
        torch.nn.init.zeros_(m.weight)
        torch.nn.init.ones_(m.bias)

model.apply(weights_init)
print(model)

lr = 1e-6

for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    model.zero_grad()
    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param.data -= lr * param.grad.data
```

torch.nn

Pytorch generates autograd easily

We should use model.zero_grad() to clear all gradients for the parameters in the model

```
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)

model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out)
)
model.cuda()

loss_fn = torch.nn.MSELoss(reduction='sum')

def weights_init(m):
    if isinstance(m, torch.nn.Linear):
        torch.nn.init.zeros_(m.weight)
        torch.nn.init.ones_(m.bias)

model.apply(weights_init)
print(model)

lr = 1e-6

for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    model.zero_grad()
    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param.data -= lr * param.grad.data
```

Optimizer

Make gradient step on each model parameter.

Question:

- How can we apply more advanced rules for updating

```python
import torch

dtype = torch.cuda.FloatTensor
N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)

model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out)
)
model.cuda()

loss_fn = torch.nn.MSELoss(reduction='sum')

def weights_init(m):
    if isinstance(m, torch.nn.Linear):
        torch.nn.init.zeros_(m.weight)
        torch.nn.init.ones_(m.bias)

model.apply(weights_init)
print(model)

lr = 1e-6

for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    model.zero_grad()
    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param.data -= lr * param.grad.data
```

Optimizer

Call nn.optim package, which contains various advanced optimizer other than SGD.

Now, all the parameters can be updated via one-line code.

```
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    #model.zero_grad()
    optimizer.zero_grad()
    loss.backward()

    optimizer.step()
#     with torch.no_grad():
#         for param in model.parameters():
#             param.data -= lr * param.grad.data
```

Define new modules

Pytorch **Module** is *a neural network layer*, it can contain weights or other modules.

```
class TwoLayerMLP(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
        super().__init__()
        self.linear1 = torch.nn.Linear(D_in, H)
        self.linear2 = torch.nn.Linear(H, D_out)
        self.weight_init()

    def weight_init(self):
        torch.nn.init.zeros_(self.linear1.weight)
        torch.nn.init.zeros_(self.linear2.weight)
        torch.nn.init.ones_(self.linear1.bias)
        torch.nn.init.ones_(self.linear2.bias)

    def forward(self, x):
        h_relu = self.linear1(x).clamp(min=0)
        y_pred = self.linear2(h_relu)
        return y_pred

model = TwoLayerMLP(D_in, H, D_out)
model.cuda()
print(model)
loss_fn = torch.nn.MSELoss(reduction='sum')
lr = 1e-6

optimizer = torch.optim.Adam(model.parameters(), lr=lr)
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

DataLoaders

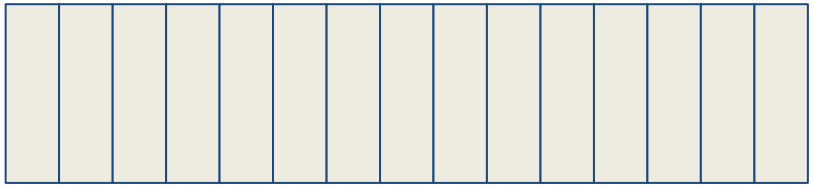
A **DataLoader** wraps a **Dataset** and provides minibatching, multithreading, etc.

When you need to load custom data, just **write your own Dataset class**

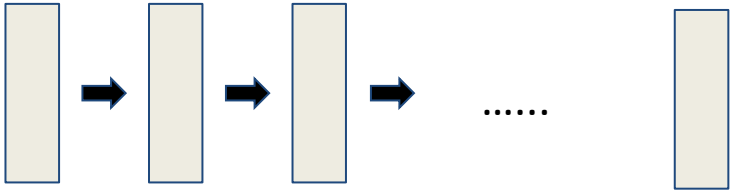
```
import torch
from torch.utils import data

class Dataset(data.Dataset):
  'Characterizes a dataset for PyTorch'
  def __init__(self, list_IDs, labels):
        'Initialization'
        self.labels = labels
        self.list_IDs = list_IDs

  def __len__(self):
        'Denotes the total number of samples'
        return len(self.list_IDs)

  def __getitem__(self, index):
        'Generates one sample of data'
        # Select sample
        ID = self.list_IDs[index]

        # Load data and get label
        X = torch.load('data/' + ID + '.pt')
        y = self.labels[ID]

        return X, y
```

(Mini)batching

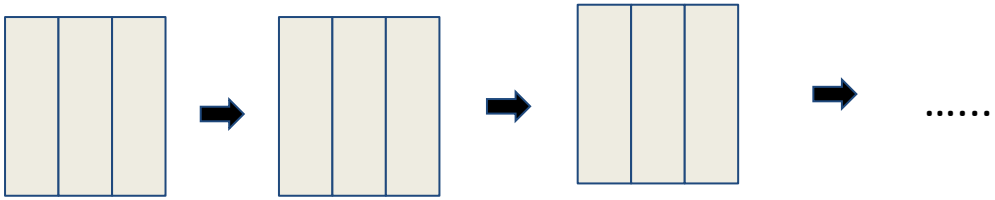
All your training data:



Standard for loop:



(Mini) batching:



Adapt Dataset to DataLoaders

```
# Parameters
params = {'batch_size': 64,
          'shuffle': True,
          'num_workers': 6}
max_epochs = 100

# Datasets
partition = # IDs
labels = # Labels

# Generators
training_set = Dataset(partition['train'], labels)
training_generator = data.DataLoader(training_set, **params)

validation_set = Dataset(partition['validation'], labels)
validation_generator = data.DataLoader(validation_set, **params)

# Loop over epochs
for epoch in range(max_epochs):
    # Training
    for local_batch, local_labels in training_generator:
        # Transfer to GPU
        local_batch, local_labels = local_batch.to(device), local_labels.to(device)

        # Model computations
        [...]
```

DataLoader perform batching automatically

Summary

1. Prepare you data
    a. Write your own Dataset (inherit torch.nn.util.dataset)
2. Create your model
    a. A sequential module if your model is super easy and will not be reused.
    b. A nn.Module module
3. Write the loop (how many epoch/steps) to train your model:
    a. Create a DataLoader that wraps the Dataset you provide
    b. Set an optimizer
    c. Set a loss for optimization
    d. For loop….
        i. Forward pass
        ii. Zero-grad
        iii. Backward pass => Get gradient
        iv. Optimizer step to update your models' weight.

Using Different Version of Data for Efficieny

- Tiny-size: for debugging syntactic bug
- Small-size: check the behavior of the model
- Mid-size: for understanding model behavior and fast development
- Full-size: conduct final experiments

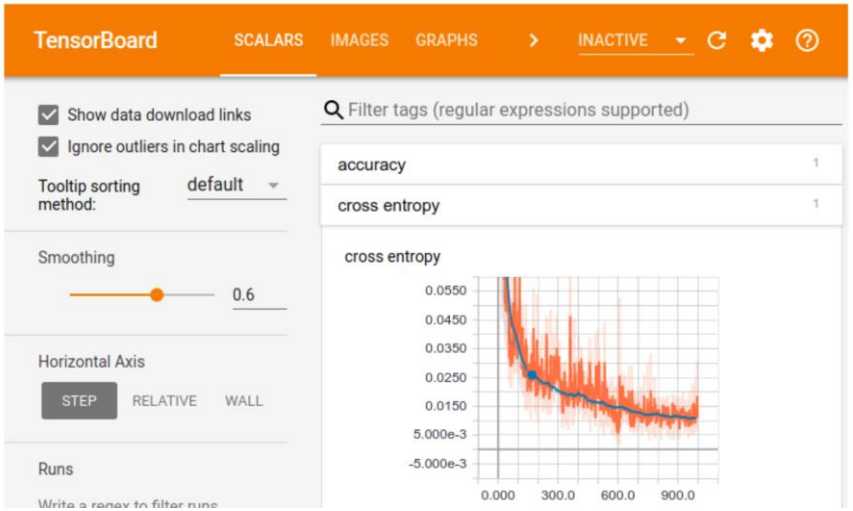
Tensorboard

- Installation: https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html
- Create a Summary Writer

```
from torch.utils.tensorboard import SummaryWriter
writer = SummaryWriter()
```

- During Training/Dev/Test

```
writer.add_scalar("Loss/train", loss, epoch)
```



End to end Examples

- https://www.analyticsvidhya.com/blog/2020/01/first-text-classification-in-pytorch/
- https://towardsdatascience.com/lstm-text-classification-using-pytorch-2c6c657f8fc0

CSCI 544, Lecture 8: Dialogue

Ron Artstein

2022-09-15

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for attending class, participating in the discussion, and taking notes. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Introducing myself

Ron Artstein

- Research Scientist at ICT
- Natural Language Dialogue
- Linguist by training
- Teaching this course since 2016

Applied NLP

- Until 2013, this course was simply called “NLP”.
- Concentrate on NLP applications.
- Learn to think about the language in the problem.

Administrative notes

Written Assignment due tonight

Written Assignment Peer Grading starts tomorrow, due September 22

Coding Assignment 2 due September 27

Project:

Due Date	Task
September 20	Form project teams (52 teams)
September 20–29	Initial discussion with assigned TA
October 4	**Project proposal**
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

More administrative notes

Use **Piazza** for questions

Make Piazza questions **public** whenever possible

- Other students can answer
- Other students can see the answer

This is a big class; many students have similar issues

Ambiguity and an NLP application

— Hey Siri, tell Nancy David wants spaghetti.

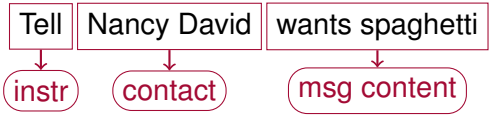
— I'm sorry, I couldn't find a Nancy David in your contacts.

Observations:

- The intended contact was "Nancy".
- Local ambiguity resolved by the end of the utterance:

  Tell Nancy David. . .

We can guess how Siri works:



- Parser does not see my contacts
- Parser might not consider grammaticality of full utterance.

☛ Many design considerations

☛ Think of the design for your project!

Dialogue systems

Dialogue systems are very common these days

- Phones: Siri (2011), Cortana (2015–2021), Google (2016), . . .
- Speakers: Alexa (2014), Google (2016), . . .

Much older history

- Spoken: >30 years
- Text-based: >50 years (Eliza)

Can be topic of a whole course

- ☛ CSCI 644 Natural Language Dialogue Systems

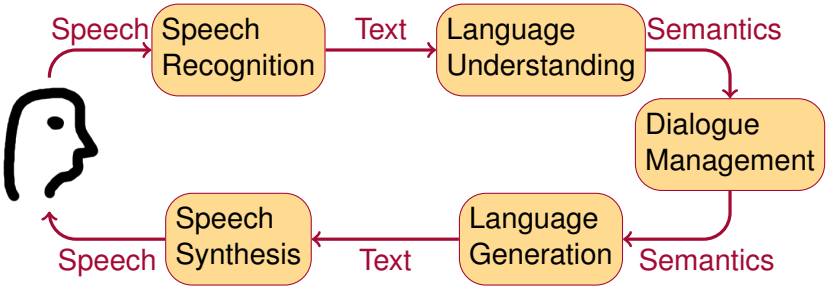
Structure of today's lecture

1. Dialogue system architecture
2. Dialogue management for task-based systems
3. Conversational systems
4. Evaluation of dialogue systems

Task/goal-based:	Specific, external task
Conversational:	Sustain conversation

Most systems have some of both elements
Conversational goals (even without external goal)

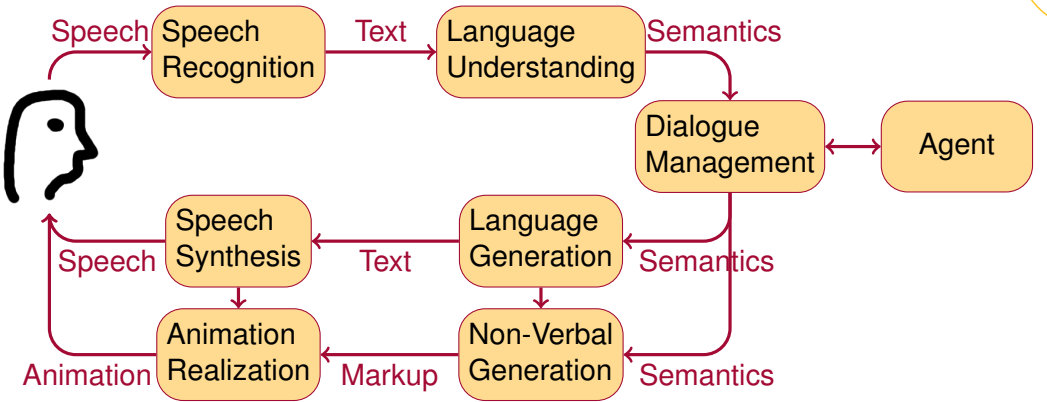
Spoken dialogue systems: prototypical architecture



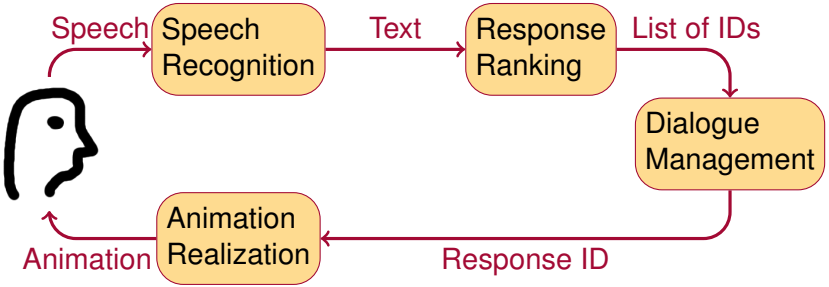
Information flow, not pipeline

- Input does not necessarily result in output

Spoken dialogue systems: additional components



Spoken dialogue systems: fewer components



Dialogue management

Policy/Strategy Action to take at each point in the dialogue

- Requires taxonomy of actions
- May relate to user input (but not only)

Dialogue State Where the conversation is now

- A set of state variables

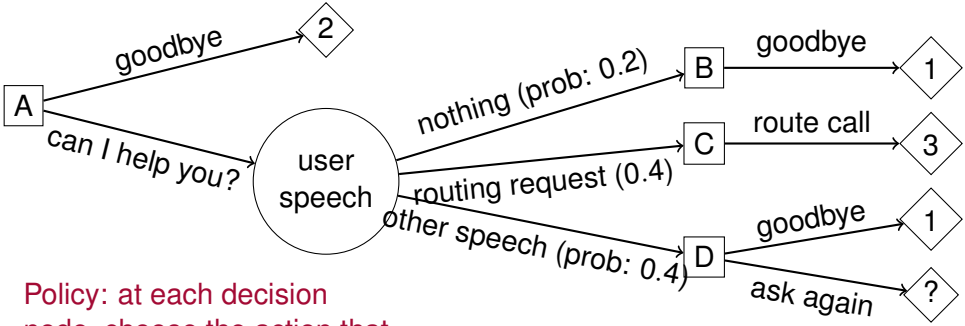
Example Policy Always maximize expected utility

- Need to calculate expected utility

Dialogue acts
Assertion
Question
Request
Command
Promise
Threat
etc.

Decision trees

□ Action node ○ Observation node ◇ Reward node



Policy: at each decision node, choose the action that maximizes expected utility

Markov Decision Process

Sets of states S, actions A, and rewards R (or costs)

- Probabilistic transition between states
- Probabilistic rewards
- Actions are chosen by the agent

Markov property for MDPs

- Probability of transition, reward depends only on the current state and action

$$
\begin{aligned}
P(s_{t+1}|\text{history}) &\approx P(s_{t+1}|s_t, a_t) \\
P(r_{t+1}|\text{history}) &\approx P(r_{t+1}|s_t, a_t)
\end{aligned}
$$

Rewards can also be fixed (for each state-action pair)

MDP policies

Policy What action to take at each point in the dialogue

- A function from states to actions: $\pi : S \rightarrow A$

Optimal policy $\pi : S \rightarrow A$ which maximizes expected reward

- = minimizes expected cost

MDP model assumes the system always knows what state it is in

Partially Observable MDP

Add a set of observations O and emission probabilities

- The probability of an observation depends only on the current state and action

$$P(o_{t+1}|\text{history}) \approx P(o_{t+1}|s_{t+1}, a_t)$$

Belief (at time t) = distribution over states

Learning (PO)MDPs

Need to learn: Transition probabilities
Observation probabilities
Immediate rewards (or reward probabilities)

Can we learn from a corpus?

- Corpus reflects the policy that produced the corpus

Reinforcement learning: learn from interaction

- update probabilities, rewards based on interaction
- Reward good outcomes, penalize bad outcomes
- Propagate rewards backwards

Requires a lot of interaction

Simulated users

User simulation

- Simulate actions
- Simulate utterances (including ASR and other errors)

Simulated users often learned from a corpus

When rewards are unknown: Inverse reinforcement learning

- Learn the reward function that results in a policy that mimics expert users

Dialogue state tracking

Dialogue State = where the conversation is now

- A set of state variables
- For task-oriented: which slots have been filled

Dialogue System Technology ~~State Tracking~~ Challenge

Relatively well-defined for simple tasks

Complex tasks might

- share slot values between tasks
- express complex goals in a single utterance
- interleave related tasks

Task lineages: framework

Lee and Stent (2016). Task Lineages: Dialog State Tracking for Flexible Interaction. Sigdial.

Multi-task dialogue (slot-filling)

What is a task? — Book a restaurant, book a ride, . . .

- Each task requires some information in order to be executed

Task schema required and optional slots for operationalization

Dialogue act item slot + value

Task frame parse tasks + associated DAI + confidence + · · ·

Task lineage history of states; maintain in parallel due to ambiguity

Update function extend, then prune (global)

Conversational dialogue systems

Emphasis on maintaining conversation

Additional goals: education, entertainment, connection

New Dimensions in Testimony demo

- Conversational question-answering character
- Primarily reactive (responds to user questions)

To be continued...

CSCI 544, Lecture 9: Dialogue (cont.); Experiment design; Annotation

Ron Artstein

2022-09-20

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for attending class, participating in the discussion, and taking notes. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes: deadlines

Written Assignment Peer Grading due September 22

- So far, about 40% of the students have completed the grading

Coding Assignment 2 due September 27

Project:

Due Date	Task
September 20	Form project teams (52 teams)
September 20–29	**Initial discussion with TA**
October 4	**Project proposal**
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Pick your TA!

Administrative notes: missed class

From the course syllabus:

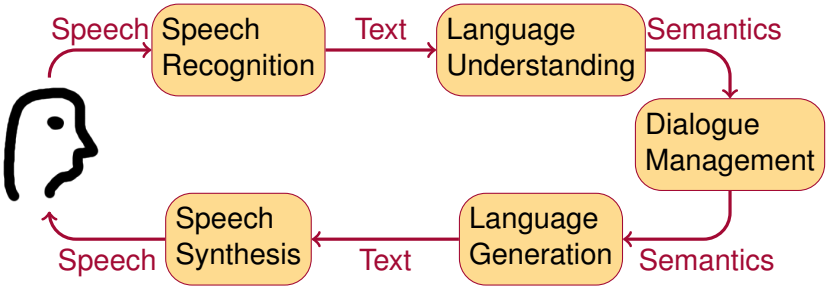
- Students who are absent from class for any reason must make up the materials themselves, and must submit their assignments on time.
- We will make the presentation slides available on Blackboard, and will endeavor to provide lectures both in-person and on Zoom.

☛ Responsibility is **on the student.**

Also on the course syllabus:

- There will be a small number of extra quizzes to allow students to make up for quizzes missed due to late registration, illness, or other reasons. Make-up quizzes will be given after all of the regular quizzes have been taken.

Spoken dialogue systems: prototypical architecture



Information flow, not pipeline

- Input does not necessarily result in output

Types of dialogue systems

Task/goal-based:	Specific, external task
Conversational:	Sustain conversation

Most systems have some of both elements
Conversational goals (even without external goal)

Dialogue state tracking

Dialogue State = where the conversation is now

- A set of state variables
- For task-oriented: which slots have been filled

Dialogue System Technology ~~State Tracking~~ Challenge

Relatively well-defined for simple tasks

Complex tasks might

- share slot values between tasks
- express complex goals in a single utterance
- interleave related tasks

Task lineages: framework

Lee and Stent (2016). Task Lineages: Dialog State Tracking for Flexible Interaction. Sigdial.

Multi-task dialogue (slot-filling)

What is a task? — Book a restaurant, book a ride, . . .

- Each task requires some information in order to be executed

Task schema required and optional slots for operationalization

Dialogue act item slot + value

Task frame parse tasks + associated DAI + confidence + · · ·

Task lineage history of states; maintain in parallel due to ambiguity

Update function extend, then prune (global)

Conversational dialogue systems

Emphasis on maintaining conversation

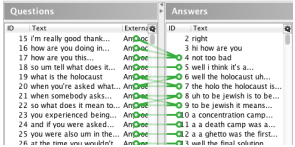
Additional goals: education, entertainment, connection

New Dimensions in Testimony demo

- Conversational question-answering character
- Primarily reactive (responds to user questions)

Retrieval-based system

Train = linked questions and answers



Estimate most likely language model for the response to Q:

$$P(w|Q) \cong \frac{\sum_j \pi_{R_j}(w) \prod_{q \in \mathrm{tok}(Q)} \pi_{S_j}(q)}{\sum_j \prod_{q \in \mathrm{tok}(Q)} \pi_{S_j}(q)}$$

Where: (S_j, R_j) are all linked question-response pairs
π_A is the language model of utterance A

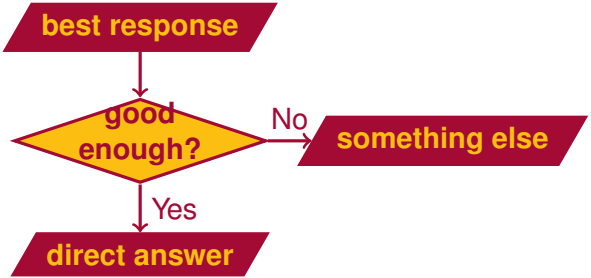
Rank available responses based on distance to estimated LM.

Top-ranked response is the most appropriate.

Non-understanding

Direct response not always available

What's your favorite restaurant?



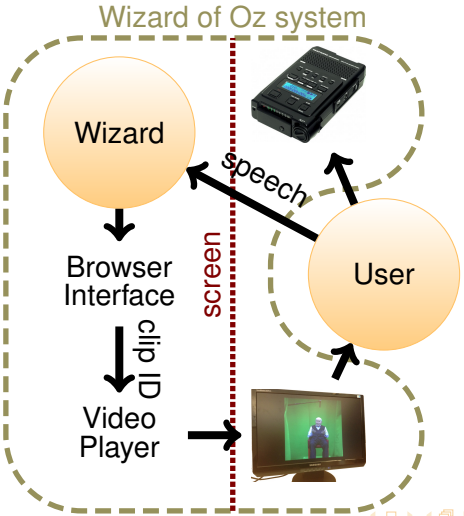
Key to success: good data

Initial system with responses

Collect data in interaction

New responses to fill gaps

More questions for training



Generation

Retrieval has limited expressibility

Generation:

- Template
- Grammar
- Sequence-to-sequence (neural)

Conditioned on what?

Evaluation

Typical evaluation of NLP: compare to “gold standard” reference

- Accuracy
- Precision, recall, F-measure

Evaluation of generation: lexical similarity to reference

- Word Error Rate (speech recognition)
- BLEU, METEOR (translation); ROUGE (summarization)
- Multiple references capture lexical variation

Are lexical similarity measures good for dialogue?

- Much more lexical variation in appropriate utterances

CSCI 544, Lecture 10: Experiment design; Annotation

Ron Artstein

2022-09-22

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for attending class, participating in the discussion, and taking notes. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes: deadlines

Written Assignment Peer Grading due tonight

- So far, about 72% of the students have completed the grading

Coding Assignment 2 due September 27

Project:

Due Date	Task
September 20	Form project teams (52 teams)
September 20–29	**Initial discussion with TA**
October 4	**Project proposal**
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Pick your TA!

Evaluation

Typical evaluation of NLP: compare to “gold standard” reference

- Accuracy
- Precision, recall, F-measure

Evaluation of generation: lexical similarity to reference

- Word Error Rate (speech recognition)
- BLEU, METEOR (translation); ROUGE (summarization)
- Multiple references capture lexical variation

Are lexical similarity measures good for dialogue?

- Much more lexical variation in appropriate utterances

Current dialogue evaluation practices

Mix of methods (Finch and Choi 2020)

- Automated: similarity to reference, similarity to context
- Human-rated: appropriateness, coherence, consistency

☛ Judge quality of a contribution

Automated measures that try to predict human ratings

Experiment design

Independent variables Manipulated by the experimenter; control

Dependent variables Measured to see if affected by the independent variables

How can we tell if the dependent variable really is affected?

> **If you need statistics to prove the results of your experiment, then you ought to have designed a better experiment.**
>
> *Attributed to Lord Ernest Rutherford*

Hypothesis testing

Is the observed outcome due to random sampling?

Null hypothesis No effect; results due to random sampling

Alternative hypothesis Results **not** due to random sampling

Statistical model: probability of various outcomes if only random sampling is at play

- ☛ If probability of observed results is low, reject null hypothesis

Some common statistical tests

- t-test: means of two samples
- ANOVA: means of multiple samples
  - ☛ main effects; interactions; simple effects
- chi-squared test: compare frequencies of categories
- correlation, regression: two (or more) continuous variables

Confounds and controls

Outcome may not be the result of random sampling, but not our variable either

Confound Another variable that may affect the result

Control variable Not interested in it, but may affect results

Random variable A variable we can't control

Participants and language

Human participants may be affected by order, etc.

Between subjects Assign different participants to each condition

Within subjects (repeated measures) Same participants in all conditions.

In language studies, the choice of items may also have an effect: not all verbs/nouns are the same.

Herbert H. Clark, The language-as-fixed-effect fallacy, Journal of Verbal Learning and Verbal Behavior 12(4):335–359, 1973

The empirical revolution (1990s)

Large-scale, broad coverage systems

Emphasis on **formal evaluation** of performance

- Track improvement over time, compare systems
- Typically single component (e.g. parser, tagger)
- Competitions: single task, multiple teams

Quantitative evaluation

- Reference target ("gold standard")
- Precision, recall, F-measure
- Objective measures (task success, time on task)

Automatic evaluation allows machine learning

Annotated corpora

Annotated corpora are needed for:

- Supervised learning – training and evaluation
- Unsupervised learning – evaluation
- Hand-crafted systems – evaluation
- Analysis of text

Annotations need to be **correct**.

Why measure annotator agreement

Agreement is measured between annotations of a single text.

Reliability measures consistency of an instrument.

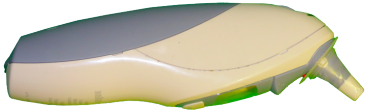
Validity is the correctness relative to a desired standard.

Reliability is a property of a process

Repeated measures with two thermometers

Mercury $\pm 0.1\,^{\circ}\mathrm{C}$

Infrared $\pm 0.4\,^{\circ}\mathrm{C}$



The mercury thermometer is more reliable.

- But what if it's not calibrated properly?

Reliability is a **minimum requirement** for an annotation process.

- Qualitative evaluation also necessary.

Reliability and agreement

Reliability = **consistency** of annotation

- Needs to be measured on the same text.
- Different annotators.
- Work **independently**

If independent annotators mark a text the same way, then:

- They have internalized the same scheme (instructions).
- They will apply it consistently to new data.
- Annotations may be correct.

Results **do not generalize** from one domain to another.

Observed (pairwise) agreement

Observed agreement: proportion of items on which 2 coders agree.

Detailed Listing

Item	Coder 1	Coder 2
a	Boxcar	Tanker
b	Tanker	Boxcar
c	Boxcar	Boxcar
d	Boxcar	Tanker
e	Tanker	Tanker
f	Tanker	Tanker
	⋮	⋮

Contingency Table

	Boxcar	Tanker	Total
Boxcar	41	3	44
Tanker	9	47	56
Total	50	50	100

Agreement: $\frac{41 + 47}{100} = 0.88$

High agreement, low reliability

Two psychiatrists evaluating 1000 patients.

	Normal	Paranoid	Total
Normal	990	5	995
Paranoid	5	0	5
Total	995	5	1000

- Observed agreement = 990/1000 = 0.99
- Most of these patients probably aren't paranoid
- No evidence that the psychiatrists identify the paranoid ones
- High agreement **does not indicate** high reliability

Chance agreement

Some agreement is expected by chance alone.

- Randomly assign two labels $\rightarrow$ agree half of the time?
  - Depends on the distributon!
- The amount expected by chance varies depending on the annotation scheme and on the annotated data.

Meaningful agreement is the agreement **above chance**.

Correction for chance

How much of the observed agreement is above chance?

	A	B	Total
A	44	6	50
B	6	44	50
Total	50	50	100

Total		*Chance*		*Above*	
44	6	**6**	6	**38**	0
6	**44**	6	**6**	0	**38**
88		**12**		**76**	

$$\begin{bmatrix} \mathbf{44} & 6 \\ 6 & \mathbf{44} \end{bmatrix} = \begin{bmatrix} \mathbf{6} & 6 \\ 6 & \mathbf{6} \end{bmatrix} + \begin{bmatrix} \mathbf{38} & 0 \\ 0 & \mathbf{38} \end{bmatrix}$$

Agreement: 88/100
Due to chance: 12/100
Above chance: 76/100

Expected agreement

Observed agreement (A_o)*: proportion of actual agreement*
Expected agreement (A_e)*: expected value of* A_o

Amount of agreement above chance: $A_o - A_e$
Maximum possible agreement above chance: $1 - A_e$

Proportion of agreement above chance attained: $\mathbf{\frac{A_o - A_e}{1 - A_e}}$

Scott's Pi, Fleiss's Kappa, Siegel and Castellan's K

Total number of judgments: $N = \sum_q \mathbf{n}_q$

Probability of one coder picking category q: $\frac{\mathbf{n}_q}{N}$

Prob. of two coders picking category q: $\left(\frac{\mathbf{n}_q}{N}\right)^2$ *[biased estimator]*

Prob. of two coders picking same category: $\mathrm{A_e} = \sum_q \left(\frac{\mathbf{n}_q}{N}\right)^2$

	Normal	*Paran*	*Total*
Normal	990	5	995
Paranoid	5	0	5
Total	995	5	1000

$\mathrm{A_o}=0.99$

$\mathrm{A_e}=.995^2 + .005^2 = 0.99005$

$K = \frac{0.99-0.99005}{1-0.99005} \approx -0.005$

Interpreting agreement

Agreement measures are not hypothesis tests

- Evaluating magnitude, not existence/lack of effect
- Not comparing two hypotheses
- No clear probabilistic interpretation

Textbook usage paradigm

Conduct a reliability study with:

- Written annotation guidelines
- Generally available coders
- Representative sample of annotation materials

In order to validate annotation **scheme** and **procedure**

With a good procedure, will annotations will be correct?

Annotation model

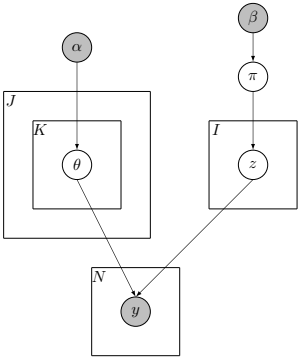
Annotation errors are not random

Affected by:

- Item category
- Annotator
- (Item difficulty)

Use this information to infer true label

- Graphical model
- Multiple annotations
- EM algorithm
- Confidence level



Passonneau and Carpenter (2014), Figure 1

Differences in the annotated material

Kang et al. 2012 , AAMAS: identify smiles in videos

- Smiles are easier to detect on some people than others

Not all coders are equal

Scott, Barone and Koeling, LREC 2012

- Annotate hedges in medical text as likelihood

> **Possible** early pneumonia. . .
> . . . **could** represent pneumonia. . .

- Two annotator populations differ in **medical training**
- Systematic differences between annotators: medically trained interpret hedges as expressing greater likelihood

Each population of coders (instrument) has a certain reliability, but one is probably more correct.

CSCI 544, lecture 11: Part-of-speech tagging, Hidden Markov models

Ron Artstein

2022-09-27

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for watching the lecture video, taking notes, and participating in class discussions. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes: deadlines

Written Assignment Peer Grading will be released after checking for errors

Coding Assignment 2 was due today

Coding Assignment 3 due October 11

Project:

Due Date	Task
September 20	Form project teams (52 teams)
September 20–29	**Initial discussion with TA**
October 4	**Project proposal**
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Parts of speech

Syntactic category of a word:

- Noun, verb, . . .

May differ between languages, grammatical theories

Open-class	Noun, verb, adjective, . . .
Closed-class	Preposition, conjunction, determiner, . . .

Tagging: in a text, identify the part of speech of each word.

Part-of-speech tagging

Task driven by ambiguity

Somewhat artificial task.

Why would we want to tag words with POS?

Part-of-speech tagging

Task driven by ambiguity

Somewhat artificial task.

Why would we want to tag words with POS?

- Could be helpful for downstream processes.
- Push technology forward.

Time flies like an arrow

Part-of-speech tagging

Task driven by ambiguity

Somewhat artificial task.

Why would we want to tag words with POS?

- Could be helpful for downstream processes.
- Push technology forward.

Time	**flies**	**like**	**an**	**arrow**	
NN	VBZ	IN	DT	NN	(Penn Treebank tags)

Part-of-speech tagging

Task driven by ambiguity

Somewhat artificial task.

Why would we want to tag words with POS?

- Could be helpful for downstream processes.
- Push technology forward.

Time	**flies**	**like**	**an**	**arrow**	
NN	VBZ	IN	DT	NN	(Penn Treebank tags)
NN	NNS	VBP	DT	NN	

Part-of-speech tagging

Task driven by ambiguity

Somewhat artificial task.

Why would we want to tag words with POS?

- Could be helpful for downstream processes.
- Push technology forward.

Time	**flies**	**like**	**an**	**arrow**	
NN	VBZ	IN	DT	NN	(Penn Treebank tags)
NN	NNS	VBP	DT	NN	
VB	NNS	IN	DT	NN	

Long-distance dependencies

Flying	planes	can	be	dangerous
{VBG, JJ}	NNS	MD	VB	JJ

Flying	planes	is	dangerous
VBG	NNS	VBZ	JJ

Flying	planes	are	dangerous
JJ	NNS	VBP	JJ

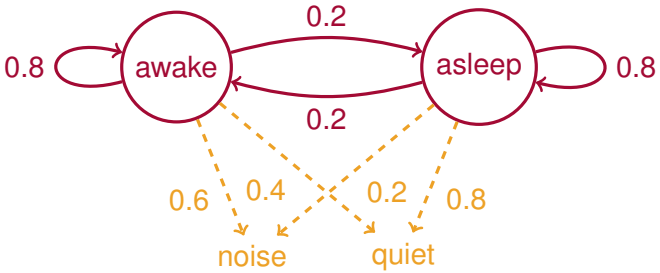
Sequence labeling: each instance may depend on preceding or following labels.

Markov chains

Probabilistic state machine.

- **Transition probabilities:** state → state
- **Emission probabilities:** state → observation

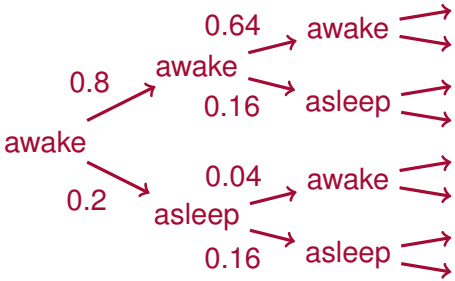
Markov assumption:
Probabilities depend only on state, not on history

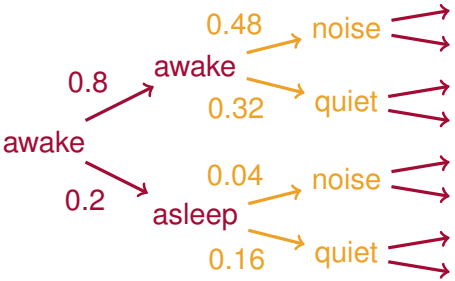


	Awake	Asleep
Awake	0.8	0.2
Asleep	0.2	0.8

	Noise	Quiet
Awake	0.6	0.4
Asleep	0.2	0.8

Markov chains are a generative model





Hidden Markov model

Only emissions are observable; states are hidden

- Infer states from observations (most likely sequence)

Decoding (known transition and emission probabilities)

- Viterbi algorithm

Learning probability matrices

- Learn from corpus
- Forward-backward algorithm = special case of EM
  - Expectation: estimate states based on parameters
  - Maximization: estimate parameters based on states

Learning state structure

- Design manually
- Start with fully connected, then learn probabilities

Viterbi decoding example

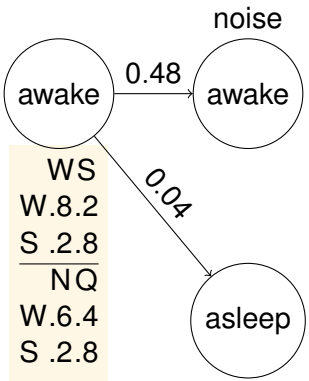
noise quiet noise quiet quiet

awake

	W	S
W	.8	.2
S	.2	.8

	N	Q
W	.6	.4
S	.2	.8

Viterbi decoding example

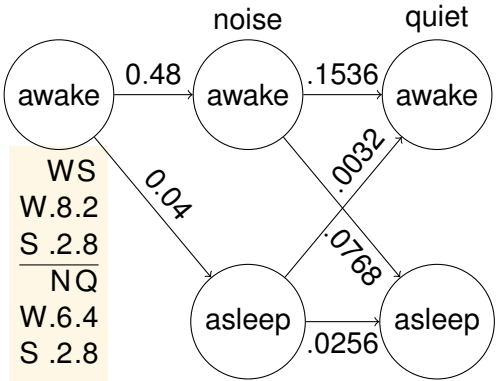


	W	S
W	.8	.2
S	.2	.8

	N	Q
W	.6	.4
S	.2	.8

quiet noise quiet quiet

Viterbi decoding example

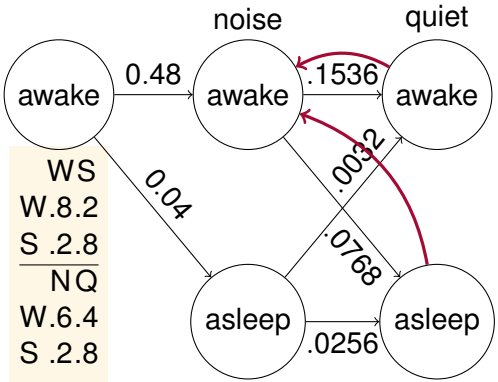


	W	S
W	.8	.2
S	.2	.8

	N	Q
W	.6	.4
S	.2	.8

noise quiet quiet

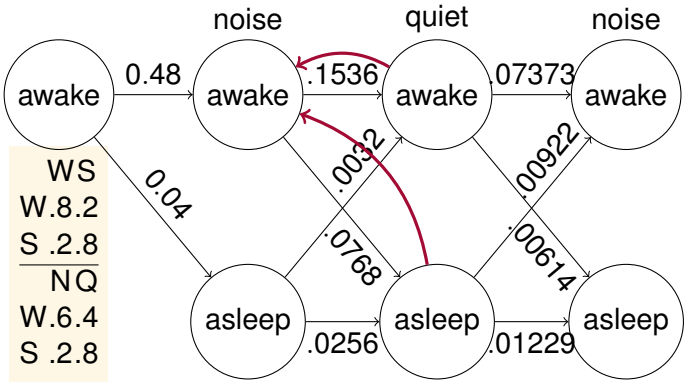
Viterbi decoding example



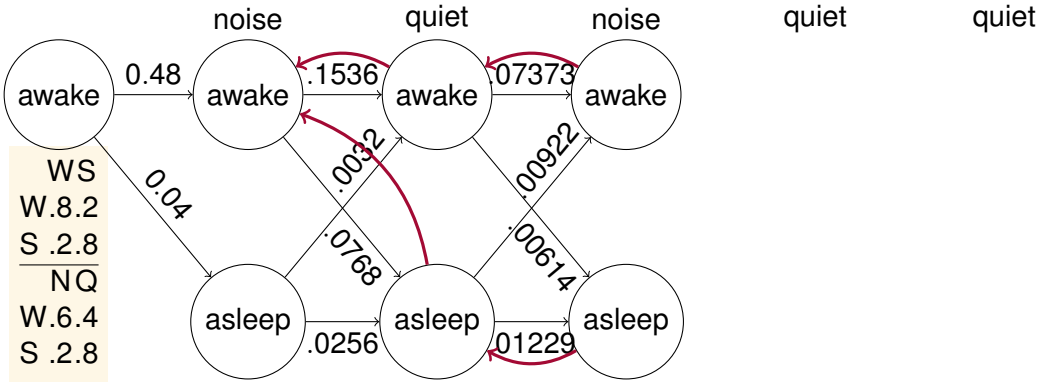
	WS
W	.8.2
S	.2.8
	NQ
W	.6.4
S	.2.8

noise quiet quiet

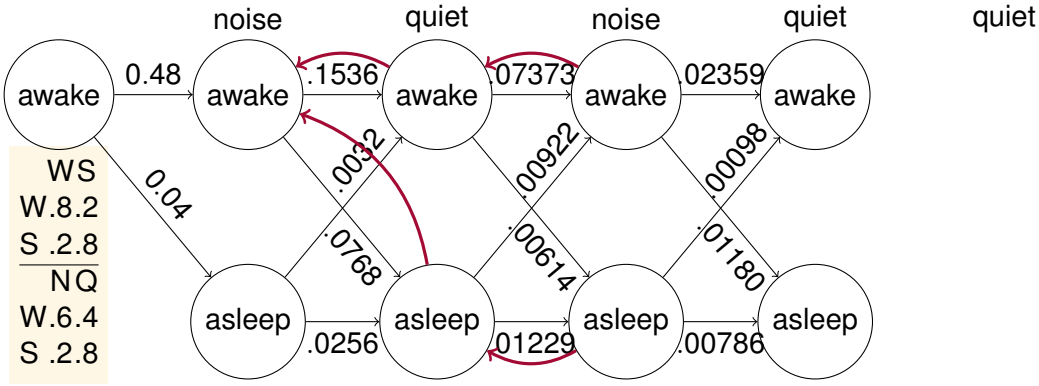
Viterbi decoding example



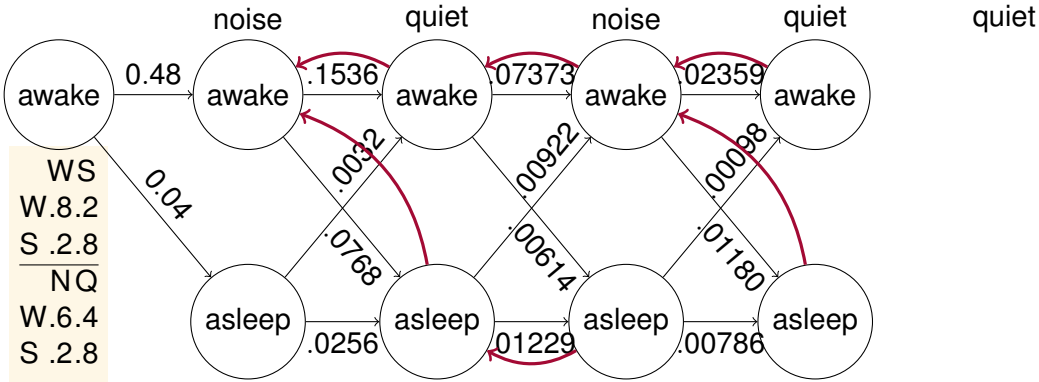
Viterbi decoding example



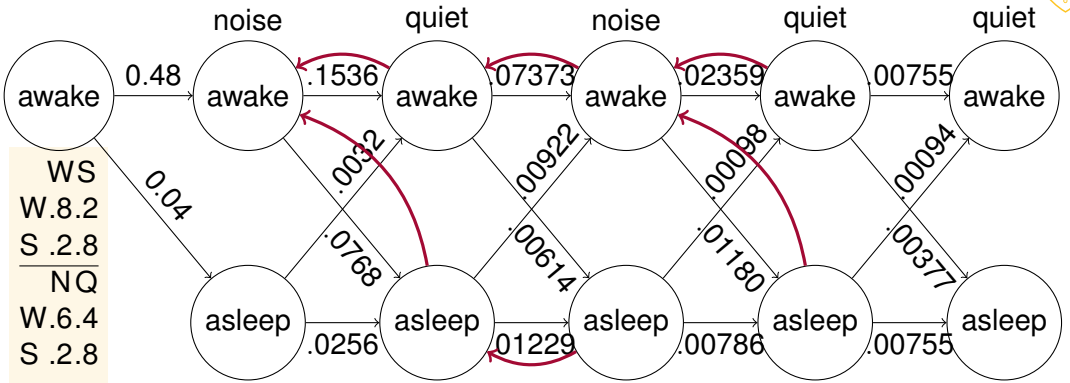
Viterbi decoding example



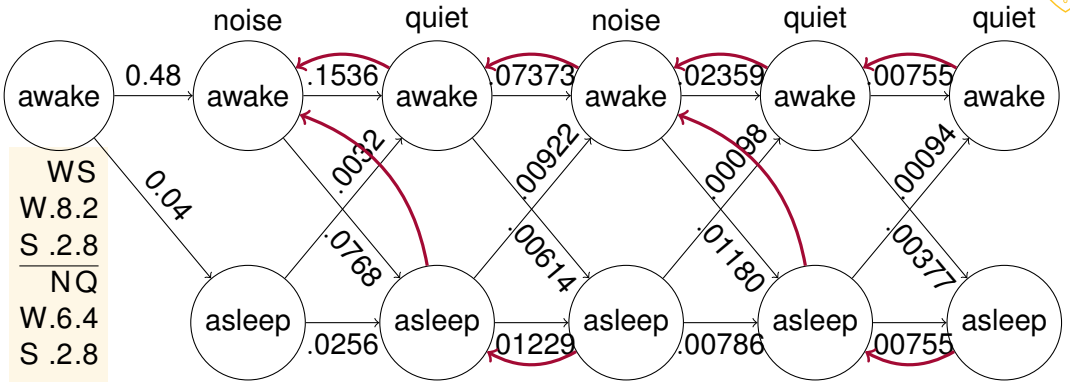
Viterbi decoding example



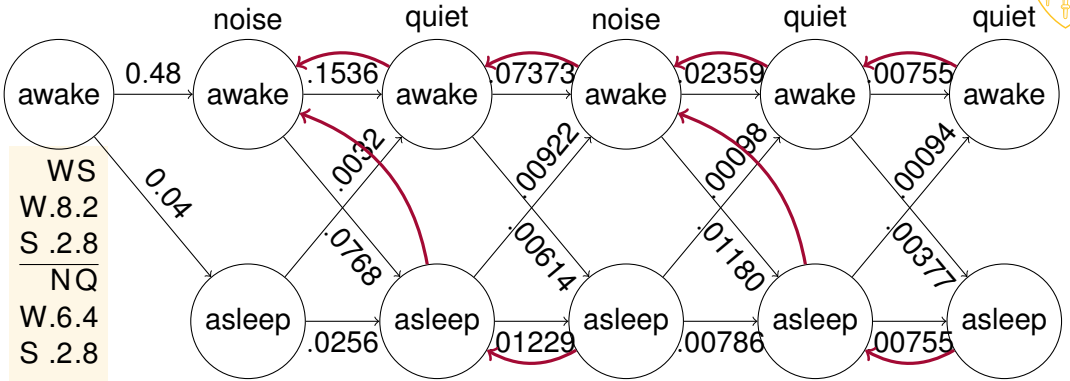
Viterbi decoding example



Viterbi decoding example



Viterbi decoding example



(previous · transition) · emission ≈ prior · conditional

HMMs for POS tagging

Observations = words; what are the states?

- States = POS tags
- Viterbi decoding = infer most probable tag sequence

Learn parameters from corpus

- Emission matrix = tag→word probabilities $P(\text{word}|\text{current tag})$
- Transition matrix = tag bigram probabilities $P(\text{tag}|\text{previous tag})$

Issues with decoding

- How to handle unknown words?
- Is smoothing useful?

More context can be given with richer states (e.g., tag bigrams)

Coding Assignment 3

Write a Hidden Markov Model part-of-speech tagger

From scratch!

Two langauges for training and development

- Test on unseen data in same langauges
- Test on surprise language

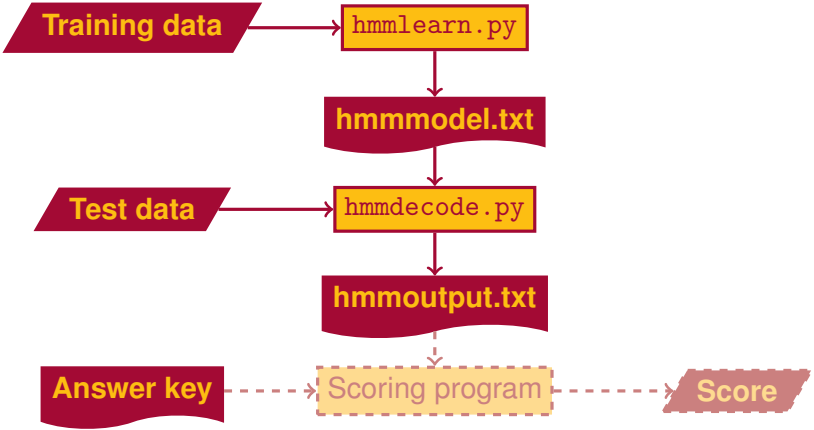
Graded on performance

Programming in Python

Submit on Vocareum

- Automatic feedback
- Submit early, submit often!

Coding assignment 3: programs



Coding assignment 3: notes

Word/tag format

- Learn tags from training data (surprise language!)
- Careful with slash character

Unseen words and transitions

- Unseen words do not have known emissions
- Unseen transitions can cause program to halt
- Reference uses smoothing on transitions, **not** emissions

Runtime efficiency

	Python 2	Python 3
`if item in dict:`	**Fast**	**Fast**
`if item in dict.keys():`	**Slow**	**Fast**

- Don't multiply zeros

Coding assignment 3: decoding efficiency

Probability of state *q* at time *t*

$$P(s,t) = \max_{s' \in State} P(s', t-1) * P(tr(s', s)) * P(em(s, o_t))$$

When $P(em(s, o_t)) = 0$, product doesn't depend on s'!

Since most emissions are 0, an if-statement cuts 90% of runtime

```
if 0 == em[q][obs[t]]:
    prob[q][t] = 0
else:
    prob[q][t] = max (prob[q1][t-1] * tr[q1][q] for q1 in Q) * em[q][obs[t]]
```

Style:	`if x == 0:`
Better style:	`if 0 == x:`

Coding assignment 3: linguistic insights

Tagging unseen words

- Unseen words do not have known emissions
- Use transition probabilities alone

Errors with unseen words

- Common error: nouns, verbs incorrectly tagged as prepositions, articles
- Rare error: prepositions, articles incorrectly tagged as nouns, verbs

Open-class	Noun, verb, adjective, . . .
Closed-class	Preposition, conjunction, determiner, . . .

Reducing overall error

- Tag unseen words only with open-class labels!
- How do we know which labels are open class?
  - ☛ Open-class items have a large vocabulary

CSCI 544, lecture 12: More Tagging: Brill, Maximum Entropy, Conditional Random Fields, Unsupervised

Ron Artstein

2022-09-29

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for watching the lecture video, taking notes, and participating in class discussions. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes: deadlines

Written Assignment Peer Grading has been released

Coding Assignment 2 was due today (really!)

Coding Assignment 3 due October 11

Project:

Due Date	Task
September 20	Form project teams (52 teams)
September 20–29	**Initial discussion with TA**
October 4	**Project proposal**
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

HMM tagging

Tag sentences using co-occurrence statistics: tag–tag, tag–word

Better performance than previous models:

- Most common tag for each word
- Rule-based

HMM can model some linguistic knowledge

Not a very satisfying model

- Not a clear relation to language knowledge

Can we have a model that performs as well as HMM, but is more interpretable?

Brill tagger

Brill (1992): A Simple Rule-Based Part of Speech Tagger

Model that is more understandable than HMMs

Initial tagger learned on 90% of the corpus

- Most common tag for each word
- Capitalized unseen words $\rightarrow$ proper nouns
- Other unseen words $\rightarrow$ most common tag for last three characters

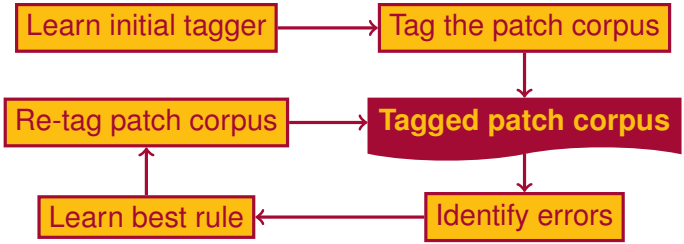
Patch rules learned on 5% of the corpus

- Fix output of the initial tagger
- Learn linguistic rules that minimize error

Test on 5% of the corpus

Brill tagger: iterative process

Each iteration learns one rule, which minimizes error on the patch corpus



Rules must be applied in the order they were learned

Brill tagger rule templates

Templates generate many possible rules

- Change tag A to tag B in context C
  e.g., if previous (or following) word is tagged Z
- Change tag A to tag B if a word has property P
  e.g., if word is capitalized
- Change tag A to tag B if a word in region R has property P
  e.g., if previous (or following) word is capitalized

Procedure for finding best rule

- Find most common error (e.g., noun tagged as verb)
- Find best rule to correct that error

Brill tagger rules

Rules are expressed with Brown corpus tags

- **TO IN NEXT-TAG AT**
  TO: *to* eat, *to* drink (complementizer)
  IN : *to* the market (preposition)
  "A word tagged TO is likely a preposition (IN) when occurring before an article (AT)"
- **VBN VBD PREV-WORD-IS-CAP YES**
  "A word tagged as a passive participle (VBN) is likely a past-tense verb (VBD) when occurring after a capitalized word"

Most rules learned on patch corpus also reduce error on test corpus

Maximum entropy tagger

Ratnaparkhi (1996): A Maximum Entropy Model for Part-Of-Speech Tagging

Demonstrate the advantages of a maximum entropy model

- Maximum entropy = logistic regression

Individually classify each instance

- Sequence properties in the features
  e.g., previous one or two tags, previous and following words
- Spelling features for rare and unseen words

Calculate probabilities for the full path

- Can't remember too many paths: **beam search**
- Individual features only look within small window

Beam search

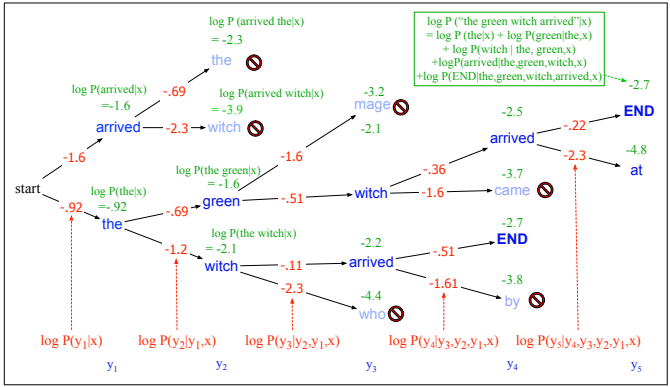


Figure 11.13 Scoring for beam search decoding with a beam width of $k = 2$. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step.

Jurafsky and Martin, formerly chapter 11, now chapter 10

Maximum entropy tagger (cont.)

Tag dictionary to reduce search space

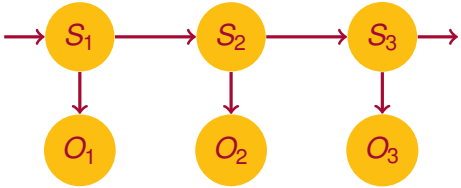
- Known words only consider tags with which they were seen
  - Minimal effect on accuracy
  - Substantial reduction in runtime

Upper bound on possible performance

- Some words are still difficult to tag
  e.g., *about*: preposition or adverb?
- Specialized features for these words show little improvement
- Hypothesis: these words are inconsistently tagged in the Penn Treebank
  - ☛ Tag distribution varies by annotator

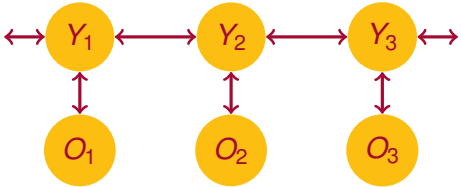
Discriminative sequence labeling

Hidden Markov Models



- Transition probabilities
- Emission probabilities
- Model generates observations
- Viterbi decoding

Conditional Random Fields (Lafferty, McCallum and Pereira 2001)



- Conditional (discriminative) model: $P(\text{labels}|\text{observations})$
- Edge features
- Vertex features

Conditional Random Fields

Conditional model learns probability of entire sequence of labels

- Avoids the **label bias problem** of next-state conditional models

Markov property for CRF: the only labels that affect the probability of a given label are the immediately preceding and following labels

Parameter estimation for CRFs: learn weights for edge features and vertex features

Tested on synthetic data, generated using a second-order Markov model

- Data cannot be learned perfectly by first-order models

Rich features relate observations to labels

- Better performance on out-of-vocabulary items

Field segmentation

Task: Learn to identify information fields in a classified ad (or bibliographic citation, or other similarly structured text) in an **unsupervised** way.

Size Features
Spacious 1 Bedroom apt. newly remodeled, gated, new appliance,
Features Location Rent
new carpet, near public transportion, close to 580 freeway, $500.00
Rent Contact
Deposit (510)655-0106

Trond Grenager, Dan Klein, and Christopher Manning. Unsupervised Learning of Field Segmentation Models for Information Extraction. ACL 2005

Unsupervised sequence labeling

Unsupervised Hidden Markov Models

Learning probability matrices

- Forward-backward algorithm = special case of EM
  - Expectation: estimate states based on parameters
  - Maximization: estimate parameters based on states

Learning state structure

- Start with fully connected, then learn probabilities

But HMMs can also model part-of-speech tags, which have a very different structure. So which structure will the model learn?

Constraining the model 1

Adjacent words tend to belong to the same field

- Bias transition matrix towards same-state transitions
  σ: probability of staying within the field

$$P(s_t|s_{t-1}) = \begin{cases} \sigma + \dfrac{1-\sigma}{|S|} & \text{if } s_t = s_{t-1} \\ \dfrac{1-\sigma}{|S|} & \text{otherwise} \end{cases}$$

Accuracy 49% $\rightarrow$ 70%

Is this still a Hidden Markov Model?

Constraining the model 2

Punctuation, function words occur in all fields

- Mix general emission with state emission
  α: probability of selecting a common term

$$P(w|s) = \alpha P_{common}(w) + (1-\alpha)P_{any}(w|s)$$

Accuracy $\rightarrow$ 71%

Is this still a Hidden Markov Model?

Constraining the model 3

State transitions tend to happen after boundary symbols

- Create separate boundary (s^+) and non-boundary (s^-) states
  σ, λ: probability of staying within the field;
  μ: probability of transitioning to boundary

$$P(s'|s^+) = \begin{cases} \sigma + \frac{1-\sigma}{|S^-|} & \text{if } s' = s^- \\ \frac{1-\sigma}{|S^-|} & \text{if } s' \in S^- \setminus s^- \\ 0 & \text{otherwise} \end{cases} \qquad P(s'|s^-) = \begin{cases} (1-\mu)(\lambda + \frac{1-\lambda}{|S^-|}) & \text{if } s' = s^- \\ \mu(\lambda + \frac{1-\lambda}{|S^-|}) & \text{if } s' = s^+ \\ \frac{1-\lambda}{|S^-|} & \text{if } s' \in S^- \setminus s^- \\ 0 & \text{otherwise} \end{cases}$$

Accuracy $\rightarrow$ 73%

Is this still a Hidden Markov Model?

Lessons from unsupervised field segmentation

Performance still below supervised methods

- But that's not the point

Learning the desired structure is possible

- Need more explicit constraints than in supervised learning

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

Slides are adopted from slides by Jurafsky and Martin

Logistical Comments

- HW3: Oct 11
- Project Status Report: Nov 3
- Dividing tasks
- Expected novelty:
- Need to be beyond assignments
- Projects will be graded relatively. The more creative project will need less empirical exploration

Probabilistic Language Models

- Language model: assigns a probability to a given sentence:
  - Spell Correction

    The office is about fifteen minutes form my house

    P(about fifteen minutes **from**) >> P(about fifteen minutes **form**)
  - Speech Recognition

    P(I saw a van) >> P(eyes awe of an)

    P(I listen to Pat and I see Mark) >> P(I listen to Pa Tennessee Mark)
  - Machine Translation:

    P(**high** winds tonight) > P(**large** winds tonight)

    P(**Put on** your shoes) > P(**Wear** your shoes)

Probabilistic Language Models

- Goal: compute the probability of a given sentence or sequence of words

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5 | w_1,w_2,w_3,w_4)$$

- A model that computes either of $P(W)$ or $P(w_n | w_1,w_2 \ldots w_{n-1})$ is called a **language model**
- **language model** or **LM** is the standard term, but in a sense, we are modeling **the grammar** of the language

Probabilistic Language Models

- How can we compute a probability for a sentence

- Naïve idea: prepare a training dataset? (infeasible)

- We can use the sequential property of a sentence:

$P(w_1,w_2,w_3,\ldots,w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\ldots P(w_n|w_1,\ldots,w_{n-1})$

P("its water is so transparent") = P(its) × P(water|its) × P(is|its, water) × P(so|its, water, is) × P(transparent|its, water, is, so)

$P(w_n|w_1,\ldots,w_{n-1}) = \frac{Count(w_1,\ldots,w_n)}{Count(w_1,\ldots,w_{n-1})}$

Ex: P(Machine| "One of the AI branches is") = 2/7

Using Google and the Web

- Too many possible sentences!

Ex: (count("One of the AI branches is NLP" = 0))

- We'll never see enough data for estimating all possibilities

Language Models with Local Dependencies

- Markov simplification

$P(w_n|w_1,w_2...w_{n-1}) = P(w_n|w_{n-1})$

P(NLP| "One of the AI branches is") = P(NLP| "is")

=135,000 /23,440,000,000

- More generally for:

$$P(w_1,w_2...,w_n) = \prod_{i=1}^{n} P(w_i|w_1,w_2...w_{i-1}) = \prod_{i=1}^{n} P(w_i|w_k,w_{k+1...}w_{i-1})$$

Unigram

- We consider full independence:

$$P(w_1,w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i)$$

- Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Bigram

- We consider H=1:

$$P(w_1, w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

- Some automatically generated sentences from a unigram model

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

N-Grams

- We can extend to trigrams, 4-grams, …
- N-grams are not good generative model of language because language data has **long-distance dependencies**:

  "The computer which I had just put into the machine room on the fifth floor crashed."

- But we can often use N-gram models to perform downstream tasks, e.g., spellcheck, machine translation

Computing Bigram Probabilities

- Maximum likelihood estimation:
- Build a training dataset
- Estimate the probabilities using frequencies

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Computing Bigram Probabilities

- Example:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Corpus

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $\quad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $\quad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Berkeley Restaurant Project

- Presented at ICLSP-94: spoken language turn into text data (9222 sentences)

- can you tell me about any good Cantonese restaurants close by
- mid priced Thai food is what I'm looking for
- tell me about chez Panisse
- can you give me a listing of the kinds of food that are available
- I'm looking for a good place to eat breakfast
- when is caffe Venezia open during the day

- Bigram Counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Berkeley Restaurant Project

- Bigram Probabilities

  Normalize by unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

  Resulting probabilities:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

  Estimating probabilities of sentences:

  P(<s> I want chinese food </s>) = P(I|<s>) × P(want|I) × P(chinese|want) × P(food|chinese) × P(</s>|food)= P(I|<s>) * 0.33 * 0.0065*0.52 * P(</s>|food)

  P(<s> I chinese want food </s>) = P(I|<s>) × P(chinese|I) × P(want|chinese) × P(food|want) × P(</s>|food)= P(I|<s>) * 0.0063 * 0* 0.0065 * P(</s>|food)

Accessible Language Models

- We do computations in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

- Language Modeling Toolkits

- SRILM: http://www.speech.sri.com/projects/srilm/
- KenLM:https://kheafield.com/code/kenlm/
- NLTK: https://www.nltk.org/api/nltk.lm.html
- Google: https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Language Model Evaluation

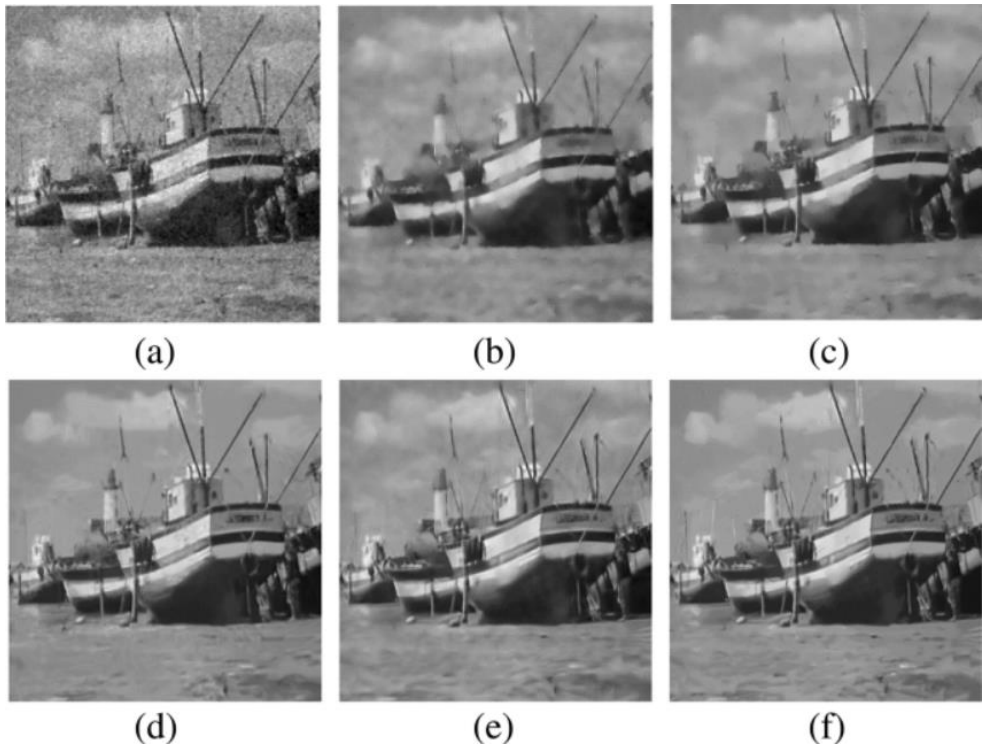
- What is a good language model?

- Assigns higher probability to “corrent” or “frequently observed” sentences Than “ungrammatical” or “rarely observed” sentences?

- We train parameters of our model on a training set and test the model’s performance on a testing set
- An **evaluation metric** is used to measure how well our model does on the test set.

Metrics

- Full Reference: Extrinsic



- Blind Metric: Intrinsic

Extrinsic Evaluation of N-Gram Models

- We can compare two models A and B by testing each model on a common task
- spelling corrector, speech recognizer, MT system
- Run the task, get an accuracy for A and for B

  How many misspelled words corrected properly?

  How many words translated correctly?
- Compare accuracy for A and B

- Extrinsic evaluation is time-consuming and depends on a task that may take a long time for completion

Intrinsic Evaluation of N-Gram Models

- **Perplexity**

- Only an approximation
  - the test data should look like the training data
  - generally only useful in pilot experiments (still helpful)

- Intuition: the best language model is the one that on an unseen test set

- Gives the highest P(sentence)
- Lower perplexity = better model

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

For bigrams

Intrinsic Evaluation of N-Gram Models

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

N-Gram Generalization

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

N-Gram Generalization

- Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

N-Gram Generalization

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

N-Gram Generalization

- The Wall Street Journal

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

N-Gram Generalization

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, this often is not the case
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P("offer" | denied the) = 0

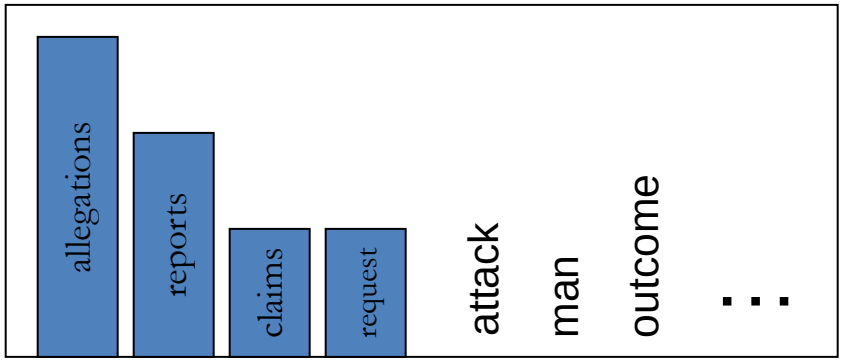
- Test set
  … denied the offer
  … denied the loan

Smoothing

- Bigrams with zero probability
  - will assign 0 probability to the test set!
- Perplexity cannot be computed (can't divide by 0)!

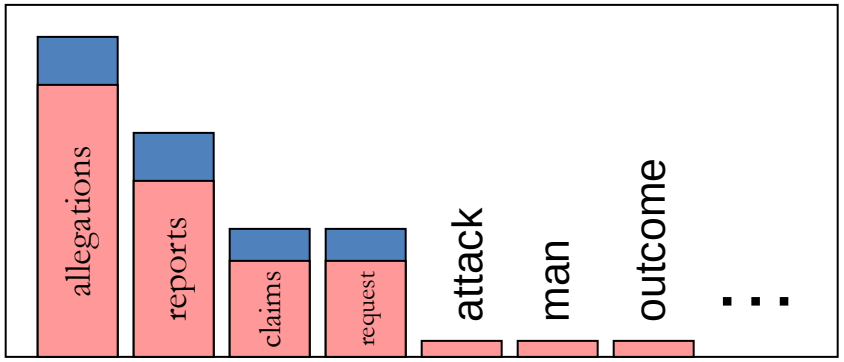
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Laplace Smoothing

- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate: $$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate: $$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

- Reconstitutes counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

- Suitable for domains where the number of zeros isn't so huge
- Domains with high-level tasks, e.g., text classification.

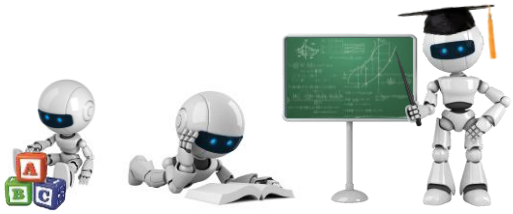
	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

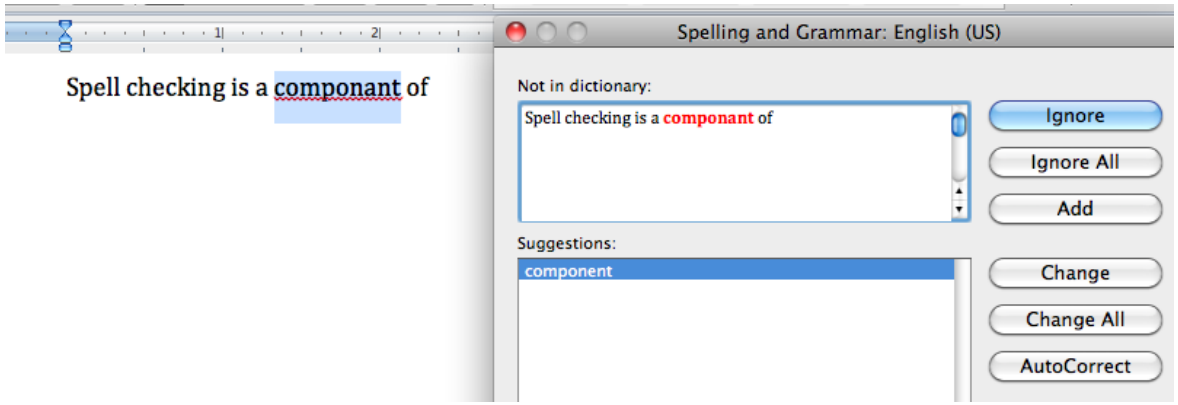


Slides adopted from slides by Jurfsky, Martin, and Collins

Spelling Correction

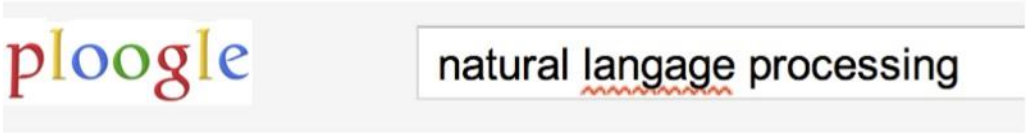
- Applications

Word processing



Phones

Web search



Showing results for natural ***language*** processing
Search instead for natural langage processing

Spelling Correction

- How common are spelling errors:

- **26**%: Web queries Wang *et al.* 2003
- **13**%: Retyping, no backspace: Whitelaw *et al.* English&German
- **7**%: Words corrected retyping on phone-sized organizer
- **1-2**%: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

Spelling Correction

- Spelling Tasks:

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect

    hte→the
  - Suggest a correction
  - Suggestion lists

- Spelling error types:

- Non-word Errors
  - *graffe →giraffe*
- Real-word Errors
  - Typographical errors
    - *three →there*
  - Cognitive Errors *piece→peace,*
    - *too → two*

Non-word Spelling Errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better
- Non-word spelling error correction:
  - Generate ***candidates***: real words that are similar to error
  - Choose the one which is best: noisy channel model

Real Word Spelling Errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spelling***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel

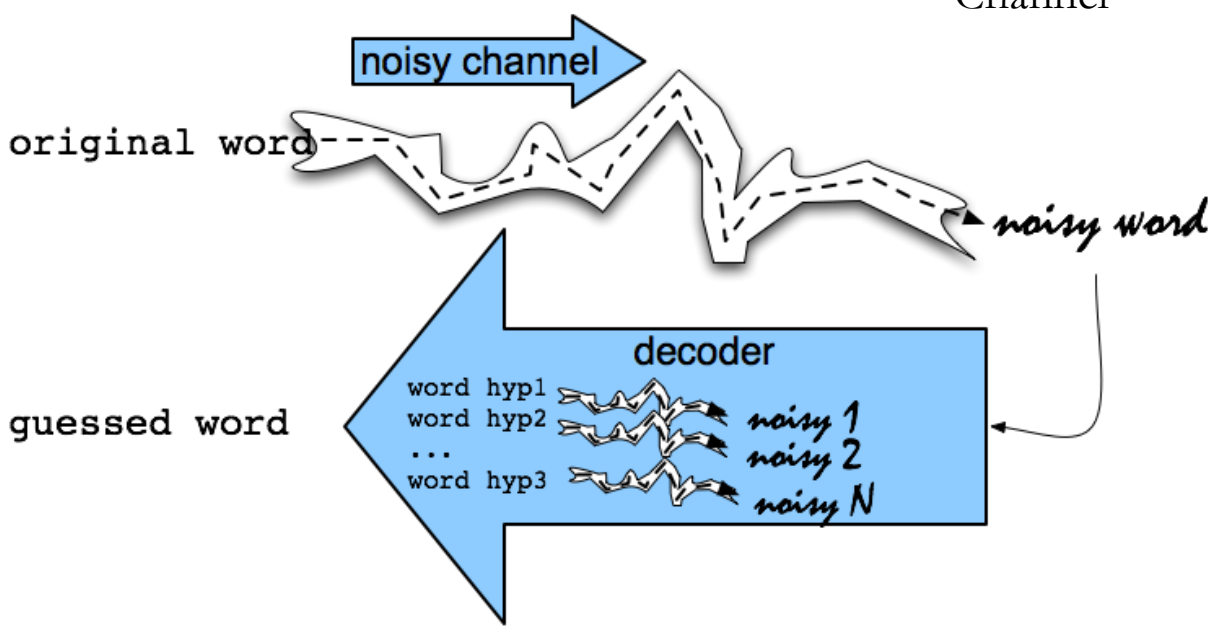
Spelling Correction: Noisy Channel Model

- We see an observation x of a misspelled word
- Find the correct word w

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x) \quad = \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)} \quad = \underset{w \in V}{\operatorname{argmax}} P(x \mid w)P(w)$$

Channel

Language Model: N-Gram



Spelling Correction

- Example:

Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210

- acress

- Candidate Generation:
  - Words with similar spelling
- Damerau-Levenshtein edit distance:
  - Insertion: cress
  - Deletion: actress
  - Substitution: caress
  - Transposition of two adjacent letters: arcoss

Candidate Generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
  - `thisidea` → `this idea`
  - `inlaw` → `in-law`

Candidate Generation

- Words within 1 edit of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion

Language Model

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

Channel Model

- Edit probability: P(x|w)
  - *Kernighan, Church, Gale 1990*
- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$
- Spelling errors tend to be character level
- studeid, freind
- pronuon, ruote

Channel Model

- Edit probability: P(x|w)
  - *Kernighan, Church, Gale 1990*
- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$
  - (deletion/insertion/substitution/transposition)
  - Insertion and deletion conditioned on previous character

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(x typed as y)
trans[x,y]:  count(xy typed as yx)
```

Confusion Matrix

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Channel Model

$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Kernighan, Church, Gale 1990

Noisy Channel Model for "acress"

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
actress	t	–	c|ct	.000117
cress	–	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	–	s	es|e	.0000321

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	–	c|ct	.000117	.0000231	2.7
cress	–	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	o	e	e|o	.0000093	.000299	2.8
acres	–	s	es|e	.0000321	.0000318	1.0

Using Bigram for "acress"

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021
  P(whose|actress) = .0010
- P(across|versatile) =.000021
  P(whose|across) = .000006

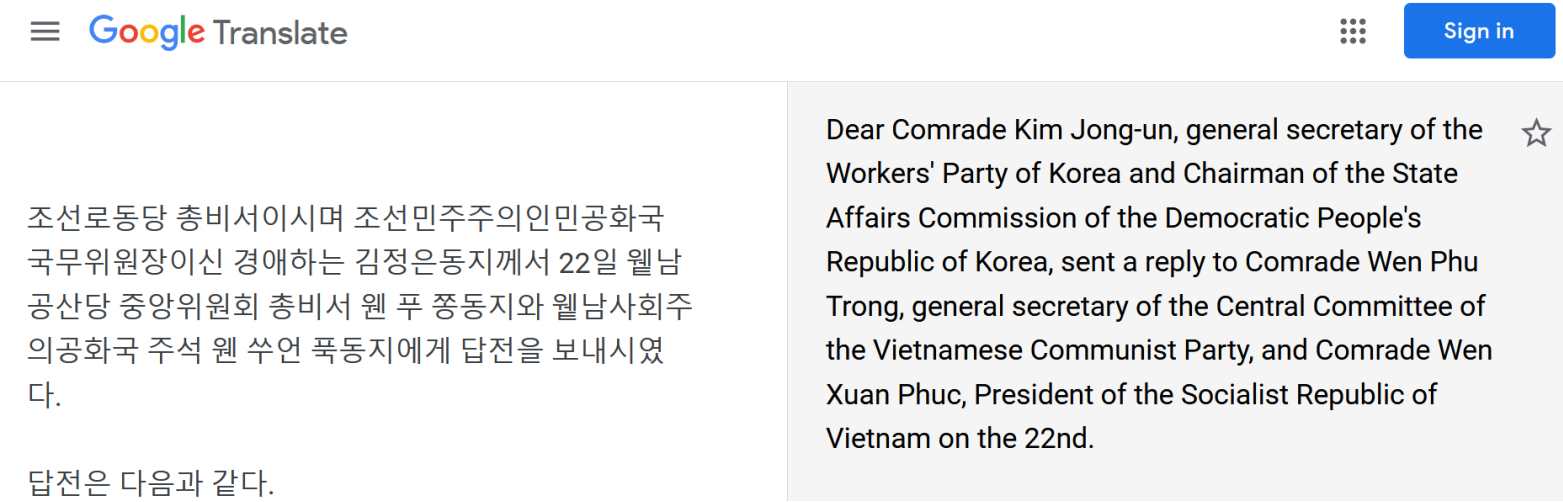
- P("versatile actress whose") = .000021*.0010 = 210 x10^{-10}
- P("versatile across whose") = .000021*.000006 = 1 x10^{-10}

Spelling Correction: Evaluation

- Some spelling error test sets
  - Wikipedia's list of common English misspelling
  - Aspell filtered version of that list
  - Birkbeck spelling error corpus
  - Peter Norvig's list of errors

Machine Translation

- **Translation: a very challenging task in general**
- poetry
- old text
- professional text
- **Machine Translation**
- Information access



- Computer-aided translation: draft translation
- Communication

MT Challenges

- Lexical ambiguity:

- River Bank -> ساحل رودخانه
- Bank Account -> حساب بانکی

- Word order may be different

- I bought a book

1 2 3 4

من یک کتاب خریدم

2 4 3 1

- English order is SVO but Persian is SOV

- Syntactic Structure may not be preserved

پسرک تلفنش را برداشت و با دوستش تماس گرفت

The little boy grabbed his phone and called his friend

MT Challenges

- Cross-language Lexical ambiguity

Important

1. مهم 2. با اهمیت 3. عمده 4. خطیر 5. پراهمیت 6. ژرف 7. فوق العاده

- Syntactic ambiguity

I'm glad I'm healthy, and so is my child.

خوشحال هستم که سلامت هستم و همین طور خوشحالم که فرزندم سلامت است.

خوشحال هستم که سلامت هستم و همین طور فرزندم خوشحال است که من سلامت هستم

- Pronoun Resolution

او آمد

She/He came

Rule-Based Machine Translation

- The basis is on word-by-word translation
- No syntactic or semantic analysis is performed on the source language to resolve potential ambiguities
- We rely on a very large bilingual dictionary that allows for translating all words
- After translating the words, we use rule-based NLP to arrange the word order

Rule-Based Machine Translation

- Example: Machine translation and human being (Panov 1960s)

- Rules for translating much or many into Russian:

```
if preceding word is how return skol'ko
else if preceding word is as return stol'ko zhe
else if word is much
    if preceding word is very return nil
    else if following word is a noun return mnogo
else (word is many)
if preceding word is a preposition and following word is noun
    return mnogii
    else return mnogo
```

Challenges for Rule-Based Machine Translation

- Reordering words based on rules becomes very challenging for long sentences

یک کتاب آبی بزرگ پایه پنجم ابتدایی را تحویل گرفتم و به دانش آموز دادم

I got a big blue fifth grade elementary school and gave it to the student

- Lexical ambiguity may make the translation unmeaningful

I told him that he should study hard

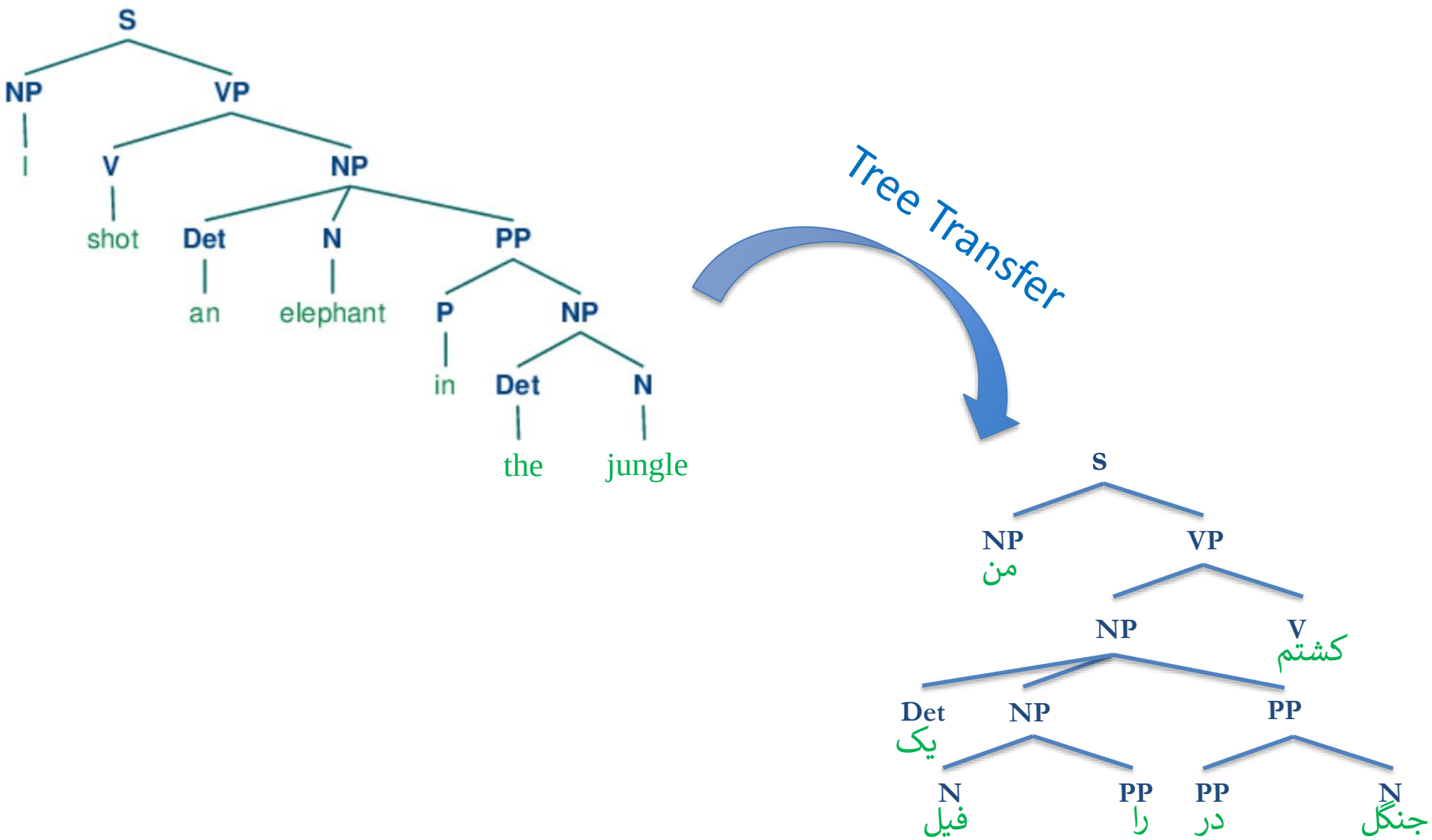
من به او گفتم که آن او باید مطالعه کند

Transfer-Based Machine Translation

- Improve Rule-based MT:
- **Source analysis:** Analyze the source language: build a syntactic model for the source text
- **Transfer:** Convert the source-language parse tree to a target-language parse tree.
- **Generation:** Convert the target-language parse tree to an output sentence.

- Transfer stage is still rule-based but rules on syntactic structures are more generalizable, e.g., POS plus a set of rules based on the grammar of the source and the target language (SOV vs SVO)
- Long sentence reordering is easier
- SYSTRAN (Founded by Peter Toma in 1968) systems are based on this approach

Transfer-Based Machine Translation

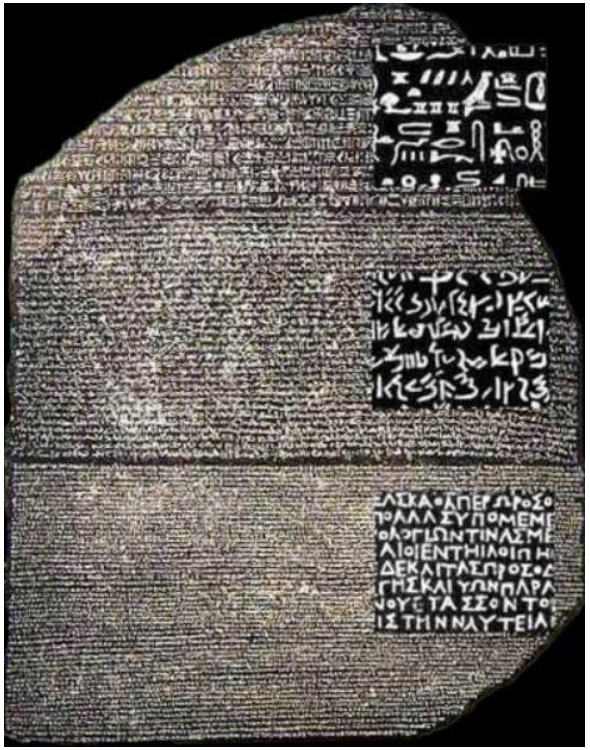


Statistical Machine Translation

- Idea: parallel corpora are available in several language pairs

- we can use a parallel corpus as a training set of translation examples and train a model to translate to relax the need for rules

- Old idea: Rosetta Stone



Statistical Machine Translation

- Parallel Texts:
  - Canadian Hansards: French-English (1.7 million sentences of 30 words or less in length), used by IBM
  - European Union
  - Translations of books
- MT Model:
  - Train a model that receives a sentence in the source language and returns the sentence in the target language
  - Use the model when the output is unknown
- IBM Models: 90s

The Noisy Channel Model for MT

- Goal: translate from French to English
- Generate a model p(e | f) which estimates conditional probability of any English sentence given the French sentence f.
- Use the training corpus to set the parameters.

- Noisy channel Model:

Language Model Translation Model

$$p(e \mid f) = \frac{p(e, f)}{p(f)} = \frac{p(e)p(f \mid e)}{\sum_e p(e)p(f \mid e)}$$

Decoding Problem

$$\text{argmax}_e p(e \mid f) = \text{argmax}_e p(e)p(f \mid e)$$

The Noisy Channel Model

- We can use a trigram as the language model

- It can be estimated using a larger English corpus

- The translation model is trained using the parallel corpus
- Example (from tutorial by Koehn and Knight)

- Translation from Spanish to English

Que hambre tengo yo
$\rightarrow$

What hunger have	$p(s\|e) = 0.000014$
Hungry I am so	$p(s\|e) = 0.000001$
I am so hungry	$p(s\|e) = 0.0000015$
Have i that hunger	$p(s\|e) = 0.000020$

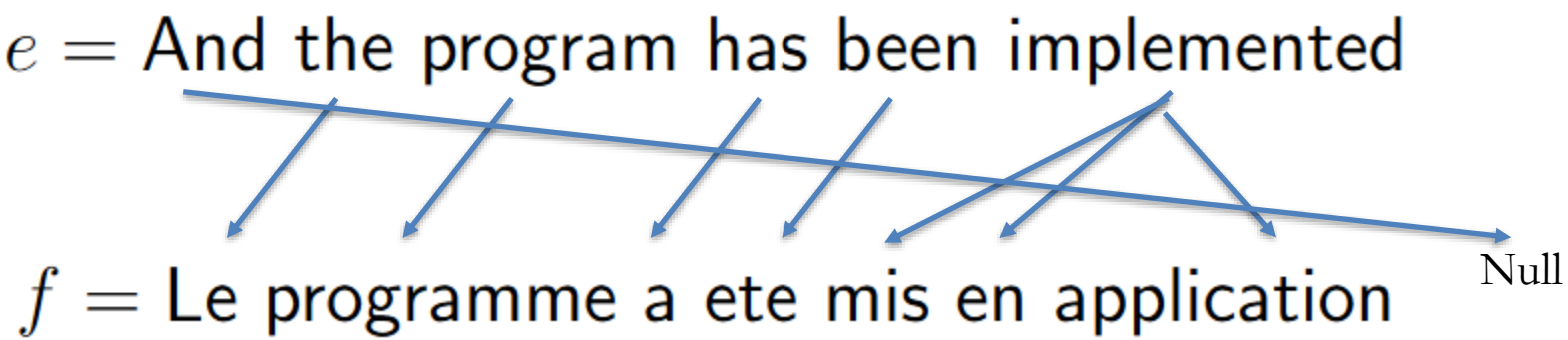
...

Que hambre tengo yo
$\rightarrow$

What hunger have	$p(s\|e)p(e) = 0.000014 \times 0.000001$
Hungry I am so	$p(s\|e)p(e) = 0.000001 \times 0.0000014$
I am so hungry	$p(s\|e)p(e) = 0.0000015 \times 0.0001$
Have i that hunger	$p(s\|e)p(e) = 0.000020 \times 0.00000098$

Translation Model: IBM Model

- How do we model the translation model?
- In the parallel corpus, consider that for a pair, the English sentence has l words and the French sentence has m words
- An alignment map determines which English word each French word originated from
- An alignment a is $\{a_1, \ldots a_m\}$, where $a_j \in \{0 \ldots l\}$
- Hence there are $(l+1)^m$ possible alignments
- Ex: {2,3,4,5,6,6,6}



Translation Model

- Total probability over all possible alignments

Alignment Distribution Conditional Translation Model

$$p(f \mid e, m) = \sum_{a \in \mathcal{A}} \underbrace{p(a \mid e, m) p(f \mid a, e, m)}_{p(f, a \mid e, m)}$$

- We will model the conditional probabilities:

$p(a \mid e, m)$ and $p(f \mid a, e, m)$

$$p(f, a \mid e, m) = p(a \mid e, m) p(f \mid a, e, m)$$

$$p(f \mid e, m) = \sum_{a \in \mathcal{A}} p(a \mid e, m) p(f \mid a, e, m)$$

- Having computed the conditional probabilities:

$$p(a \mid f, e, m) = \frac{p(f, a \mid e, m)}{\sum_{a \in \mathcal{A}} p(f, a \mid e, m)}$$

$p(f \mid e, m)$

Most Likely Alignment

$$a^* = \arg\max_a p(a \mid f, e, m)$$

IBM Model 1

- Equally likely Alignment Probability

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

- Conditional Translation Model

Model Parameters

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

- Ex: $l = 6,\ m = 7$ $\quad a = \{2, 3, 4, 5, 6, 6, 6\}$

$e =$ And the program has been implemented

$f =$ Le programme a ete mis en application

$$p(f \mid a, e) = t(Le \mid the) \times t(programme \mid program) \times t(a \mid has) \times t(ete \mid been) \times t(mis \mid implemented) \times t(en \mid implemented) \times t(application \mid implemented)$$

IBM Model 1

- An example of model parameters

English	French	Probability
position	position	0.756715
position	situation	0.0547918
position	mesure	0.0281663
position	vue	0.0169303
position	point	0.0124795
position	attitude	0.0108907

IBM Model 1: Generative Process

- Pick an alignment randomly: $\frac{1}{(l+1)^m}$

- Pick the corresponding French words

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

- Compute the conditional translation probability

$$p(f, a \mid e, m) = p(a \mid e, m) \times p(f \mid a, e, m) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

IBM Model 2

- Non-uniform alignments: distortion parameters

Model 1

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

Model 2

$$p(a \mid e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j = i \mid j, l, m)$$

j's French word is generates from i's English word given the lengths

- Conditional Translation Model

Model 2

$$p(f, a \mid e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m)\mathbf{t}(f_j \mid e_{a_j})$$

Model 1

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

IBM Model 2

- Example

$$
\begin{aligned}
l &= 6 \\
m &= 7 \\
e &= \text{And the program has been implemented} \\
f &= \text{Le programme a ete mis en application} \\
a &= \{2, 3, 4, 5, 6, 6, 6\}
\end{aligned}
$$

$$
\begin{aligned}
p(a \mid e, 7) = {} & \mathbf{q}(2 \mid 1, 6, 7) \times \mathbf{q}(3 \mid 2, 6, 7) \times \mathbf{q}(4 \mid 3, 6, 7) \times \\
& \mathbf{q}(5 \mid 4, 6, 7) \times \mathbf{q}(6 \mid 5, 6, 7) \times \mathbf{q}(6 \mid 6, 6, 7) \times \mathbf{q}(6 \mid 7, 6, 7)
\end{aligned}
$$

$$
\begin{aligned}
p(f \mid a, e, 7) = {} & t(Le \mid the) \times t(programme \mid program) \times t(a \mid has) \times t(ete \mid been) \times \\
& t(mis \mid implemented) \times t(en \mid implemented) \times t(application \mid implemented)
\end{aligned}
$$

IBM Model 2: Generative Process

- Pick an alignment randomly: $\prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m)$

- Pick the corresponding French words

$$p(f \mid a, e, m) = \prod_{j=1}^{m} \mathbf{t}(f_j \mid e_{a_j})$$

- Compute the conditional translation probability

$$p(f, a \mid e, m) = p(a \mid e, m) p(f \mid a, e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m) \mathbf{t}(f_j \mid e_{a_j})$$

IBM Model Parameter Estimation

- Input: sentence pairs $(e^{(k)}, f^{(k)})$
- Output: parameters $t(f|e)$ and $q(i|j,l,m)$
- Primary Challenge: alignments are not known

- Data annotation is expensive

$$
\begin{aligned}
e^{(100)} &= \text{And the program has been implemented} \\
f^{(100)} &= \text{Le programme a ete mis en application}
\end{aligned}
$$

- Expectation Maximization (EM) algorithm

IBM Model Parameter Estimation

- Assume the alignments are accessible

$$
\begin{aligned}
e^{(100)} &= \text{And the program has been implemented} \\
f^{(100)} &= \text{Le programme a ete mis en application} \\
a^{(100)} &= \langle 2, 3, 4, 5, 6, 6, 6 \rangle
\end{aligned}
$$

- We will have triplets $(e^{(k)}, f^{(k)}, a^{(k)})$
- ML estimates for parameters boils down to counting, ex, t(position|position)

$$t_{ML}(f|e) = \frac{\text{Count}(e,\ f)}{\text{Count}(e)} \qquad q_{ML}(j|i,l,m) = \frac{\text{Count}(j,i,l,m)}{\text{Count}(i,l,m)}$$

IBM Model Parameter Estimation

Ex:
e= the position
f=La position
a = {1,2}

Input: A training corpus $(f^{(k)}, e^{(k)}, a^{(k)})$ for $k = 1 \ldots n$, where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$, $a^{(k)} = a_1^{(k)} \ldots a_{m_k}^{(k)}$.

Algorithm:

- Set all counts $c(\ldots) = 0$
- For $k = 1 \ldots n$
  - For $i = 1 \ldots m_k$, For $j = 0 \ldots l_k$,

$$
\begin{aligned}
c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j) \\
c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k,i,j) \\
c(j|i,l,m) &\leftarrow c(j|i,l,m) + \delta(k,i,j) \\
c(i,l,m) &\leftarrow c(i,l,m) + \delta(k,i,j)
\end{aligned}
$$

English Position

French Position

where $\delta(k,i,j) = 1$ if $\boxed{a_i^{(k)} = j,\ 0}$ otherwise.

Pair Index

Output: $t_{ML}(f|e) = \frac{c(e,f)}{c(e)}$, $q_{ML}(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$

Expectation Maximization

- Dempster et al., 1977: An algorithm for computing maximum likelihood from incomplete data:

- incomplete data means that data annotation is not complete to allow for estimating the model paramters
- if we had complete data, would could estimate model
- if we had model, we could fill in the gaps in the data

- EM in a nutshell:

1. initialize model parameters, e.g., random
2. assign probabilities to the missing data
3. estimate model parameters from completed data
4. iterate steps 2–3 until convergence

EM Algorithm for MT

- We don't have the alignments:

1. The algorithm is **iterative**: we start with some arbitrary random choice for the q and t parameters. At each iteration we compute the "counts" based on the data together with our current parameter estimates. We then re-estimate the parameters with these counts, and iterate
2. $\delta(k, i, j)$ is computed as follows

$$\delta(k, i, j) = \frac{q(j|i, l_k, m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i, l_k, m_k)t(f_i^{(k)}|e_j^{(k)})}$$

EM Algorithm for MT

- S ~ 10-20

For $s = 1 \ldots S$

- Set all counts $c(\ldots) = 0$
- For $k = 1 \ldots n$
  - For $i = 1 \ldots m_k$, For $j = 0 \ldots l_k$

- Delta parameters:

$$\delta(k,i,j) = P(a_j^{(k)} = i | e^{(k)}, f^{(k)})$$

M-Step

$$\begin{aligned} c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j) \\ c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k,i,j) \\ c(j|i,l,m) &\leftarrow c(j|i,l,m) + \delta(k,i,j) \\ c(i,l,m) &\leftarrow c(i,l,m) + \delta(k,i,j) \end{aligned}$$

where

E-Step

$$\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}$$

- EM would converge to local ML optimums

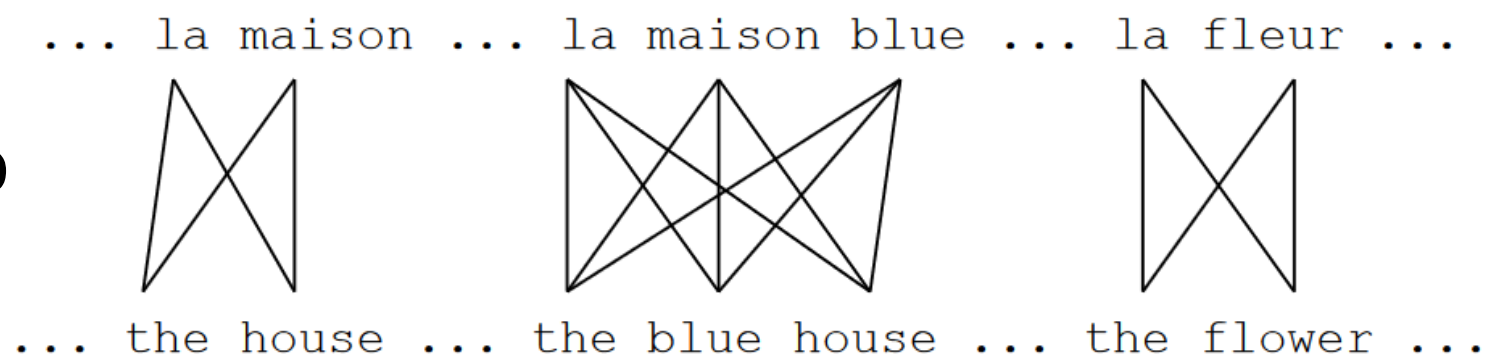
- Recalculate the parameters:

$$t(f|e) = \frac{c(e,f)}{c(e)} \qquad q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$$

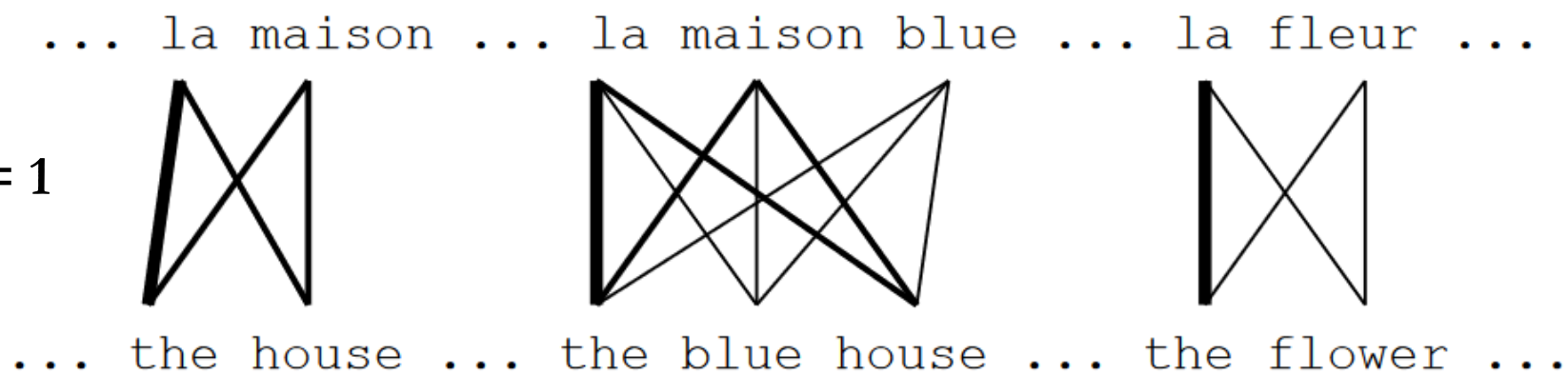
EM Algorithm for MT

- Initialization: set all assignments equally likely
- Model learns ‘La’ is often aligned with ‘the’

Iter = 0



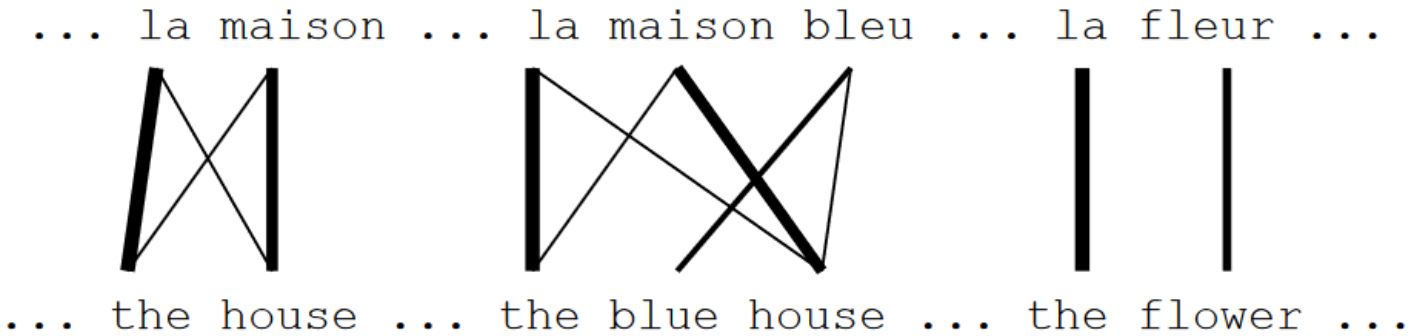
Iter = 1



EM Algorithm for MT

- After one more iteration `fleur' is aligned with 'flower'
- Convergence: after One more iteration

Iter = 2



Iter = 1

Model Evaluation

- Perplexity: derived from probability of the training data according to the model

$$\log_2 PP = -\sum_s \log_2 p(\mathbf{e}_s|\mathbf{f}_s)$$

- Ex:

	initial	1st it.	2nd it.	3rd it.	...	final
p(the haus|das haus)	0.0625	0.1875	0.1905	0.1913	...	0.1875
p(the book|das buch)	0.0625	0.1406	0.1790	0.2075	...	0.25
p(a book|ein buch)	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8

CSCI 544, Lecture 15: Language models and speech recognition

Ron Artstein

2022-10-11

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for watching the lecture video, taking notes, and participating in class discussions. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes

No class on October 13

Coding Assignment 3 was due today

- Late sumbission by tomorrow with 10% penalty

Article selection for presentation due October 18

Coding Assignment 4 due October 25

Project:

Due Date	Task
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Article selection

- Each group selects an article for presentation.
- Select from top-tier NLP venues of 2022: ACL, NAACL, TACL, Findings of ACL, or Findings of NAACL.
- Put title and link on shared spreadsheet; check for conflicts.
  - Sheet accessible to USC Google accounts.
- Article does not need to be related to project.
- Pick article with interesting theme, point, or result.
  - If you cannot identify something interesting, choose a different article.
- Presentation does not have to cover the entire article.
- Presentation aimed for audience with the background covered in class.

Coding assignment 4

Write a Perceptron classifier

From scratch!

Corpus of hotel reviews

- Two classification problems: true/fake and positive/negative
- Two models: vanilla and averaged

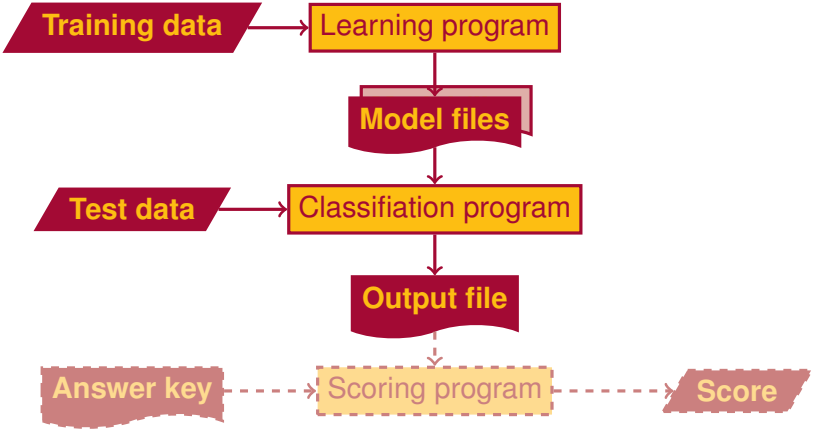
Graded on performance

Programming in Python

Submit on Vocareum

- Automatic feedback
- Submit early, submit often!

Coding assignment 4: programs



Coding assignment 4: notes

Problem formulation

Two binary classifiers: two sets of parameters

Features and tokenization

Experiment to see what works best!

. . . after you get the basic program working

Runtime efficiency

Compact representation

Don't multiply zeros

Over/underfitting

Choose number of iterations

Probability of a sentence

Does it make sense to talk about the probability of a sentence?

> or 'situations'. But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term. On empirical grounds, the probability of my producing some given sentence of English – say, this sentence, or the sentence "birds fly" or "Tuesday follows Monday", or whatever – is indistinguishable from the probability of my producing a given sentence of Japanese.

Noam Chomsky. Quine's empirical assumptions. *Synthese* 19(1/2): 53–68, 1968.

Language models

Probability of a sentence/utterance $\approx$
Probability of word given previous words

Unigram language model (bag of words): Independence assumption (similar to naïve Bayes)

$$P(w_1 \ldots w_n) \approx \prod_i P(w_i) = P(w_1) \cdot P(w_2) \cdots P(w_n)$$

Bigram language model: Markov assumption

$$P(w_1 \ldots w_n) \approx \prod_{i=0}^{n} P(w_{i+1}|w_i) = P(w_1|\#) \cdot P(w_2|w_1) \cdots P(\#|w_n)$$

N-gram language models

Trigram, 4-gram etc. use probabilities for longer word sequences

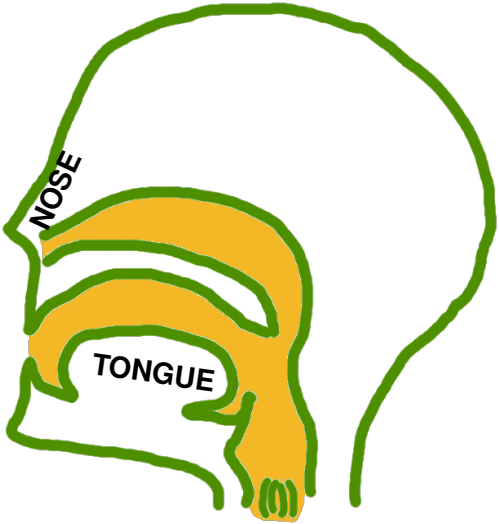
Some smoothing methods:

- Backoff: Use probability of highest available n-gram
- Interpolation: weighted mixture of trigram, bigram, unigram probabilities
- Etc.

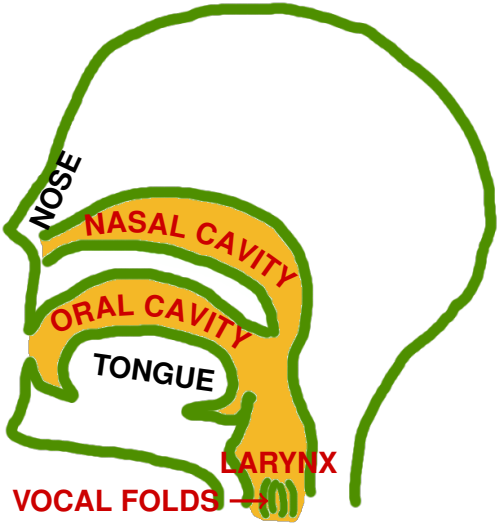
Language models can give a **prior** probability on sentences

Human vocal tract (and some phonetics)

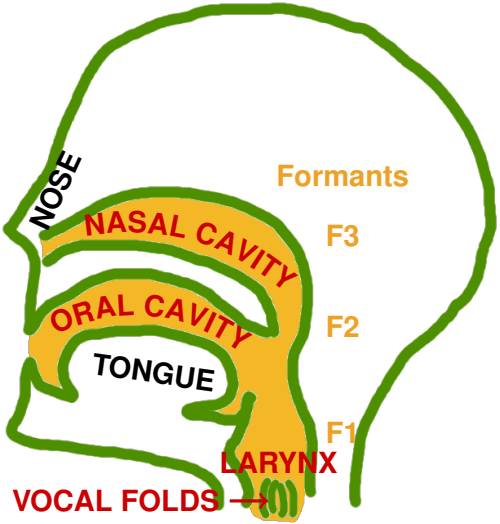
Human vocal tract (and some phonetics)

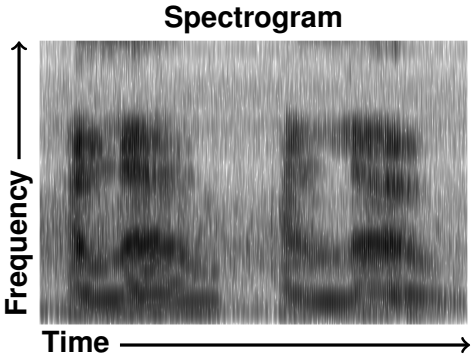


Human vocal tract (and some phonetics)

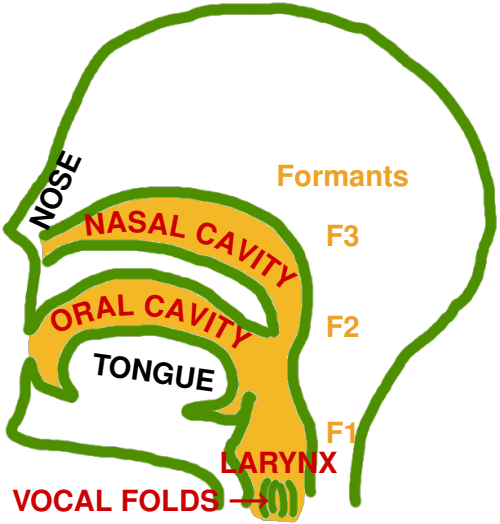


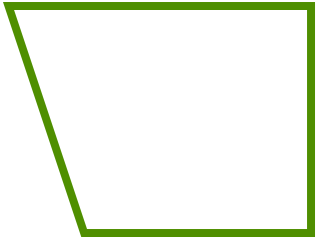
Human vocal tract (and some phonetics)



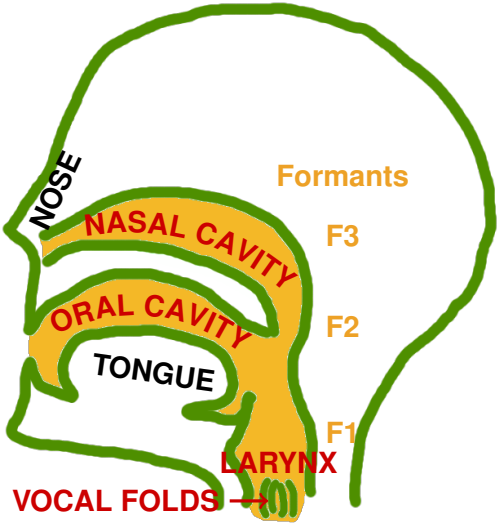


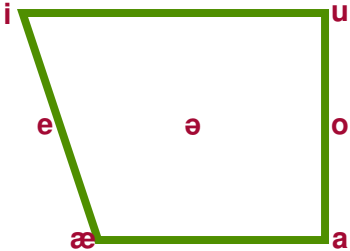
Human vocal tract (and some phonetics)



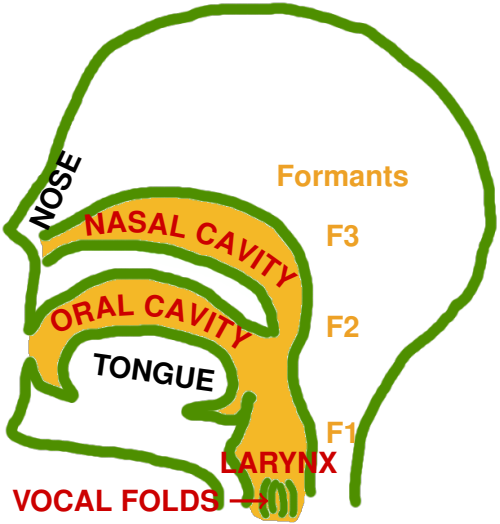


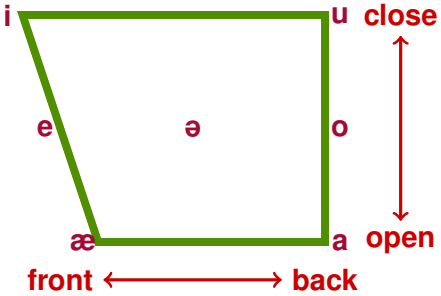
Human vocal tract (and some phonetics)



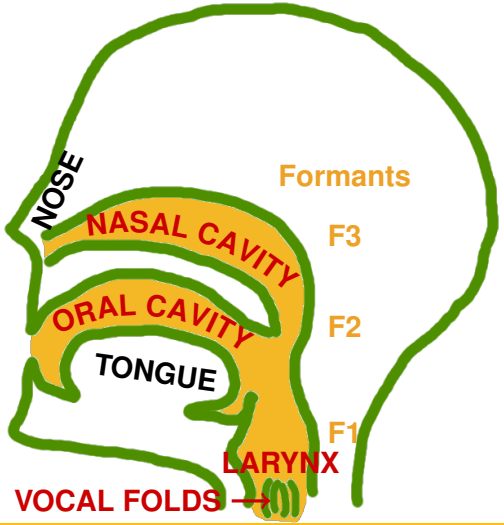


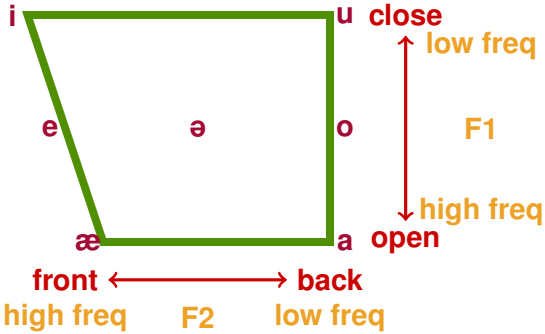
Human vocal tract (and some phonetics)





Human vocal tract (and some phonetics)





Speech recognition basics

Sample the waveform (e.g., every 10 msec)

Extract features

- Mel Frequency Cepstral Coefficients: MFCC
- Mel scale: pitch $\rightarrow$ perceptual pitch (works better)

Calculate $P(\text{word}|\text{sound})$

[word string] [time series of acoustic vectors]

Easier to calculate $P(\text{sound}|\text{word})$;
need prior distribution on word strings

$$P(\text{word}|\text{sound}) \propto \underbrace{P(\text{sound}|\text{word})}_{\text{[acoustic model]}} \cdot \underbrace{P(\text{word})}_{\text{[language model]}}$$

Words are too big...

Predict acoustic model from shorter units of sound: **phones**

	feeling					
IPA	f	i	l	ɪ	ŋ	(One character per phone)
ARPABET	F	IY	L	IH	NG	(Phones separated by spaces)

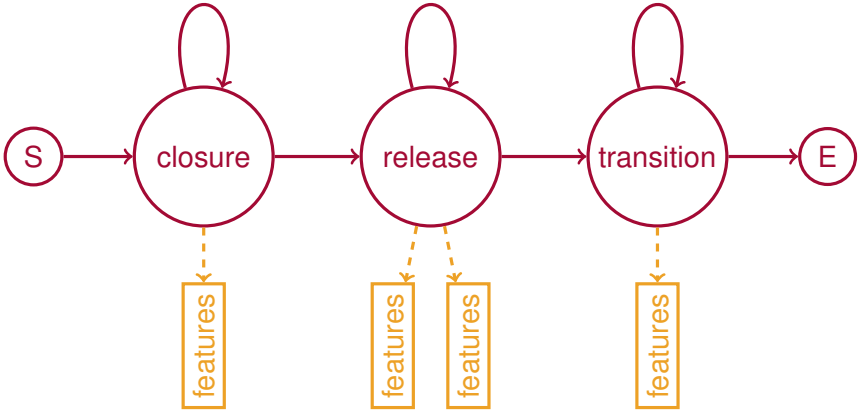
Vowels and voiced sounds have formants throughout

Stops are silent, affect formants before closure and after release

Phones modeled as 3-state HMMs: closure, release, transition

- States, transitions are discrete
- Observations are continuous:
  $P(\text{observation}|\text{state}) = \text{Normal}(\text{mean}, \text{variance})$

HMMs for phone recognition



Decoding speech

Find the most likely sequence of words, given a sequence of feature vectors

- Transitions *within* phones are for structure
- Transitions *between* phones constrained by language model and lexicon

Search through a lattice of hypotheses, with pruning (e.g. Viterbi decoding, beam search)

Speech recognition basics (review)

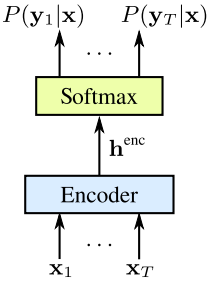
$$P(\text{word}|\text{sound}) \propto \underbrace{P(\text{sound}|\text{word})}_{\text{[acoustic model]}} \cdot \underbrace{P(\text{word})}_{\text{[language model]}}$$

- Acoustic model conditioned on phones
- Language model = distribution on words
- Phones and words linked by dictionary (lexicon)

Splitting the problem into acoustic and language models

- Acoustic model may vary by speaker
- Language model may vary by domain

Neural acoustic modeling: CTC



(a.) CTC

Graves et al., ICML 2006

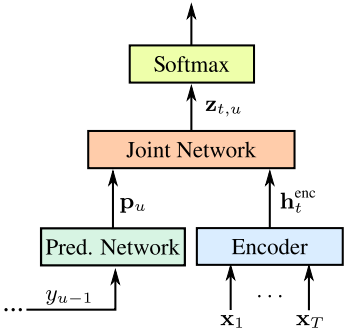
- Acoustic feature vectors → transcription symbols
- Output includes blank symbol
- ✔ Works well for phonetic transcription

From phonetic symbols to words: dictionary + LM

Prabhavalkar et al., Interspeech 2017

- ✘ Mapping acoustic feature vectors directly to characters doesn't work so well

Neural language modeling: RNN-Transducer



(b.) RNN-Transducer

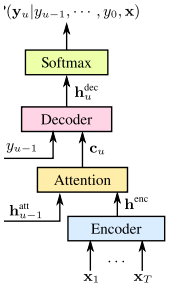
Graves, ICML workshop 2012

- Prediction network: dependencies between output symbols
- Analogous to a language model
- ✔ Predict phonetic transcription

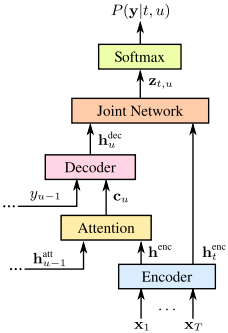
Prabhavalkar et al., Interspeech 2017

- ✔ Mapping acoustic vectors to characters
- **Streaming** with unidirectional networks

Neural modeling: attention



c.) Attention-based Model



(d.) RNN-Transducer with Attention

Chan et al., ICASSP 2016

- End-to-end: Listen-attend-spell

Attention: non-local dependencies

Prabhavalkar et al., Interspeech 2017

✔ Acoustic vectors to characters

✘ Difficulty with streaming

Chiu et al., ICASSP 2018

- Wordpieces instead of characters
- Other oprimizations

Evaluating speech recognizers

Word Error Rate: Levenshtein edit distance between reference and hypothesis

$$\text{WER} = \frac{\text{Deletions} + \text{Insertions} + \text{Substitutions}}{\text{Length of reference string}}$$

Ref: She is going to the beach
Hyp: She is **gone** to **am** the beach

$$\text{WER} = \frac{2}{6} = 33.3\%$$

Ref: I'm
Hyp: I am

$$\text{WER} = \frac{2}{1} = 200\%$$

Other measures: Concept Error Rate (what's a concept?)
Morpheme Error Rate
Phone Error Rate

Phones can match better than words

Speech:	Are you **married**
ASR:	Are you **Mary**

Word Error Rate
1/3 = 0.33

Speech:	AA R Y UW M EH R IY **D**
ASR:	AA R Y UW M EH R IY

Phone Error Rate
1/9 = 0.11

Wang, Artstein, Leuski, Traum
FLAIRS 2011

Speech:	Are **all** **soldiers deployed**
ASR:	Are **also just** **avoid**

Word Error Rate
3/4 = 0.75

AA R AO L S OW **L** JH **ER Z D IH P L** OY D	
AA R AO L S OW JH **IH S T AH V** OY D	

Phone Error Rate
7/16 = 0.44

CSCI 544, Lecture 16: Grammars and statistical parsing

Ron Artstein

2022-10-18

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for watching the lecture video, taking notes, and participating in class discussions. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes

Coding Assignment 3 will be graded soon

Coding Assignment 4 due October 25

- Submit early, submit often

Presentation:

Due Date	Task
October 18	**Article selection**
October 27	Presentation slides
November 1–10	Presentations

Project:

Due Date	Task
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Article presentations

7 minutes, 7 slides

- Practice to fit in time slot
- Invlove entire team

On Zoom
Sign-up sheet available

Use slides, but talk about research, not slides

The audience has not read the article

- Assume no prior knowledge of the topic
- Explain new methods or techniques

Concentrate on **main idea**

- Explain necessary details, but don't get bogged down
- Explain the linguistic aspects

Include **your own** critical evaluation

Syntax

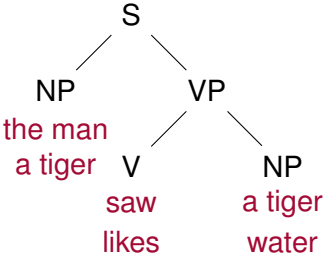
Structure of sentences

Context-Free Grammar (one grammatical formalism)

Context-free rules

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$



Other grammar formalisms

Regular grammars (not very expressive)

- Cannot capture center-embedding dependencies

a a a b b b ☛ Captured by context-free grammars

Tree-Adjoining Grammars
Combinatory Categorial Grammars } "mildly context sensitive"

- Can capture cross-serial dependencies

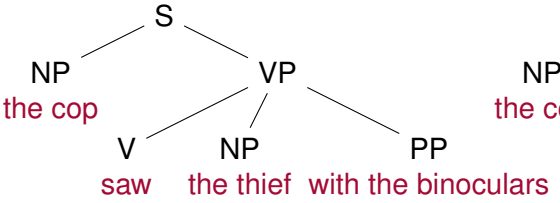
a a a b b b ☛ Is human language context-free?

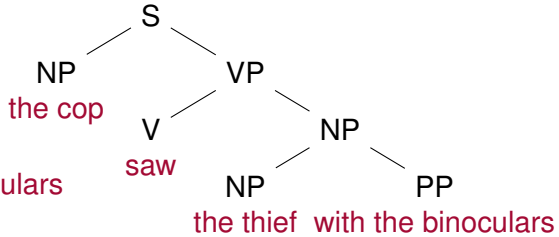
Dependency grammars

Ambiguity

A context-free grammar can produce ambiguous sentences:

The cop saw the thief with the binoculars **attachment ambiguity**





The parsing problem

1. Find possible structures according to the grammar
2. Find the most likely parse

☛ Charniak argues against **linguistic** grammaticality, not against formal grammaticality

Chart parsing (Earley, CYK, etc.): $O(n^3)$

CYK algorithm: bottom-up

- Chomsky Normal Form: all rules of form X $\rightarrow$ Y Z or X $\rightarrow$ word

CYK parsing example

$S \rightarrow NP\ VP$	$NP \rightarrow$ United
$VP \rightarrow VP\ NP$	$NP \rightarrow$ flights
$VP \rightarrow VP\ PP$	$NP \rightarrow$ Houston
$NP \rightarrow NP\ PP$	$VP \rightarrow$ diverted
$PP \rightarrow P\ NP$	$P \rightarrow$ to

United diverted flights to Houston

CYK parsing example

S $\rightarrow$ NP VP	NP $\rightarrow$ United
VP $\rightarrow$ VP NP	NP $\rightarrow$ flights
VP $\rightarrow$ VP PP	NP $\rightarrow$ Houston
NP $\rightarrow$ NP PP	VP $\rightarrow$ diverted
PP $\rightarrow$ P NP	P $\rightarrow$ to

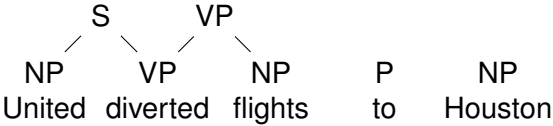
NP	VP	NP	P	NP
United	diverted	flights	to	Houston

CYK parsing example

$S \rightarrow NP\ VP$	$NP \rightarrow$ United
$VP \rightarrow VP\ NP$	$NP \rightarrow$ flights
$VP \rightarrow VP\ PP$	$NP \rightarrow$ Houston
$NP \rightarrow NP\ PP$	$VP \rightarrow$ diverted
$PP \rightarrow P\ NP$	$P \rightarrow$ to

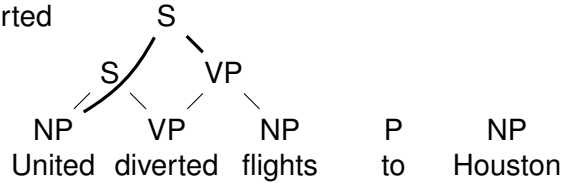
CYK parsing example

S $\rightarrow$ NP VP	NP $\rightarrow$ United
VP $\rightarrow$ VP NP	NP $\rightarrow$ flights
VP $\rightarrow$ VP PP	NP $\rightarrow$ Houston
NP $\rightarrow$ NP PP	VP $\rightarrow$ diverted
PP $\rightarrow$ P NP	P $\rightarrow$ to



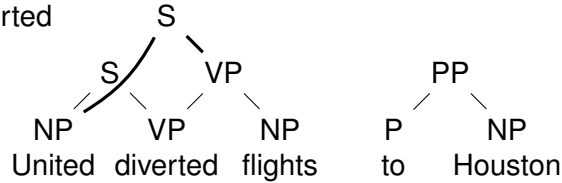
CYK parsing example

S → NP VP	NP → United
VP → VP NP	NP → flights
VP → VP PP	NP → Houston
NP → NP PP	VP → diverted
PP → P NP	P → to



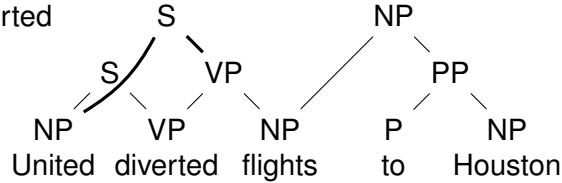
CYK parsing example

S → NP VP	NP → United
VP → VP NP	NP → flights
VP → VP PP	NP → Houston
NP → NP PP	VP → diverted
PP → P NP	P → to



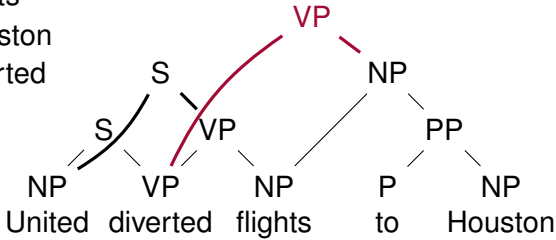
CYK parsing example

S → NP VP	NP → United
VP → VP NP	NP → flights
VP → VP PP	NP → Houston
NP → NP PP	VP → diverted
PP → P NP	P → to



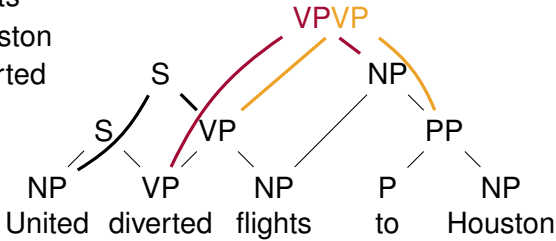
CYK parsing example

S → NP VP	NP → United
VP → VP NP	NP → flights
VP → VP PP	NP → Houston
NP → NP PP	VP → diverted
PP → P NP	P → to



CYK parsing example

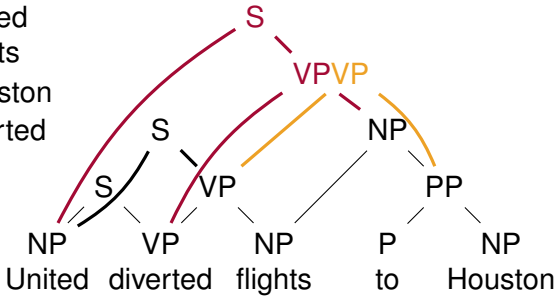
S → NP VP	NP → United
VP → VP NP	NP → flights
VP → VP PP	NP → Houston
NP → NP PP	VP → diverted
PP → P NP	P → to



CYK parsing example

$S \rightarrow NP\ VP$
$VP \rightarrow VP\ NP$
$VP \rightarrow VP\ PP$
$NP \rightarrow NP\ PP$
$PP \rightarrow P\ NP$

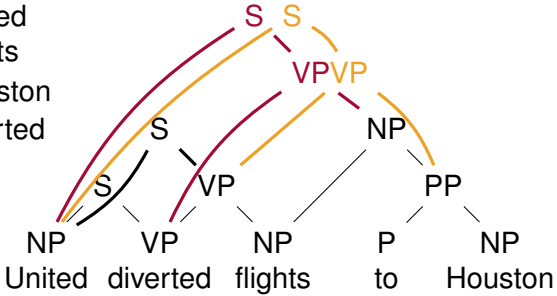
$NP \rightarrow$ United
$NP \rightarrow$ flights
$NP \rightarrow$ Houston
$VP \rightarrow$ diverted
$P \rightarrow$ to



CYK parsing example

$S \rightarrow NP\ VP$
$VP \rightarrow VP\ NP$
$VP \rightarrow VP\ PP$
$NP \rightarrow NP\ PP$
$PP \rightarrow P\ NP$

$NP \rightarrow$ United
$NP \rightarrow$ flights
$NP \rightarrow$ Houston
$VP \rightarrow$ diverted
$P \rightarrow$ to



Bottom-up or top-down?

CYK is a **bottom-up algorithm**: it finds potential constituents that can never fit into the full parse of the sentence

A **top-down algorithm** only predicts constituents that fit. . .
. . . but it predicts constituents that don't match the data

At the end of class today (if we have time):
top-down agenda-based algorithm (bottom-up version also exists)

Statistical parsing

How do we find the most likely parse?

- United $[_{VP}$ $[_{VP}$ diverted flights$]$ $[_{PP}$ to Houston$]]$
- United $[_{VP}$ diverted $[_{NP}$ flights $[_{PP}$ to Houston$]]]$

Probabilistic CFG: probabilities for rules

$P(S \rightarrow NP\ VP)$ $P(VP \rightarrow VP\ NP)$

$P(VP \rightarrow VP\ PP)$ $P(NP \rightarrow NP\ PP)$

Words matter too: I ate the animal crackers

I fed the animal crackers

More detailed symbols in the grammar: $V_{transitive} \rightarrow$ ate

$V_{ditransitive} \rightarrow$ fed

Or go fully lexical

Lexicalized PCFG

I ate the animal crackers | **I fed the animal crackers**

P(rule | head) **A superscript $^{(h)}$ indicates the head constituent**

$P(\text{VP} \rightarrow \text{VP}^{(h)}\ \text{NP} \mid \text{VP}^{(h)} = \text{ate})$

$P(\text{VP} \rightarrow \text{VP}^{(h)}\ \text{NP NP} \mid \text{VP}^{(h)} = \text{ate})$

$P(\text{VP} \rightarrow \text{VP}^{(h)}\ \text{NP} \mid \text{VP}^{(h)} = \text{fed})$

$P(\text{VP} \rightarrow \text{VP}^{(h)}\ \text{NP NP} \mid \text{VP}^{(h)} = \text{fed})$

P(head | head of parent)

$P(\text{crackers} \mid \text{parent} = \text{ate})$

$P(\text{animal} \mid \text{parent} = \text{ate})$

$P(\text{crackers} \mid \text{parent} = \text{fed})$

$P(\text{animal} \mid \text{parent} = \text{fed})$

Dependency grammars

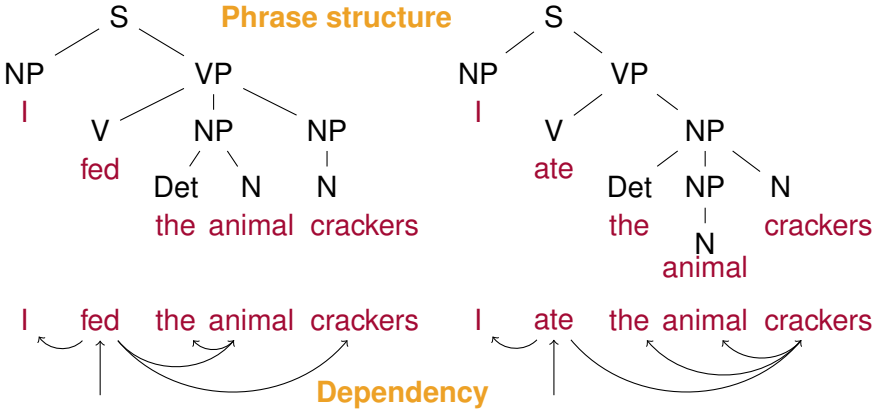
A different grammatical theory

- No phrases
- Dependencies = relations between words

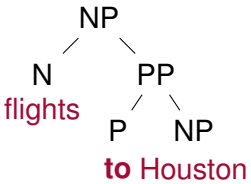
  head $\longrightarrow$ dependent

- “Head” notion different from phrase-structure grammars

Phrase structures and dependency structures



Heads in dependency structures



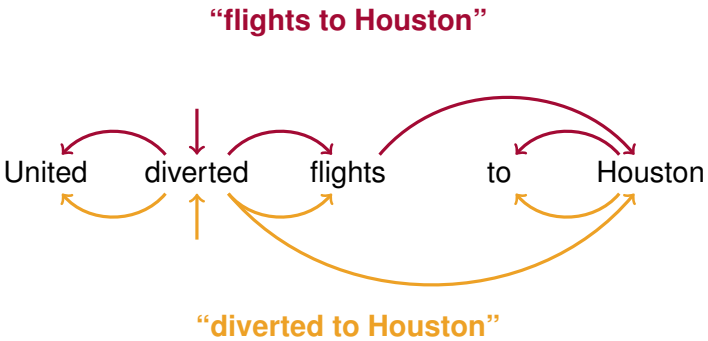
Dependency parsing

Arc-standard parsing (shift-reduce)

- Linear time
- Non-exhaustive search
- Use heuristics for finding the best path (trained oracle)
- Greedy

Other approaches (beam search, etc.)

Dependency parsing ambiguity



Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston		

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston	Shift	
root, diverted, flights, to, Houston			

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United ← diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston	Shift	
root, diverted, flights, to, Houston		LeftArc	to ← Houston
root, diverted, flights, Houston			

Dependency parsing derivation 1

"flights to Houston"

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston	Shift	
root, diverted, flights, to, Houston		LeftArc	to $\leftarrow$ Houston
root, diverted, flights, Houston		RightArc	flights $\rightarrow$ Houston
root, diverted, flights			

Dependency parsing derivation 1

“flights to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston	Shift	
root, diverted, flights, to, Houston		LeftArc	to $\leftarrow$ Houston
root, diverted, flights, Houston		RightArc	flights $\rightarrow$ Houston
root, diverted, flights		RightArc	diverted $\rightarrow$ flights
root, diverted			

Dependency parsing derivation 1

"flights to Houston"

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United ← diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**Shift**	
root, diverted, flights, to	Houston	Shift	
root, diverted, flights, to, Houston		LeftArc	to ← Houston
root, diverted, flights, Houston		RightArc	flights → Houston
root, diverted, flights		RightArc	diverted → flights
root, diverted		RightArc	root → diverted
root			

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston		

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston		

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston	Shift	
root, diverted, to	Houston		

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston	Shift	
root, diverted, to	Houston	Shift	
root, diverted, to, Houston			

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston	Shift	
root, diverted, to	Houston	Shift	
root, diverted, to, Houston		LeftArc	to $\leftarrow$ Houston
root, diverted, Houston			

Dependency parsing derivation 2

“diverted to Houston”

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston	Shift	
root, diverted, to	Houston	Shift	
root, diverted, to, Houston		LeftArc	to $\leftarrow$ Houston
root, diverted, Houston		RightArc	diverted $\rightarrow$ Houston
root, diverted			

Dependency parsing derivation 2

"diverted to Houston"

Stack	Word list	Action	Relations
root	United, diverted, flights, to, Houston	Shift	
root, United	diverted, flights, to, Houston	Shift	
root, United, diverted	flights, to, Houston	LeftArc	United $\leftarrow$ diverted
root, diverted	flights, to, Houston	Shift	
root, diverted, flights	to, Houston	**RightArc**	diverted $\rightarrow$ flights
root, diverted	to, Houston	Shift	
root, diverted, to	Houston	Shift	
root, diverted, to, Houston		LeftArc	to $\leftarrow$ Houston
root, diverted, Houston		RightArc	diverted $\rightarrow$ Houston
root, diverted		RightArc	root $\rightarrow$ diverted
root			

CSCI 544, Lecture 17: Semantics

Ron Artstein

2022-10-20

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for watching the lecture video, taking notes, and participating in class discussions. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes

Coding Assignment 3 will be graded soon

Coding Assignment 4 due October 25

Submit early, submit often!

Presentation:

Due Date	Task
October 18	Article selection
October 27	Presentation slides
November 1–10	Presentations

Project:

Due Date	Task
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Semantics

semantics *noun* **1** the study of meanings (Merriam-Webster)

Large field, multiple NLP tasks.

Generally fall under two broad categories:

Lexical semantics Meanings of words, word-like items (multi-word expressions)

Compositional semantics How word meanings are put together

Model-theoretic semantics

Generally associated with the work of Richard Montague

Map human language expressions to logical representation instantiated in a model

human language → logical representation ↔ model

Model-theoretic semantics

Generally associated with the work of Richard Montague

Map human language expressions to logical representation instantiated in a model

human language → logical representation ↔ model

A mouse lives in every building

↓

$\exists x(\text{mouse}(x) \wedge \forall y(\text{building}(y) \rightarrow x \text{ lives in } y))$
$\forall y(\text{building}(y) \rightarrow \exists x(\text{mouse}(x) \wedge x \text{ lives in } y))$

↔

Scope ambiguity

Generalized quantifiers

From the meanings of the sentences, we can derive meanings for phrases and words.

A mouse lives in every building

$$\begin{cases} \exists x(\text{mouse}(x) \wedge \forall y(\text{building}(y) \rightarrow x \text{ lives in } y)) \\ \forall y(\text{building}(y) \rightarrow \exists x(\text{mouse}(x) \wedge x \text{ lives in } y)) \end{cases}$$

Generalized quantifiers

A mouse	$\lambda P.\exists x(\text{mouse}(x) \wedge P(x))$
Every building	$\lambda Q.\forall y(\text{building}(y) \rightarrow Q(y))$

Determiners

A	$\lambda P \lambda Q.\exists x(P(x) \wedge Q(x))$
Every	$\lambda P \lambda Q.\forall x(P(x) \rightarrow Q(x))$

More scope ambiguities

Negation

- All that glitters isn't gold
- All doors won't open

Intension

- John finds a unicorn (unambiguous)
- John seeks a unicorn
  - $\exists x(\text{unicorn}(x) \wedge \text{John seeks } x)$
  - John seeks (^unicorn)

Event semantics

Pat ate the soup with a spoon at the restaurant on Sunday

"Flat" representation (Davidson, Hobbs)

e: agent(e) = Pat
theme(e) = soup
manner(e) = with a spoon
location(e) = the restaurant
time(e) = Sunday
eat(e)

PropBank (proposition bank) roles

Argument roles: Arg0 (~agent), Arg1 (~theme), Arg2...
Adjunct roles: manner, instrument, ...

Semantic role labeling

Identify the roles of each element in the sentence

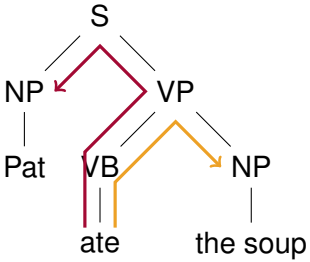
- Identify the arguments (“markables”)
- Identify the roles

Gildea and Palmer use features and classifier

Phrase Type	Parse Tree Path	Position	Voice	Head Word

Main question: to what extent are parse features useful?

Parse tree paths



Path to subject: Ate $\rightarrow$ Pat $=$ VB $\uparrow$ VP $\uparrow$ S $\downarrow$ NP

Path to object: Ate $\rightarrow$ The Soup $=$ VB $\uparrow$ VP $\downarrow$ NP

Lexical semantics

What is a word? — Dictionary entry

Homonyms: two (or more) words with the same spelling and pronunciation

bat[1] **bat[2]** **pitcher[1]** **pitcher[2]**

Polysemes: two (or more) senses of the same word

newspaper 1. **2.**

Is the distinction between homonymy and polysemy psychologically real?

Frazier and Rayner, Journal of Memory and Language 29(2):181–200, 1990

Distributional properties of words

"If we consider **oculist** and **eye-doctor** we find that, as our corpus of actually-occurring utterances grows, these two occur in almost the same environments... In contrast, there are many sentence environments in which **oculist** occurs but **lawyer** does not...

Zellig S. Harris, Distributional Structure,
Word 10(2-3):146–162, 1954

Association between words and contexts

	c1	c2	c3	c4	c5	c6	c7	c8	c9	$\cdots$
w1							•			
w2										
w3		•								
w4							•			
w5			•							
w6										
$\vdots$										

Word $\times$ context matrix

PPMI: positive pointwise mutual information

$$\max\left(0, \log \frac{\hat{P}(w,c)}{\hat{P}(w)\hat{P}(c)}\right)$$

Sparse

Dimensionality reduction

SVD (Singular Value Decomposition)

- Decompose sparse matrix
  - Reconstruct in lower dimension
- Used in Latent Semantic Analysis, etc.

Neural Language Models

- By-product: word embedding matrix

Omer Levy, Yoav Goldberg and Ido Dagan: Improving Distributional Similarity with Lessons Learned from Word Embeddings, TACL 2015

Why are neural models more successful?

Designing a low-dimension matrix

Preprocessing: What counts as a “context”?

- What constitutes the columns of the sparse matrix?

Association: How are words counted in contexts?

- What constitutes the cells of the sparse matrix?

Postprocessing:

- How are the values of the dense matrix optimized?

All of these are **hyperparameters** which determine the usefulness of the word embeddings.

Comparing traditional and neural embeddings

Multiple embeddings, varying:

- Hyperparameters
- Factorization algorithm
- Amount of data

Hyperparameters appear to have the largest impact.

CSCI 544, Lecture 18: Reference, Generation, Summarization

Ron Artstein

2022-10-25

These notes are not comprehensive, and do not cover the entire lecture. They are provided as an aid to students, but are not a replacement for attending class, participating in the discussion, and taking notes. Any distribution, posting or publication of these notes outside of class (for example, on a public web site) requires my prior approval.

Administrative notes

Coding Assignment 3 grading in progress

- 88% auto-graded; working on reaining 12%

Coding Assignment 4 due today

- Will be extended till end-of-day: submit working code!
- Late deadline (with penalty) will be extended

Presentation:

Due Date	Task
October 27	Presentation slides
November 1–10	Presentations

Project:

Due Date	Task
November 3	Project status report
Nov 29/Dec 1	Poster presentations (in class)
December 1	Final report
December 3	Self-evaluation and peer grading

Reference (resolution)

Also known as co-reference, anaphora

Identify which expressions refer to the same things

- **mention:** a referring expression (“markable”)
- **entity:** a set of mentions

Common operationalization

- Identify mentions
- Group them into entities

But there are more complex situations

Also, expressions may refer to events

Anaphoric ambiguity

Poesio and Artstein, ACL workshop 2005

18.1 S: ....
18.6 it turns out that the boxcar at Elmira
18.7 has a bad wheel
18.8 and they're .. gonna start
fixing that at midnight
18.9 but it won't be ready until 8
19.1 M: oh what a pain in the butt

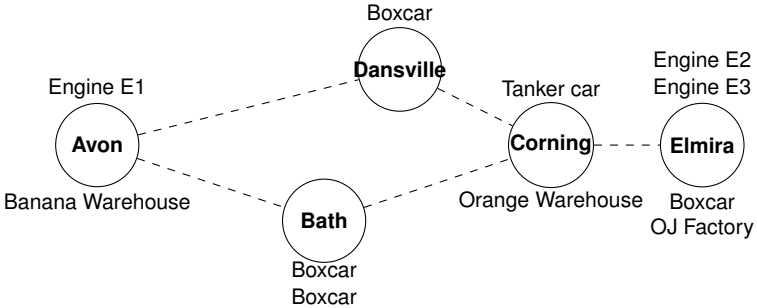
Experiment setup

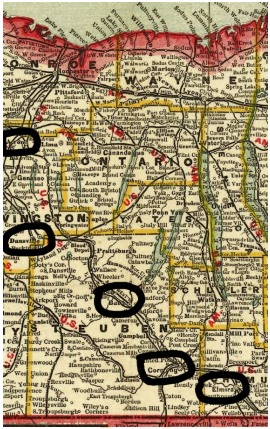
18 naïve subjects (annotators)

Dialogue 3.2 from the TRAINS 91 corpus

MMAX 2 annotation tool

Map of the “TRAINS world”





Annotation scheme

- Each NP marked with one of four attributes:

  `Place` one of Avon, Bath, Corning, Dansville, Elmira
  = nominal attribute

  `Phrase` previously mentioned object
  = pointer(s) to previous markable(s)

  `Segment` previously discussed plan, event or action
  = pointer(s) to previous turn(s)

  `None` novel object, non-referential NP
  = nominal attribute

- Subjects instructed to use multiple pointers for cases of ambiguity

Agreement between annotators

19.10: we need to get the bananas to Corning by 3
19.11: uh
19.12: **maybe** it 's gonna be faster if we
19.13: send E1
19.14: E1 's boxcar picks up at Dansville
19.15: instead of going back to Avon
19.16: have it go on to Corning
19.17: uh pick up the tanker get the oranges send them to Elmira
19.18: cause that 's gonna be the longest thing

Agreement between annotators

19.10: we need to get the bananas to Corning by 3
19.11: uh
19.12: **maybe** it 's gonna be faster if we
19.13: send E1
19.14: E1 's boxcar picks up at Dansville
19.15: instead of going back to Avon
19.16: have it go on to Corning
19.17: uh pick up the tanker get the oranges send them to Elmira
19.18: cause that 's gonna be the longest thing

Key: Full agreement

Agreement between annotators

19.10: we need to get the bananas to Corning by 3
19.11: uh
19.12: **maybe** it 's gonna be faster if we
19.13: send E1
19.14: E1 's boxcar picks up at Dansville
19.15: instead of going back to Avon
19.16: have it go on to Corning
19.17: uh pick up the tanker get the oranges send them to Elmira
19.18: cause that 's gonna be the longest thing

Key: Full agreement One outlier

Agreement between annotators

19.10: we need to get the bananas to Corning by 3
19.11: uh
19.12: **maybe** it 's gonna be faster if we
19.13: send E1
19.14: E1 's boxcar picks up at Dansville
19.15: instead of going back to Avon
19.16: have it go on to Corning
19.17: uh pick up the tanker get the oranges send them to Elmira
19.18: cause that 's gonna be the longest thing

Key: Full agreement One outlier Implicit

Agreement between annotators

19.10: we need to get the bananas to Corning by 3
19.11: uh
19.12: **maybe** it 's gonna be faster if we
19.13: send E1
19.14: E1 's boxcar picks up at Dansville
19.15: instead of going back to Avon
19.16: have it go on to Corning
19.17: uh pick up the tanker get the oranges send them to Elmira
19.18: cause that 's gonna be the longest thing

Key: Full agreement | One outlier | Implicit | Explicit

Implicit ambiguity

Ambiguity not always noticed by all the coders.

1.4 M: first thing **I'd** like you to do
1.5 is send engine E2 off with a boxcar to Corning to pick up oranges
1.6 uh as soon as possible
2.1 S: okay [6 sec]
3.1 M: and while it's there it should pick up the tanker

Reference of it:	a boxcar	3 coders
	engine E2	6 coders
	ambiguous	9 coders

Marking segment antecedents

Artstein and Poesio, Semdial 2006

All non-temporal NPs are markables on the **phrase** level.

Antecedents of discourse deixis are markables on the **segment** level.

S: if we took [the engine from Avon]
M: +to Bath+ apparently
S: if we took [it] to Bath
[that] would take 8 hours

Segments are arbitrary regions of text defined by the annotators.

Fine-grained marking

M: so
essentially we have to
... again get [the boxcar]
and [engine]
to Corning
so [the fastest way to do [that]] is from Elmira
so we 'll do [that]

M: so
essentially we have to
... again get [the boxcar]
and [engine]
to Corning
so [the fastest way to do [that]] is from Elmira
so we 'll do [that]

Variability in annotation (noise)

70.1 S: okay which
70.2 : ah well at the moment
70.3 : the plan is to have the .. engine E1
70.4 : is gonna take the boxcar from Dansville
71.1 M: right
72.2 S: so
72.3 : if we wanna keep **that** then we'd have to have
72.4 : engine E2 pick up the boxcar from Bath

that → 2 1 1 9 2 new:1 non-referential:3

Overlapping segments

3.3–3.5 : ... again get the boxcar and engine to Corning
3.6 : so the fastest way to do that is from Elmira
3.7 : so we'll do that
⋮
7.3 : so we ship one
7.4 : boxcar
7.5 : of oranges to Elmira
7.6 : and **that** takes another 2 hours

that → 1 3 2 3 1

Generation

Creation of new texts

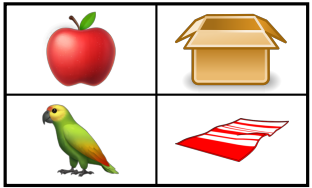
- Examples: translation; image captioning
- Typically based on some input
- Somewhat removed from input

Common paradigm: neural language model decoding

Other approaches: template, syntax-based, etc.

Generating the best referring expression

Visual world paradigm

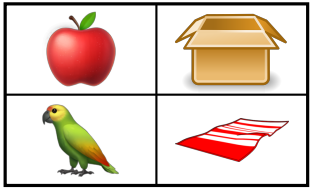


put the apple on the towel

put the apple in the box

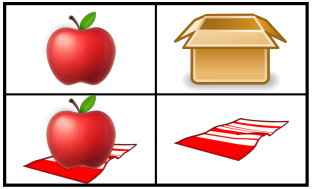
Generating the best referring expression

Visual world paradigm



put the apple on the towel

put the apple in the box



put the apple on the towel in the box

Summarization

Extractive

- Most important sentences in a document
- Most important words/phrases in a sentence

Abstractive

- Generate new paragraph/sentence based on existing document/sentence

Unsupervised extractive document summarization:

- Yihong Gong and Xin Liu. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. SIGIR Proceedings, September 2001
- Josef Steinberger and Karel Ježek. Text Summarization and Singular Value Decomposition. Advances in Information Systems, 2004

Evaluating Summarization

ROUGE: Recall-Oriented Understudy for Gisting Evaluation

- N-gram based similarity to reference summaries
- Inspired by BLEU (Machine Translation)
- Oriented towards recall rather than precision

Why is recall more important for summarization?

Final thoughts

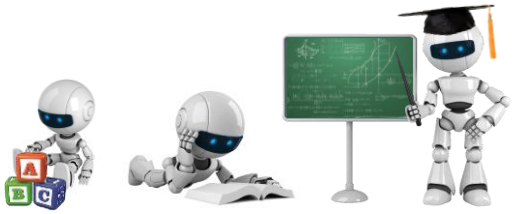
What is NLP?

- **Computational modeling of human language**
- Learn to think about the language in the problem
  - Learn from related fields
- Think about insights to achieve the goal
- Critically examine the evaluation methods
- More than chasing performance metrics

USC Viterbi School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Portions from Slides by Wang Yue and Giuseppe Attardi

Recurrent Neural Networks

- We can consider NL data as sequential data points, where the current word depends on the previous words in the sequence:

  1 2 3 4 5 6 7 8 9 10 11

- Ex: Today, I want to play football and then watch a movie.
- Learning representations by back-propagating errors, 1986
- Core Idea: the function approximator can receive the input word by word such that its output depend on the history, i.e., relying on a notion of memory

FNN $$y = f(x)$$

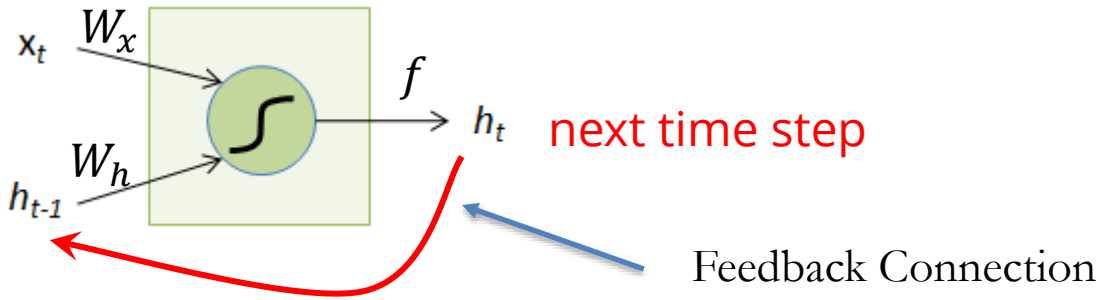
RNN $$y_t = f(x_t, h_{t-1})$$

output input memory

Recurrent Neural Networks

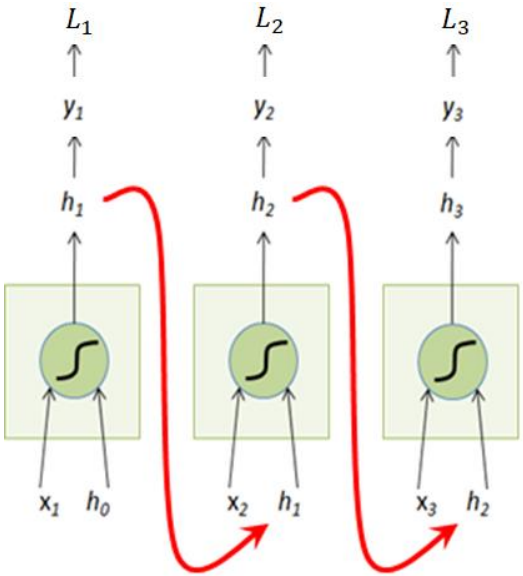
- Equipping perceptron with memory

- x_t: Input at time t
- h_{t-1}: State at time t-1



$$h_t = f(W_x x_t + W_h h_{t-1}), W_x \in \mathbb{R}^{M*N}, W_h \in \mathbb{R}^{H*H}$$

- Unfolding RNN
- We can make the unit multi-layer



Figures adopted from slides by Arun Mallya

Recurrent Neural Networks

- The weight matrices are shared across time

- multi-output

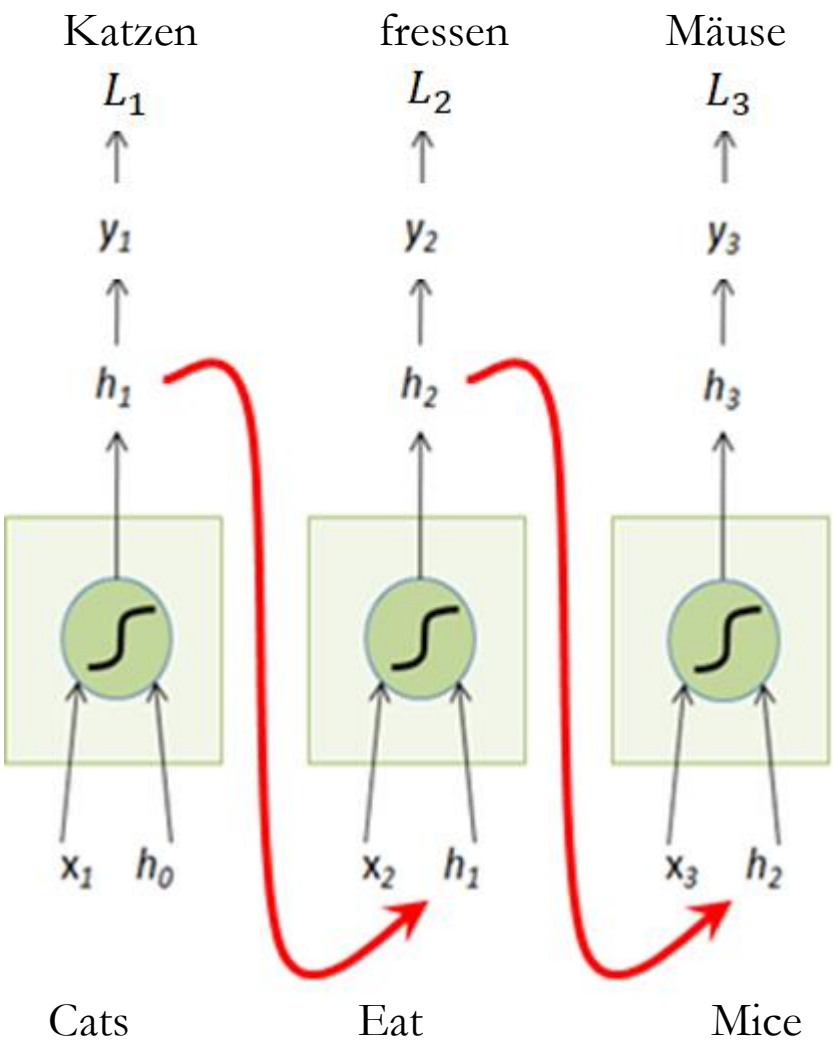
$$L = L_1 + L_2 + L_3$$

$$L = \sum_{t=1}^{T} L_t$$

$$y_t = f(x_t, h_{t-1})$$

$$L_t = l(y, \hat{y}_t)$$

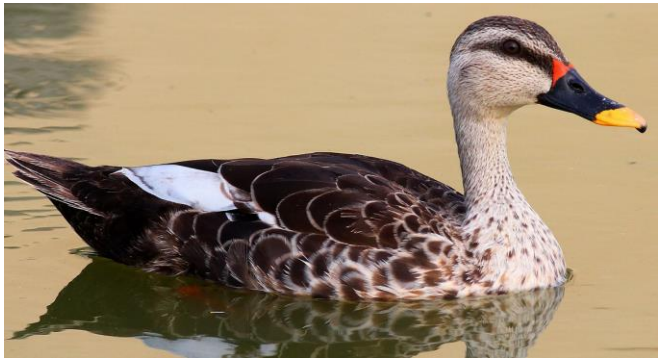
- Sequential processing
- Hard to train
- Long range dependencies



Figures adopted from slides by Arun Mallya

Attention Mechanism

Images



Indian Spot-billed Duck

Mandarin Duck

Text

Deep Learning

- Table of Contents
- Acknowledgements
- Notation
- 1 Introduction
- Part I: Applied Math and Machine Learning Basics
  - 2 Linear Algebra
  - 3 Probability and Information Theory
  - 4 Numerical Computation
  - 5 Machine Learning Basics
- Part II: Modern Practical Deep Networks
  - 6 Deep Feedforward Networks
  - 7 Regularization for Deep Learning
  - 8 Optimization for Training Deep Models
  - 9 Convolutional Networks
  - 10 Sequence Modeling: Recurrent and Recursive Nets
  - 11 Practical Methodology
  - 12 Applications
- Part III: Deep Learning Research
  - 13 Linear Factor Models
  - 14 Autoencoders
  - 15 Representation Learning
  - 16 Structured Probabilistic Models for Deep Learning
  - 17 Monte Carlo Methods
  - 18 Confronting the Partition Function
  - 19 Approximate Inference
  - 20 Deep Generative Models

Convolutional Neural Networks: Motivation

- Some differentiating patterns are local

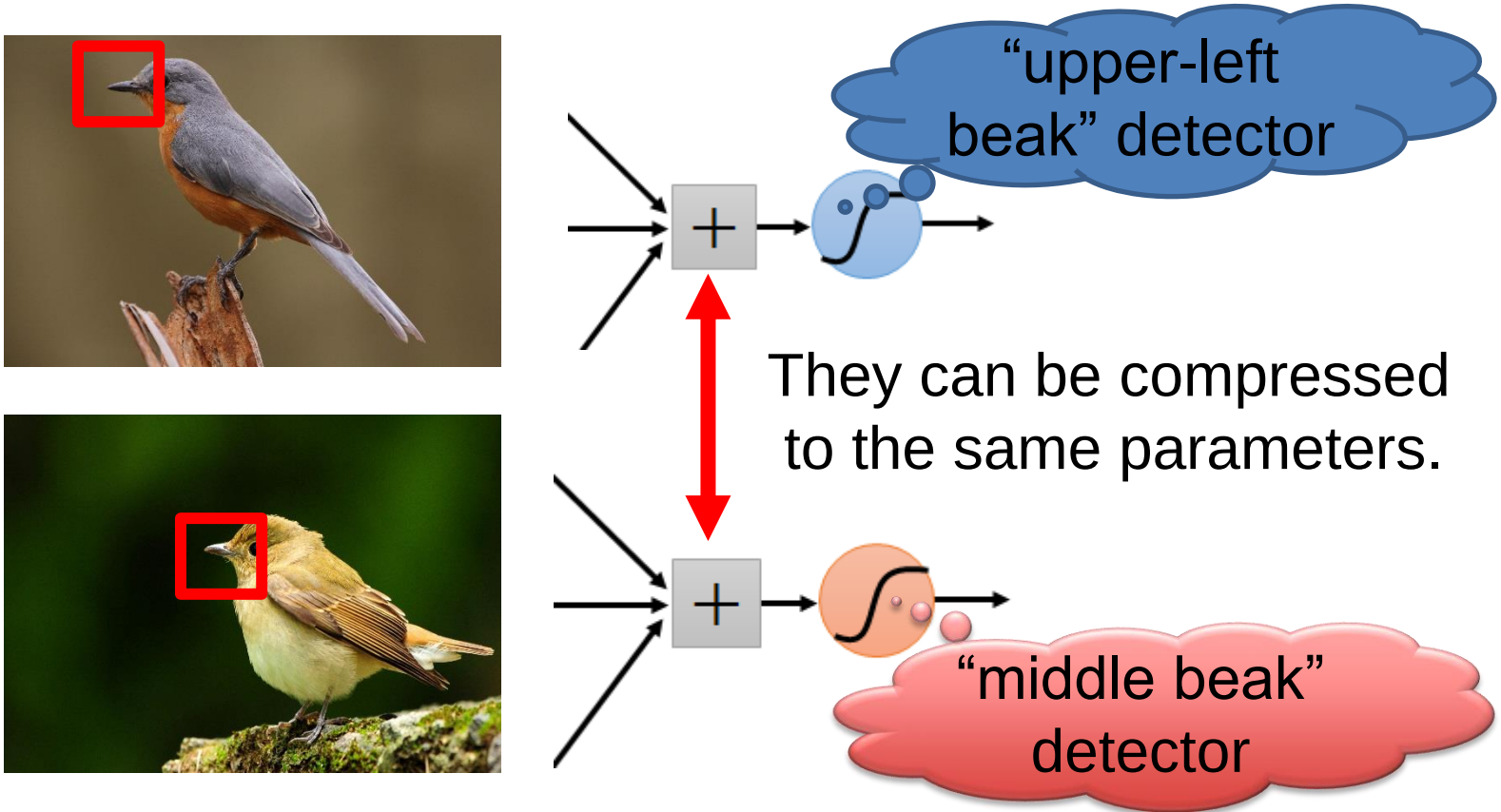
 - What is a bird?

- We should attend to smaller locations to perform a task

Convolutional Neural Networks: Motivation

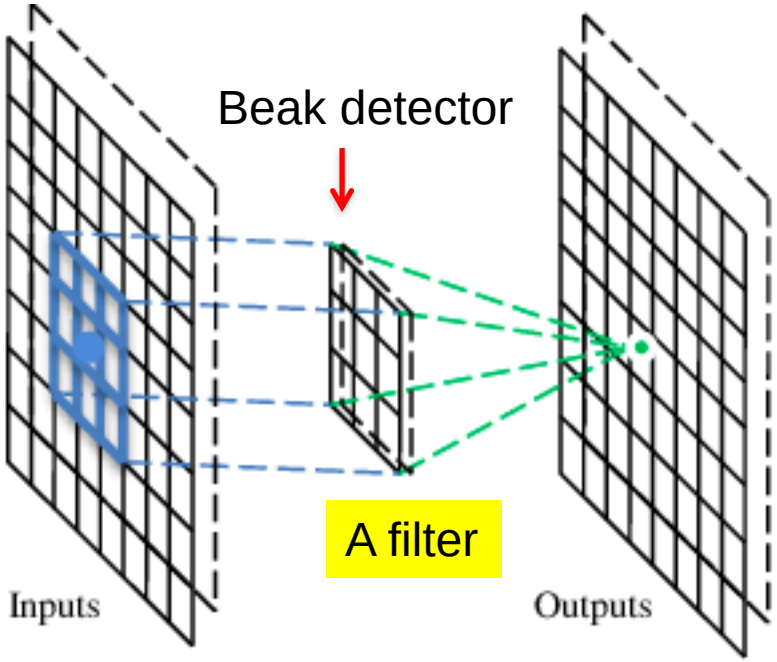
Local pattern appears in different places.

Solution: training a lot of “local” detectors and move around each detector to make the model spatially invariant.



Convolutional Layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolutional Layer

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

$$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$$

Each filter detects a small pattern (3 x 3).

Convolutional Layer

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

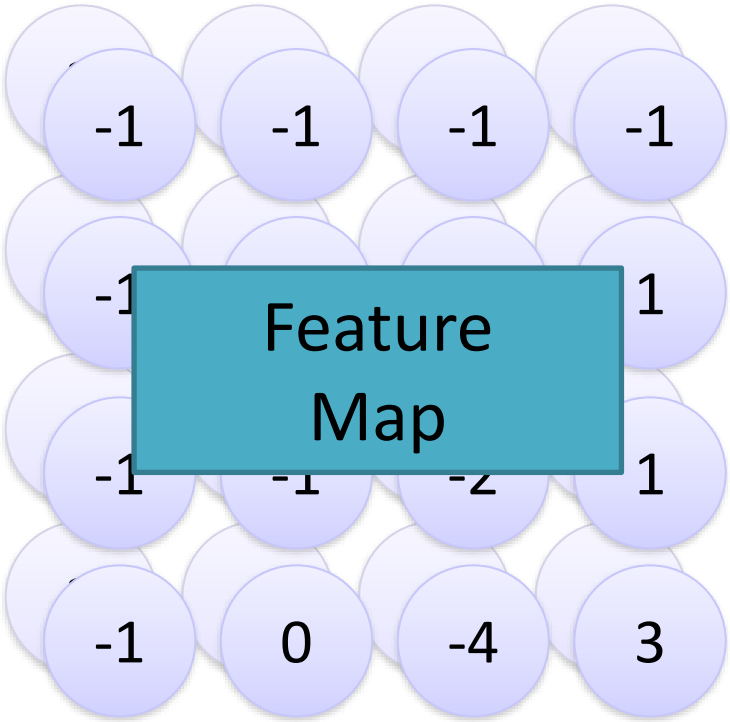
Convolutional Layer

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Convolutional Layer

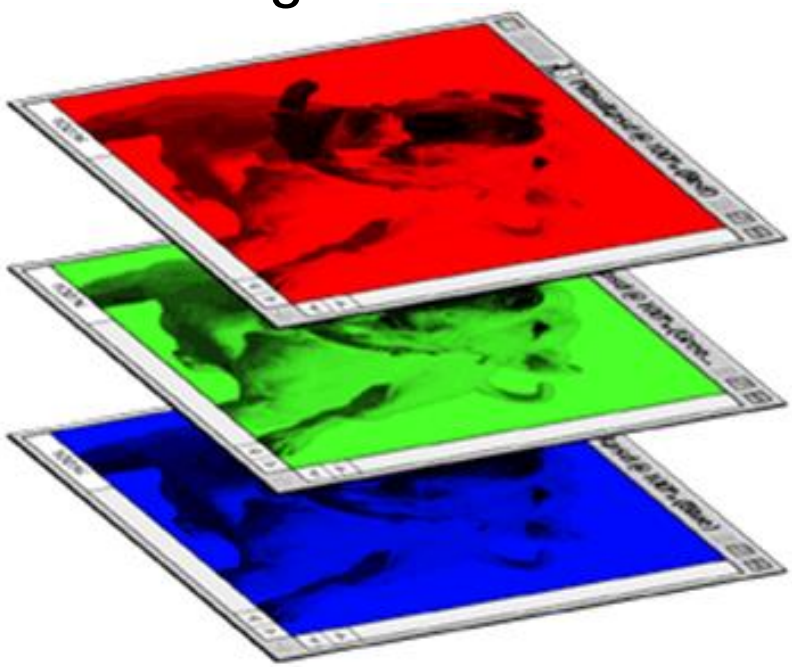
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

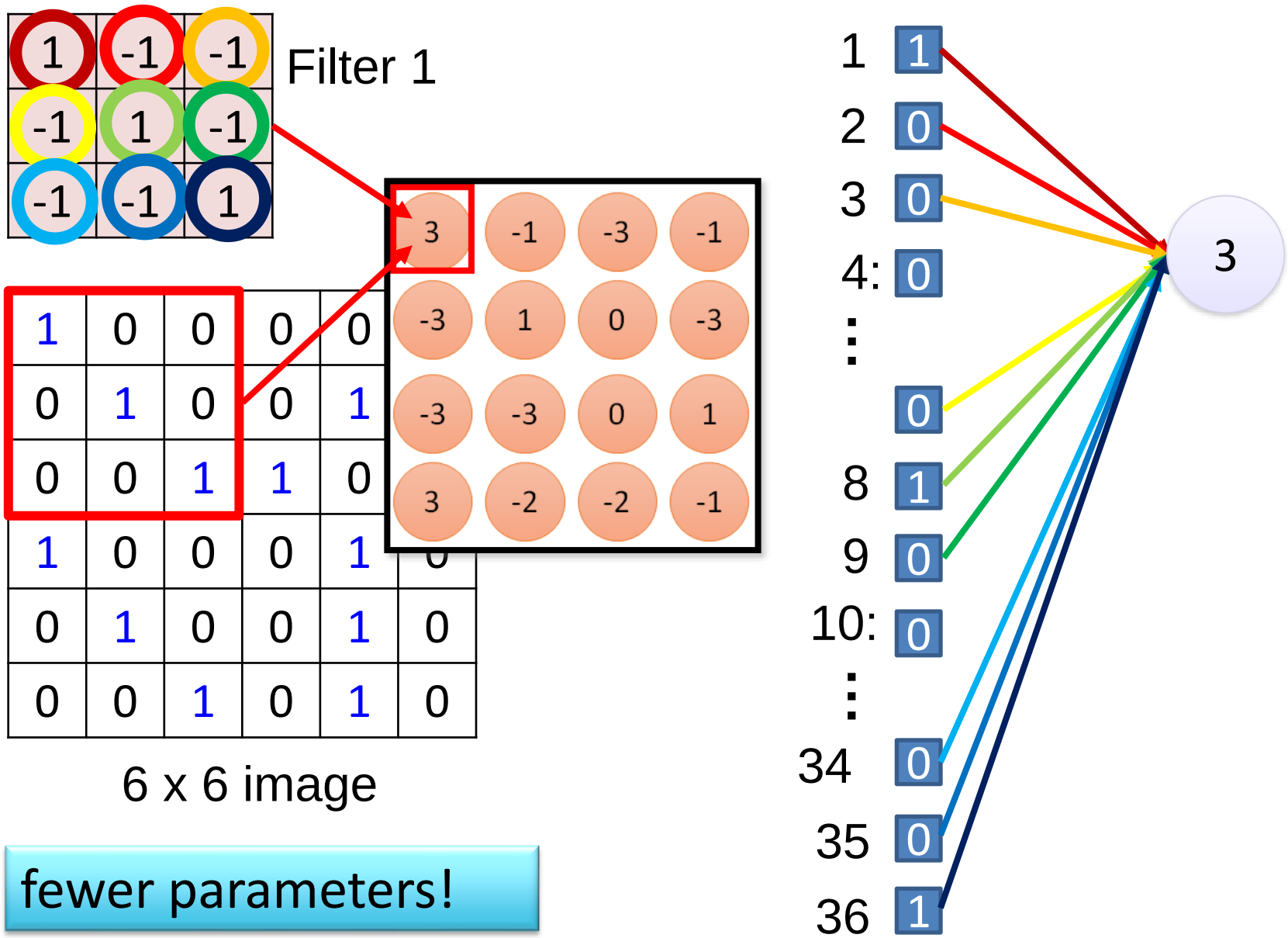
Filter 2

Color image



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Convolutional vs FCL



fewer parameters!

Max Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

- fewer parameters to characterize the image

Max Pooling Layer

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

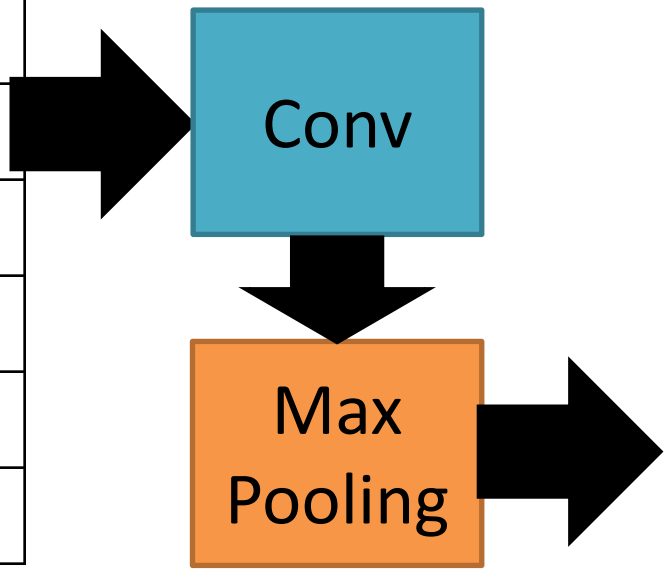
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

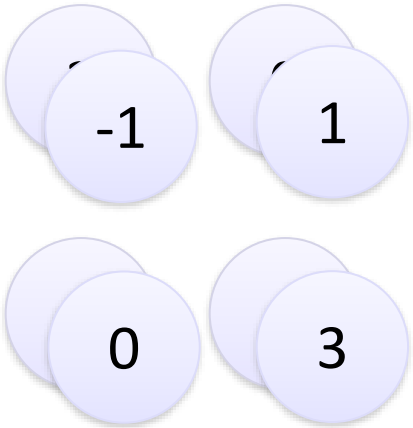
Max Pooling Layer

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



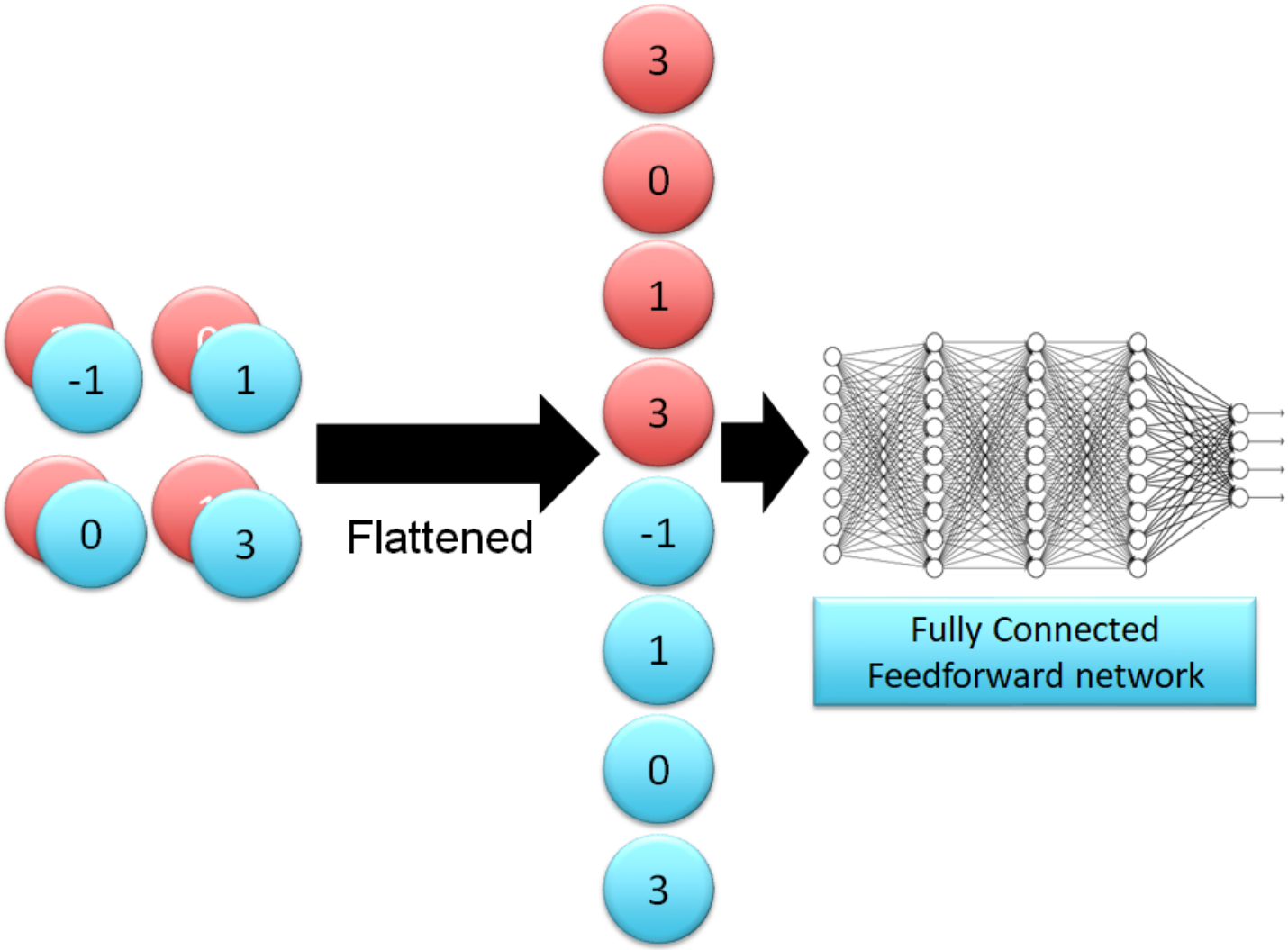
New image but smaller



2 x 2 image

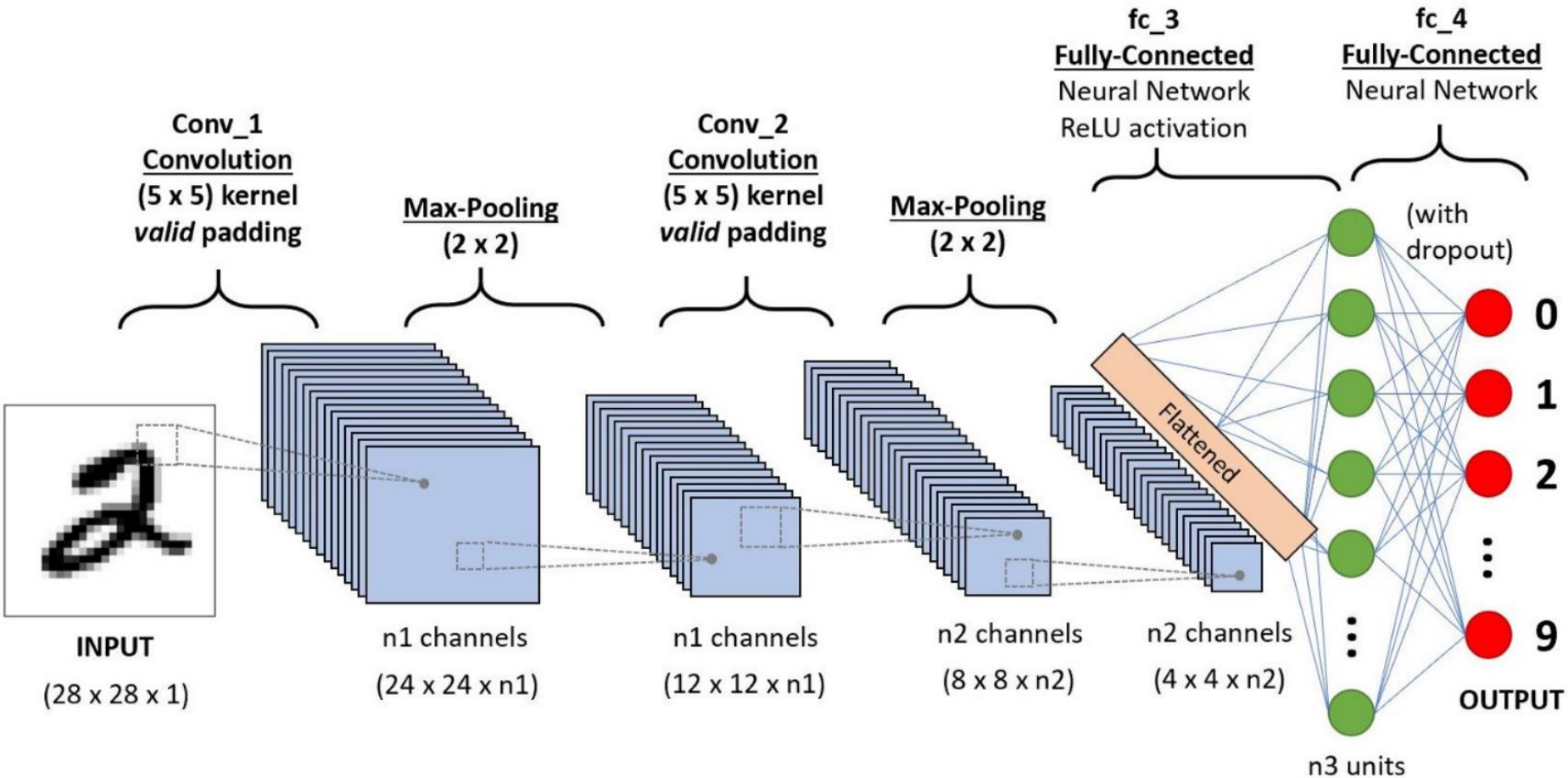
Each filter is a channel

Flattening



Revolutionized Computer Vision!

Convolutional Neural Networks

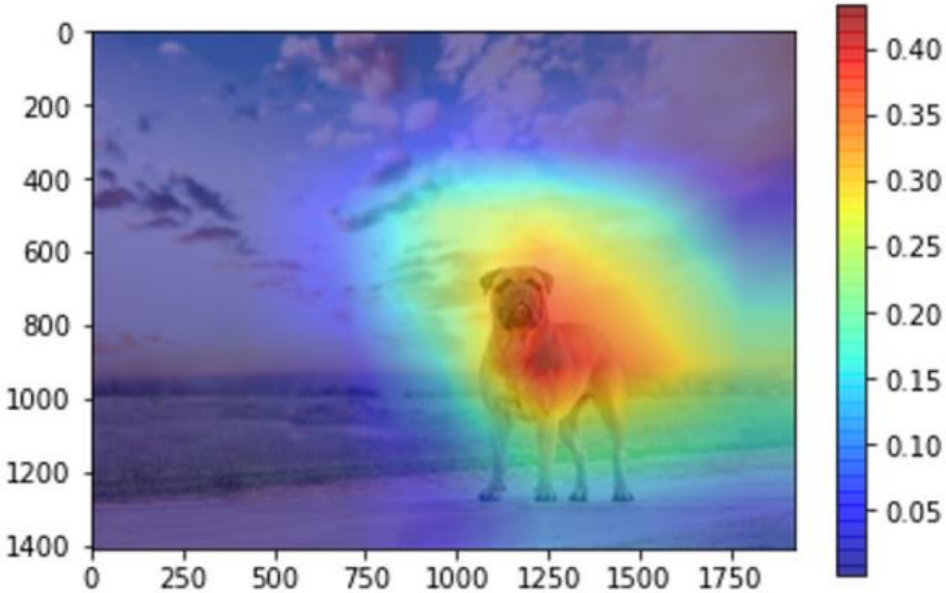


Revolutionized Computer Vision!

Attention in CNNs

- The network learns to attend to the proper spatial location to perform a downstream task

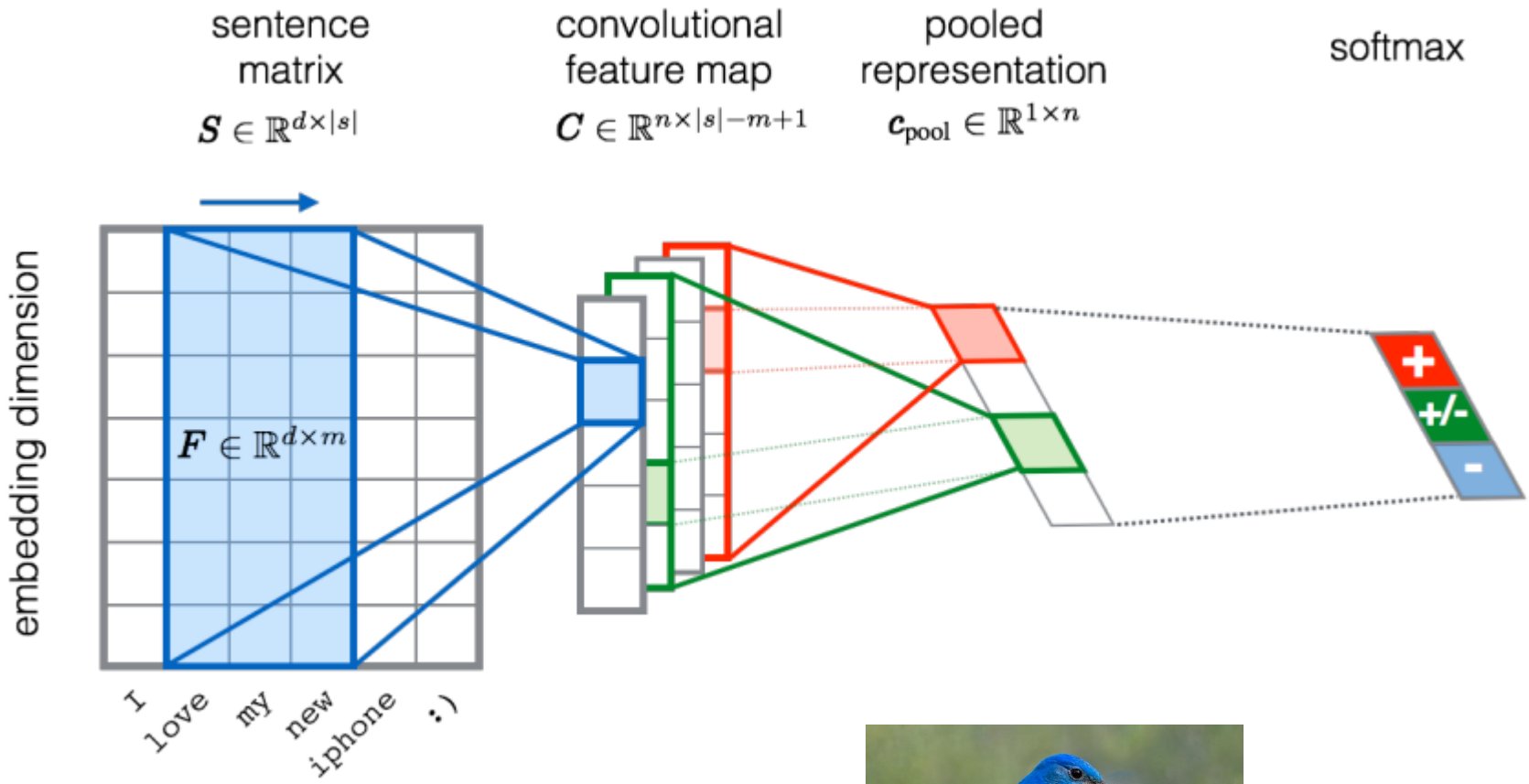
(a) Original image



(b) Attention map

CNNs for Text Classification

- Sentence Sizes are different



- We can use 1D convolution

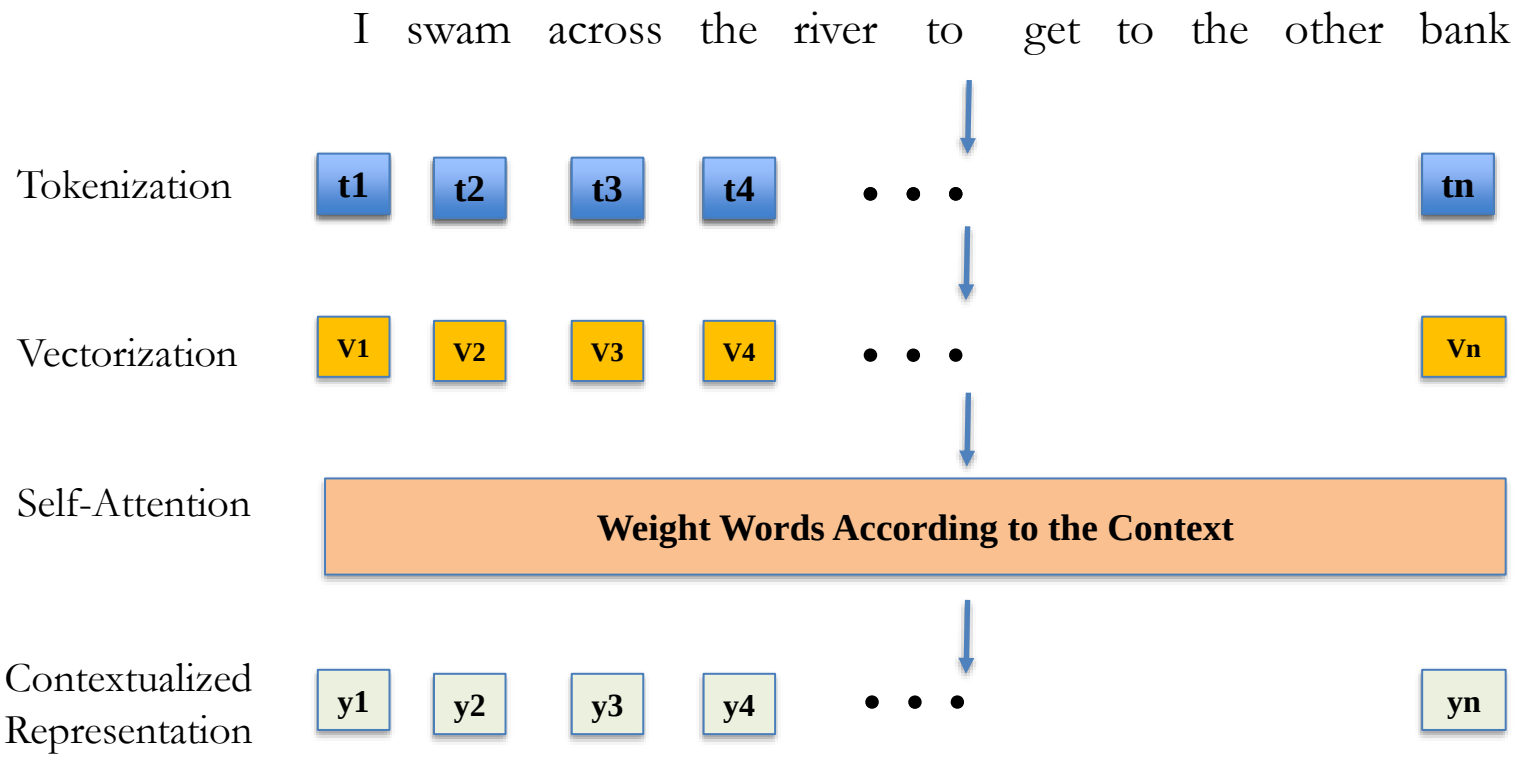
Transformers: intuition

- Example:

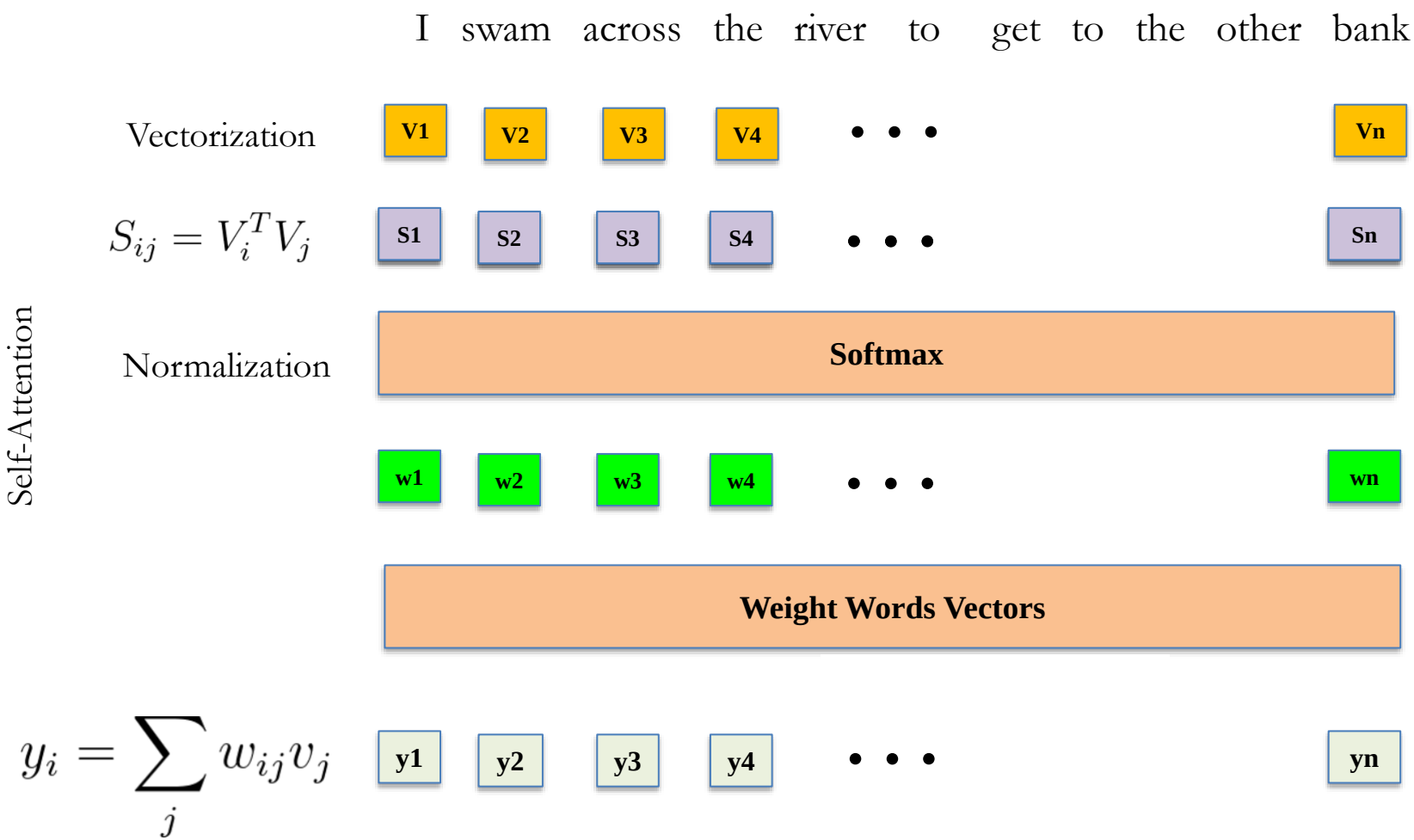
- I swam across the river to get to the other **bank**
- I drove across the road to get to the other **bank**

Context Matters

- Can we build an architecture that weighs neighboring words to understand meaning of words?



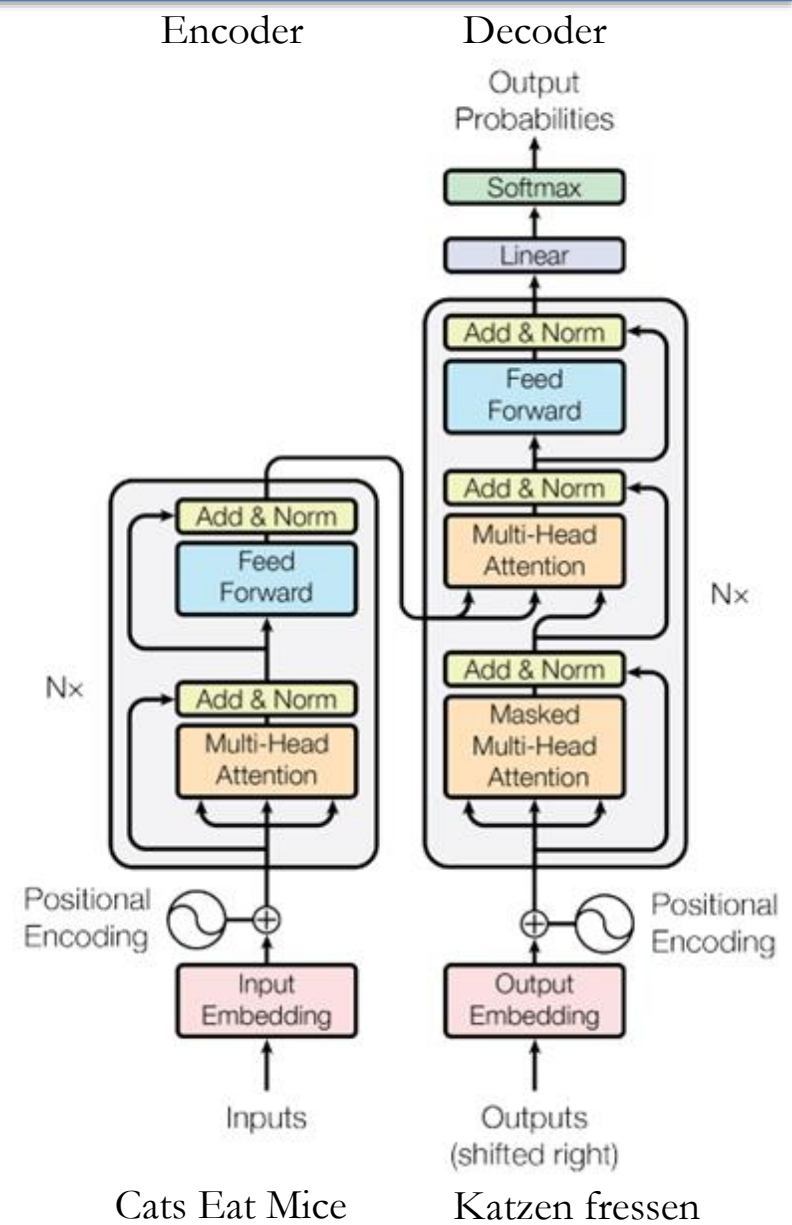
Transformers: Attention Mechanism Basics



Transformers

- Attention over entire input: path lengths between words become constant
- Parallelization

- Inputs: word vectors
  - Source sentence
  - Input sentence produced so far
- Output: word vectors
  - Probability distribution for the next word
- Position Encoding



Transformers

Embeddings and Softmax

We maximize the probability of the next output token

Share embedding weights and the pre-softmax linear transformation

Repetitive Blocks

Attention Blocks

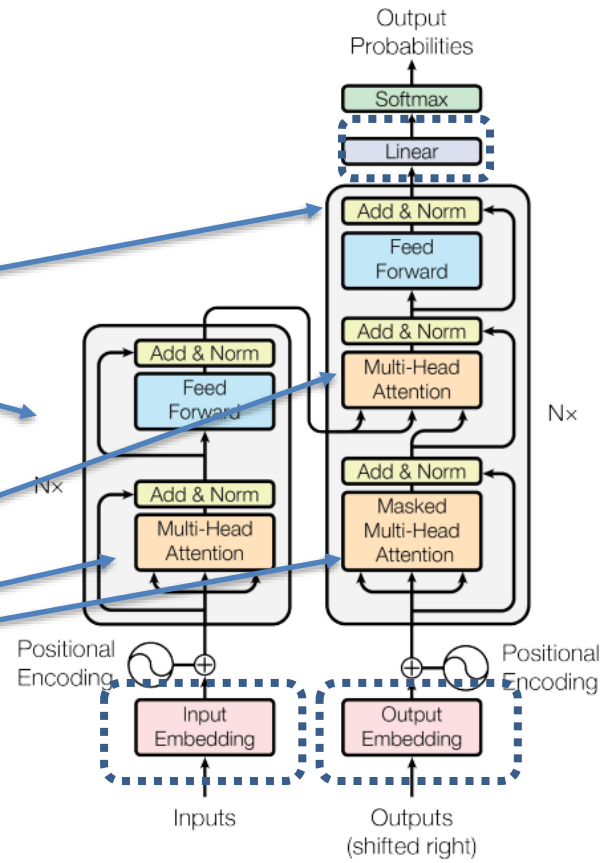
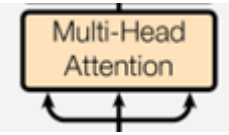


Figure 1: The Transformer - model architecture.

Transformers: Attention Mechanism

Which word in the source sentence we should pay more attention at each step?

Queries: Target Sentence
Keys: Source Sentence
Values: Source Sentence



Three inputs

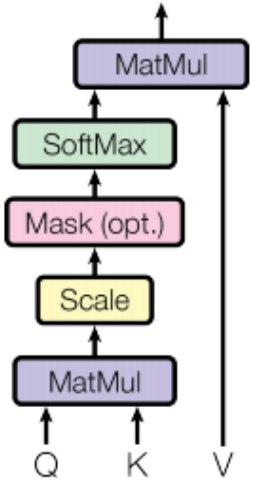
Scaled Dot-Product Attention: it selects the most important word in the source

Similarity

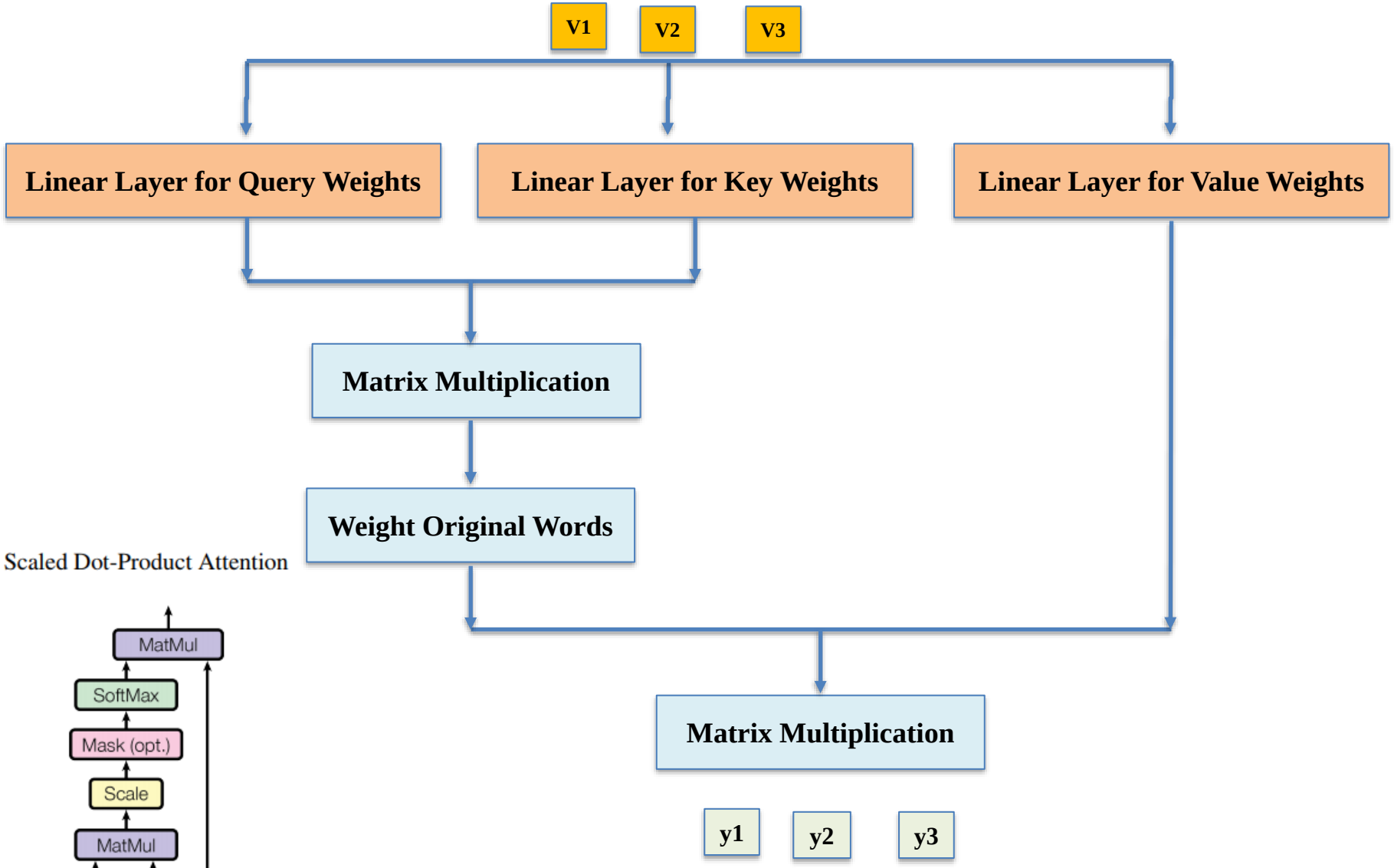
Q: queries
K: keys
V: values

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

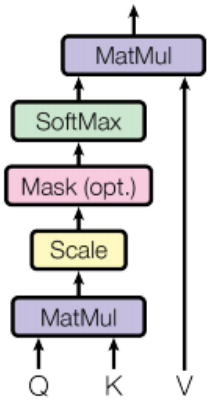
Scaled Dot-Product Attention



Transformers: Attention Mechanism

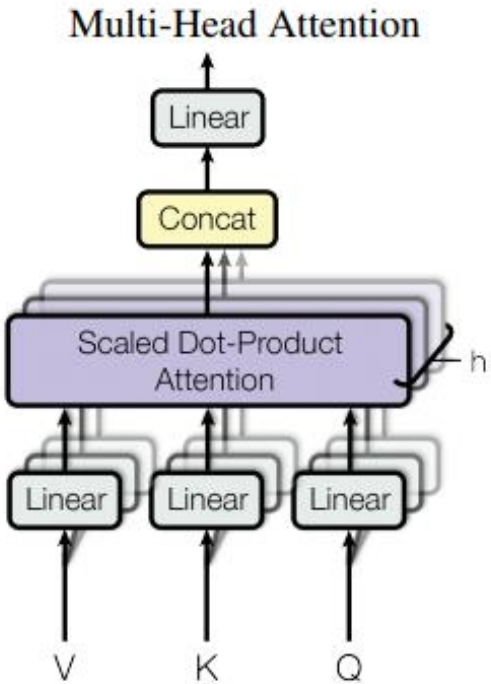


Scaled Dot-Product Attention



Transformers: Attention Mechanism

Multi-Head Attention: Learn **h** attention heads to learn different representation simultaneously so each head pays to different types of things



For projecting back to the input dimension

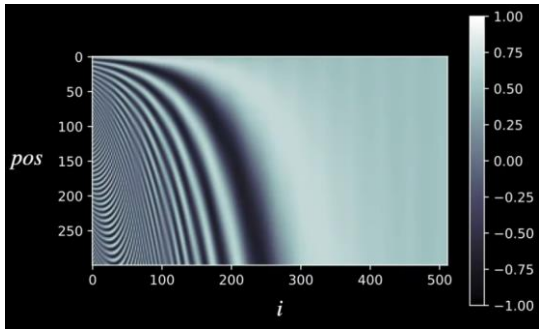
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Transformers: Positional Encoding

How do we take words orders into accounts?

- Augment the embeddings with a position identifier



Reason: no RNN to model the sequence position

Two types:

- learned positional embeddings Sinusoid:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Bounded, Periodic, Unique

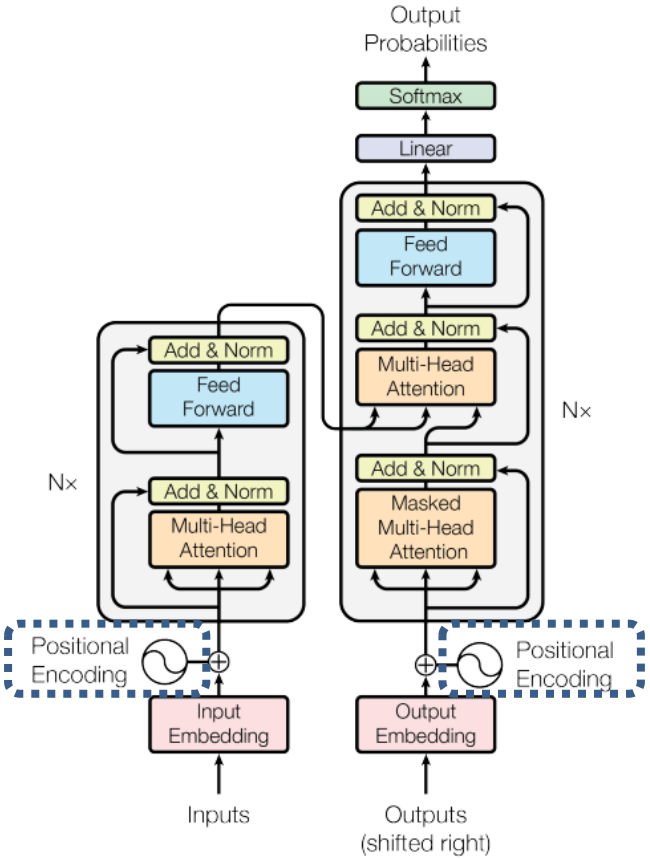


Figure 1: The Transformer - model architecture.

Transformers

Position-Wise Feed-Forward Networks

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

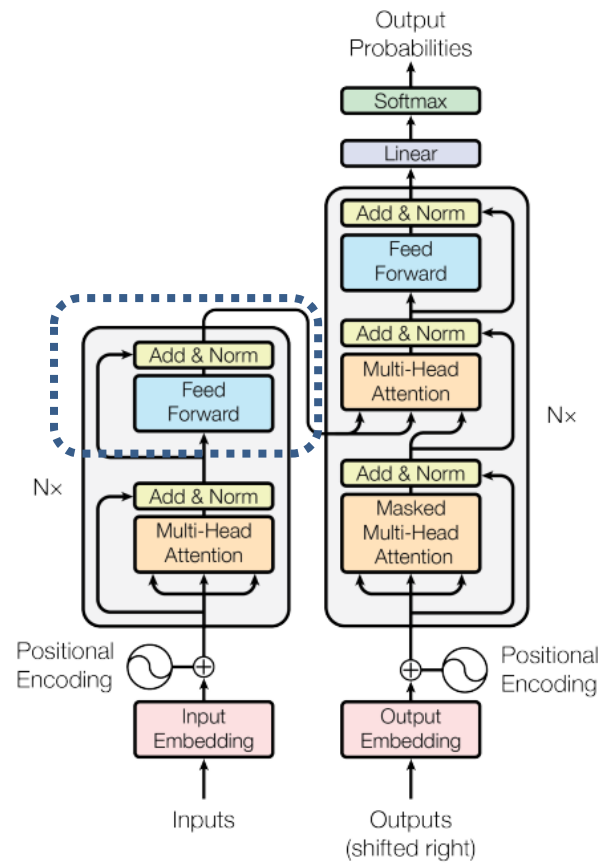


Figure 1: The Transformer - model architecture.

Transformers: Decoder

Masked Multihead Attention: limits decencies only to prior words

$$\text{Attention}(Q, K, V) = \text{softmax}(\text{Mask} + \frac{QK^T}{\sqrt{d_k}})V$$

Improvements over LSTM:

- The source languages and their semantics and grammars are analyzed
- Mapping to the source language is learned separately

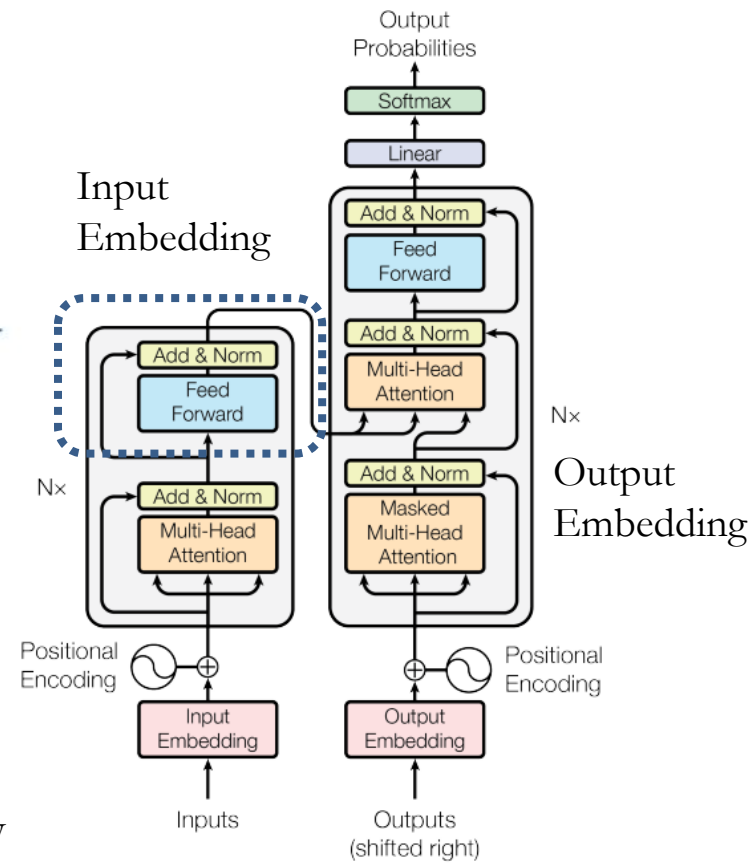


Figure 1: The Transformer - model architecture.

Performance Comparison

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

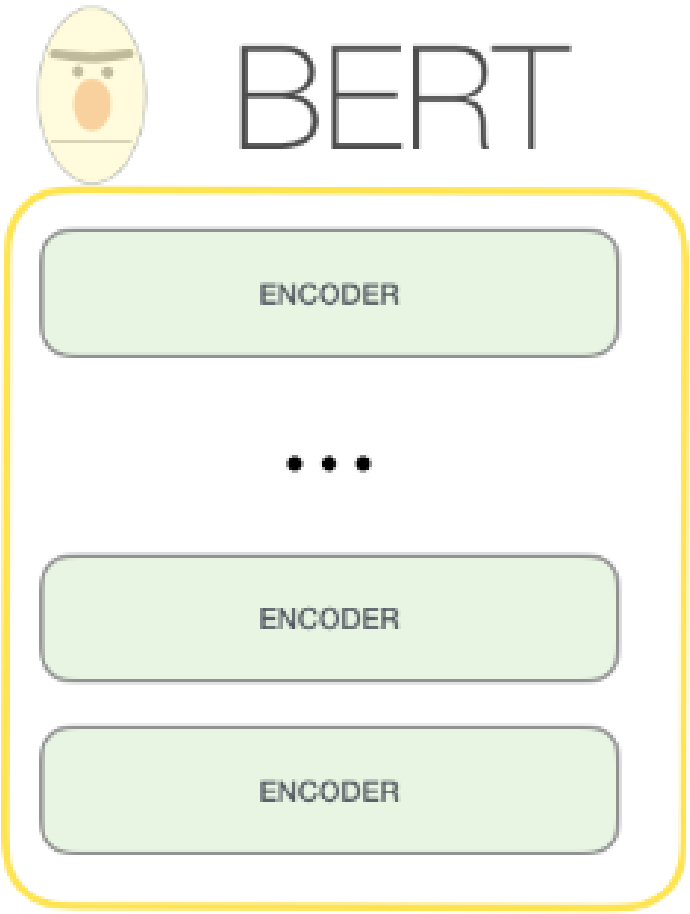
GPTs and BERT

- A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
- If we do not have input and just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPTs.
  - Application: question answering, summarization
- If we are only interested in training a model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.
  - Application: classification

GPT and BERT

Two steps:

- Pretraining using self-supervised learning
- Finetuning

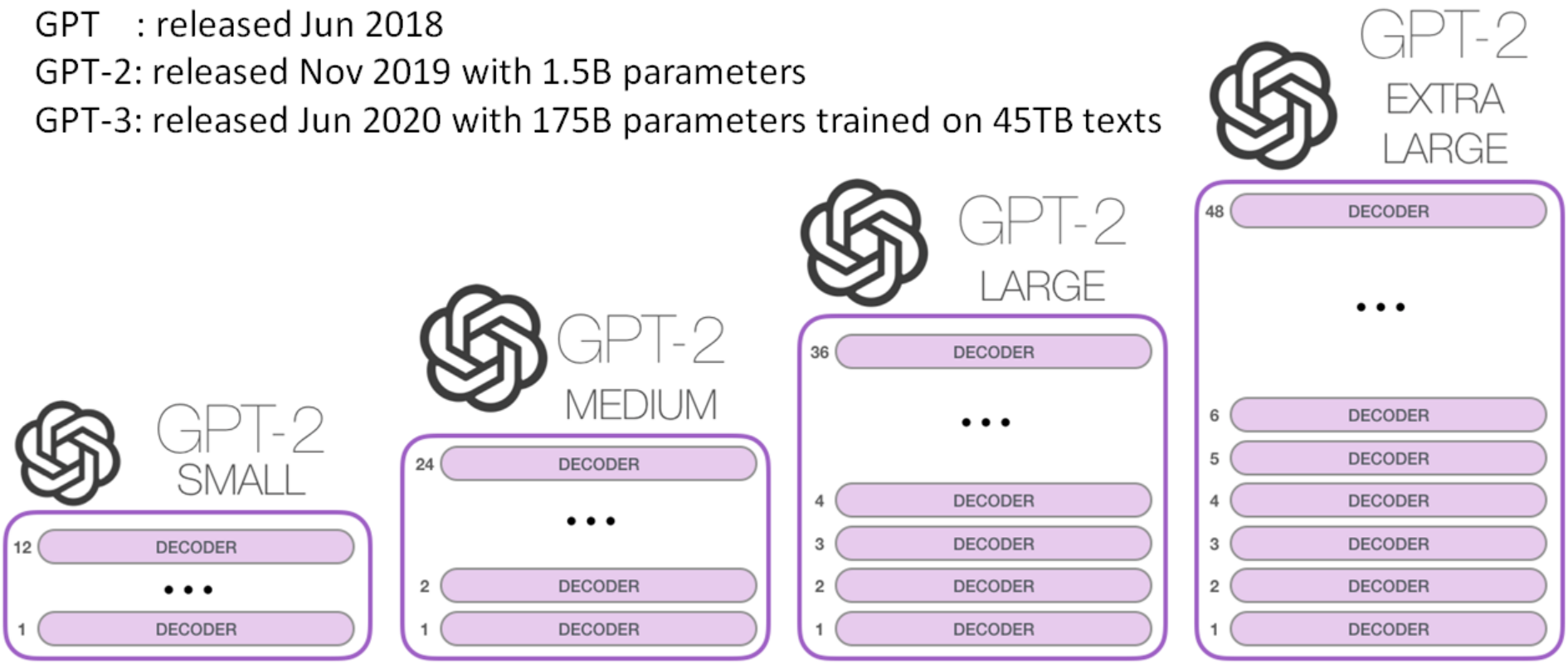


Tasks: classification, similarity, multiple choice, next word

Tasks: masking, next sentence prediction

GPT Versions

GPT : released Jun 2018
GPT-2: released Nov 2019 with 1.5B parameters
GPT-3: released Jun 2020 with 175B parameters trained on 45TB texts



Model Dimensionality: 768
117M parameters

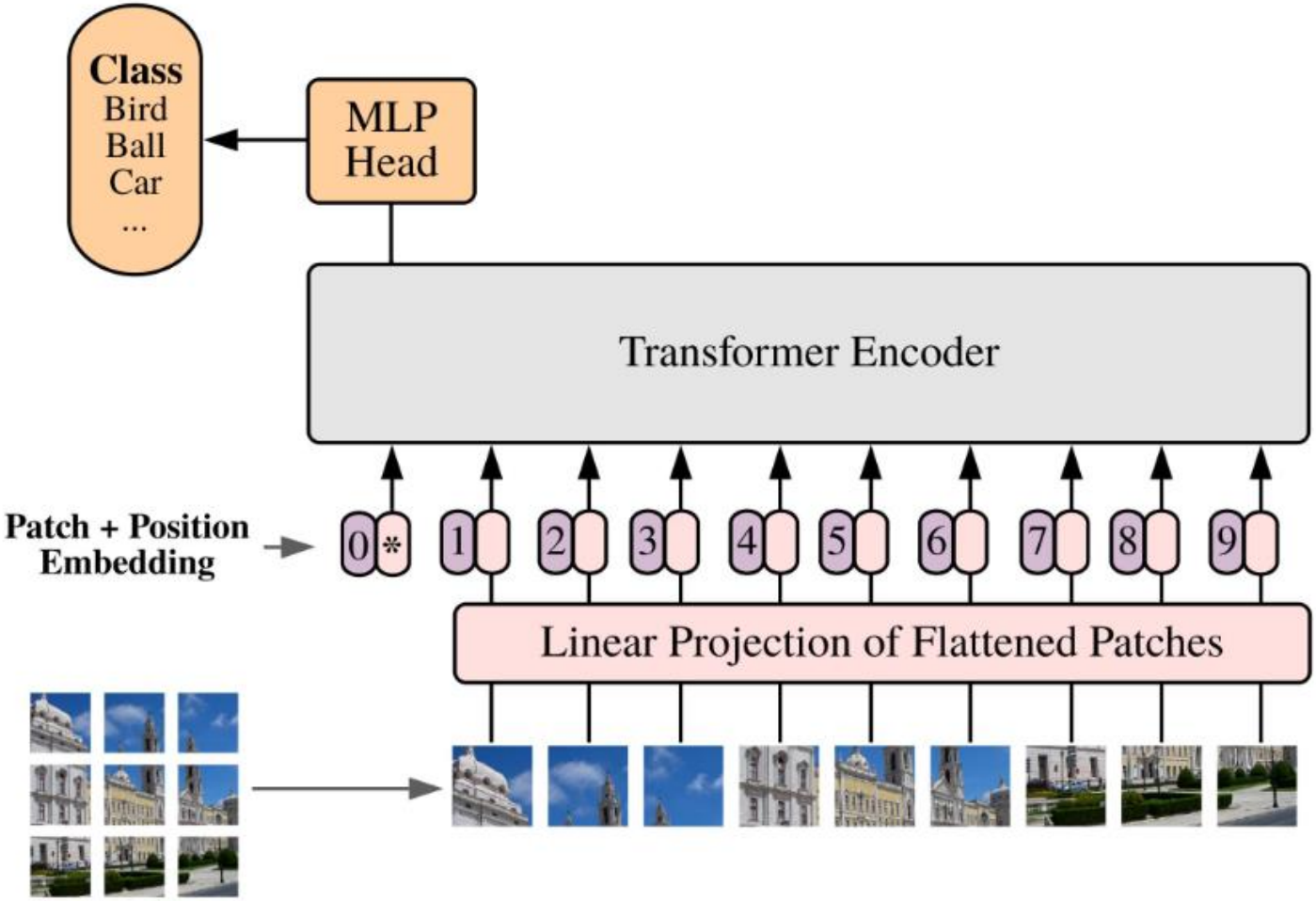
Model Dimensionality: 1024
345M

Model Dimensionality: 1280
762M

Model Dimensionality: 1600
1542M

Visual Transformer

Idea: Representing an image as a sequence



Similar performance to CNNs with less computational load

Visual Language Transformer

Representing coupled vision-language data with a single integrated embedding

- Application: question answering, caption generation

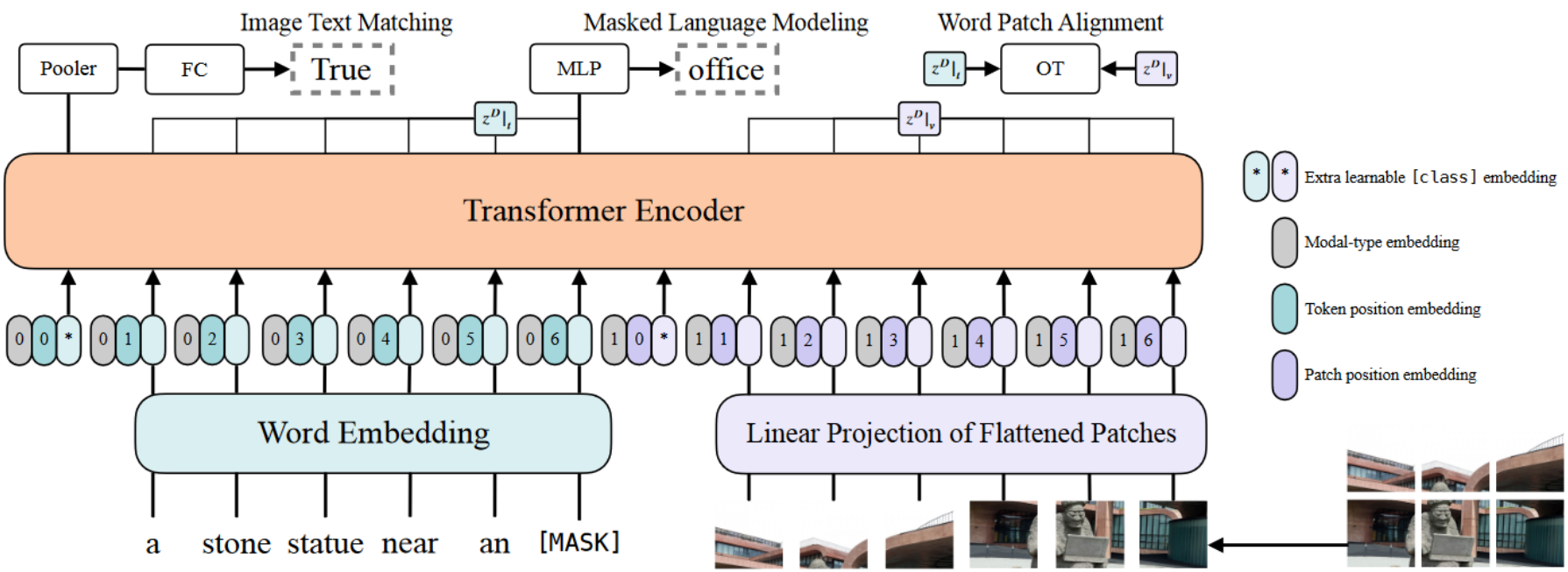


Figure 3. Model overview. Illustration inspired by Dosovitskiy et al. (2020).

Preconditioned Inference in Commonsense Knowledge Reasoning

Ehsan (Eric) Qasemi

Dec 2nd 2022

Overview

- Motivation & Background
  - What is Commonsense Knowledge?
  - What are Preconditions of Common Sense?
- PaCo: Preconditions Attributed to Common Sense Knowledge (EMNLP 2022)
- PInKS: Preconditioned CS Inference with Minimal Supervision (AACL 2022)
- Future/Ongoing Research Directions
  - Visual Commonsense motivation
  - VIPHY: Probing "Visible" Physical Commonsense Knowledge (Under Rev-EACL 2023)
  - Preconditioned Visual Language Inference with Weak Supervision (Under prep-ACL 2023)

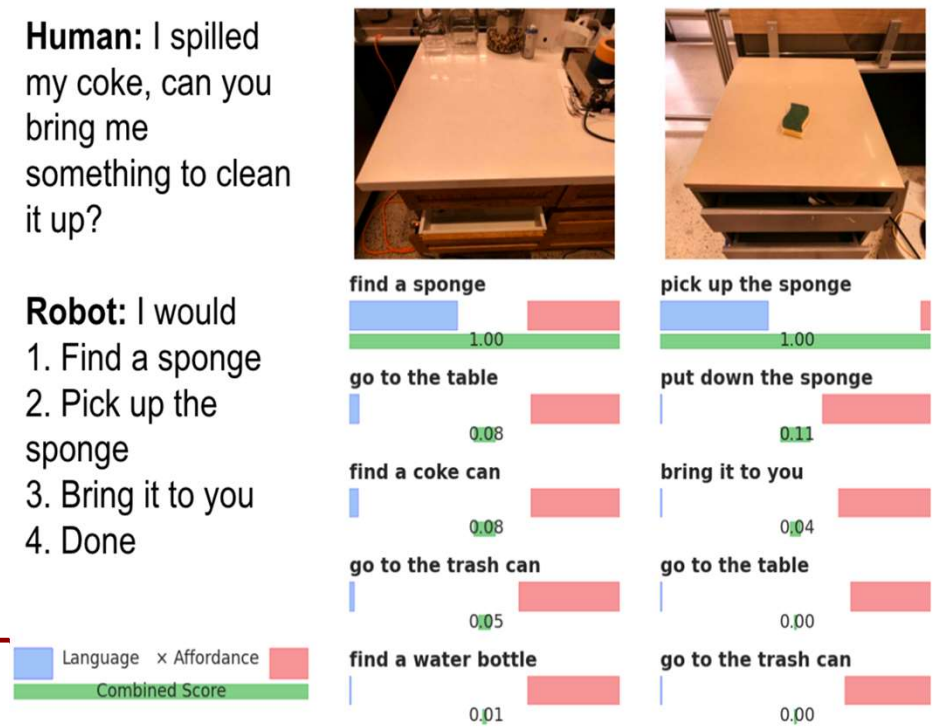
An Example and Some Definitions

- CommonSense Statements:
  - A Glass is used for drinking water.
- Preconditions that Enable the statement
  - You are able to drink
  - You have access to water
  - There is gravity
- Preconditions that Disable the statement
  - The glass is shattered
  - Water is toxic

Why Preconditions are Important?

- Theory of Affordance
  - Understanding the circumstances in which an action or fact is feasible or impossible is a key aspect of human intelligence.

  - In Practice:
    - Ahn, Michael, et al 2022. https://arxiv.org/abs/2204.01691
      - "... generally, with failures as a result of **affordance function misclassification**"



PaCo: Preconditions Attributed to CS Knowledge

- Systematic study on the problem of preconditions expressed in natural language
- Develop PaCo, a rich crowdsourced dataset with enabling and disabling preconditions of commonsense statements
- Extensive NLP benchmarking based on PaCo.
  - Natural Language Inference
  - Multiple-Choice Question Answering
  - Generation

Preconditioned Natural Language Inference

- Paco (our work):
  "You are in a desert" Disables "A net is used for catching fish"
  - Hard preconditions (Enable/Disable) (Causes)
  - Relaxed statement and update format
- **δ**-NLI:
  "The men are facing their granary" strengthens "Two men and a dog are standing among rolling green hills. So the men are farmers"
  - Soft preconditions (strengthen/weaken) (Tends to Cause)
  - Fixed two sentence structure on statement and update

Large language models do not understand preconditions

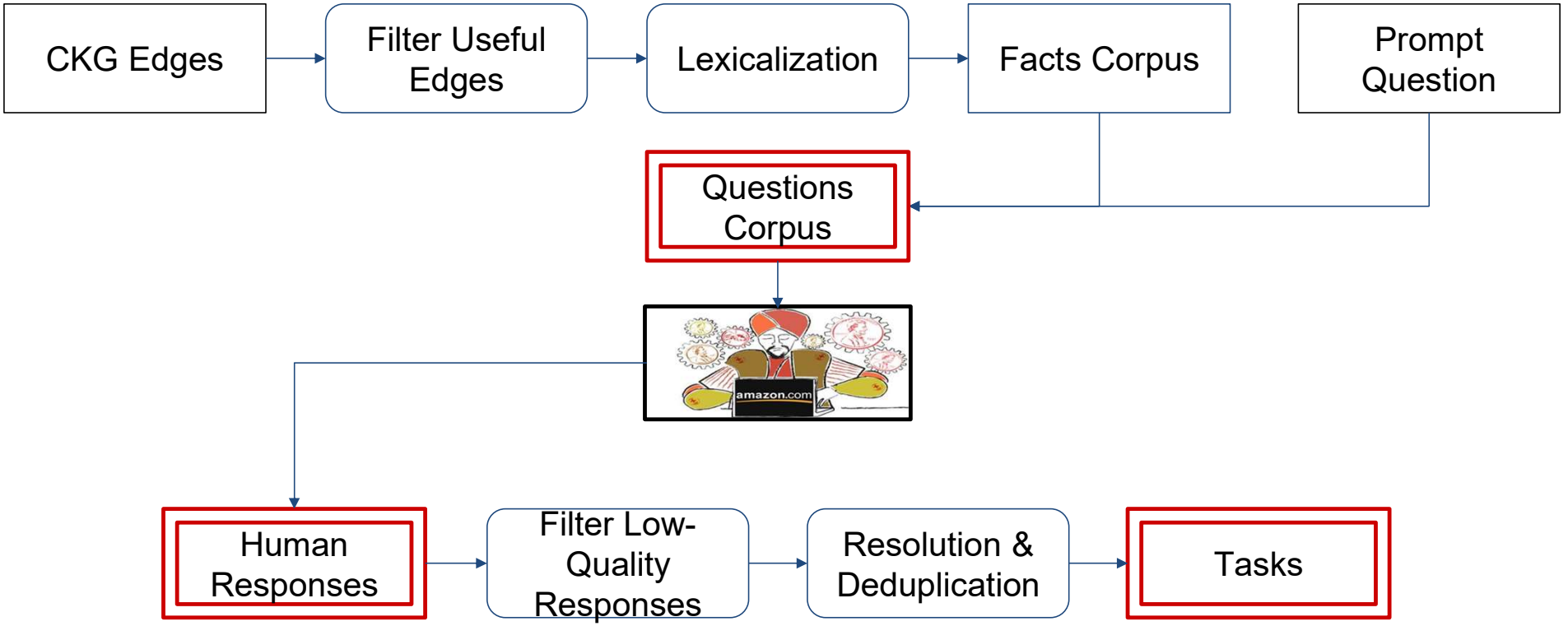
Model	Input	Output
GPT2	A glass is used for drinking water only if	it is being heated inside.
GPT2	A glass is used for drinking water only if, the glass	is covered in a protective coat or can be removed with cold water.
GPT2	A glass is used for drinking water only if, the water	is acidic, not fresh.
GPT2	A glass is used for drinking water. what makes this impossible?	Because drinking water in China is very expensive and water from all over can only be found on a certain area, so what is really happening?"
UnifiedQA-base	A net is used for catching fish. What makes this impossible? (A) you are in water (B) you are in downtown LA	you are in water
UnifiedQA-base	A net is used for catching fish. What makes this impossible?	a net is used for catching fish.

Collecting PaCo

- Statements from ConceptNet
  - Not all statements lend themselves naturally for preconditions annotation
    - E.g. “RelatedTo” is underspecified, and “IsA” is truisms.
  - Three representative types
    - Utility: “UsedFor”
    - Temporal: “Causes”
    - Motivational: “Desires”

- Distantly Supervised Quality Control

Collecting PaCo



Tasks

- Natural Language Inference (P-NLI)
  - Simplest case
- Multiple-Choice Question Answering (P-MCQA)
  - Discriminative without annotation artifacts
  - More Challenging
- Precondition Generation (P-G)
  - The ultimate challenge

ID	Instance
P-NLI	*Hypothesis*: *A net is used for catching fish* *Premise*: *We are in a desert* *Label*: *Contradiction*
P-MCQA	*Question*: *A net is used for catching fish. When is this impossible?* *Choices*: *(A) You are in sea, (B) The boat is moving, (C) Net has a large hole in it.*
P-G	*Question*: *A net is used for catching fish. When is this impossible?* *References*: *(-) Net has a large hole in it, (-) You are in downtown LA, (-) There are no fish in the water*

Table 2: Example of the three tasks in *PaCo*.

Evaluation Results: P-NLI

- Key observations:
  - Even the simple NLI data is not trivial in zero-shot setup

AllenNLP TE	0.34	0.85
RoBERTa-large-MNLI	0.47	0.90
BART-large-MNLI	0.48	0.90
DeBERTa-base-MNLI	0.37	0.91
DeBERTa-large-MNLI	0.36	0.94
DeBERTa-xl-MNLI	0.37	0.91
Expert Human	0.99	-
Random Baseline	0.5	-

Table 3: F1-Macro results of SOTA systems on P-NLI

Evaluation Results: P-MCQA

- Key observations:
  - Significant gap between the human and machine performance
  - LMs lack understanding of linguistic permutations like negations.
  - Large models tend to confuse the enabling v.s. disabling cases.

Model	0-Shot	Tuned
RoBERTa-base	0.24	0.42
RoBERTa-large	0.22	0.22
UnifiedQA-small	0.32	0.50
UnifiedQA-base	0.23	0.59
UnifiedQA-large	0.28	0.68
Expert Human	0.92	-
Random Baseline	0.25	-

Table 4: Accuracy results of SOTA systems on P-MCQA task based on *PaCo*. Best values are highlighted.

Evaluation Results: P-G

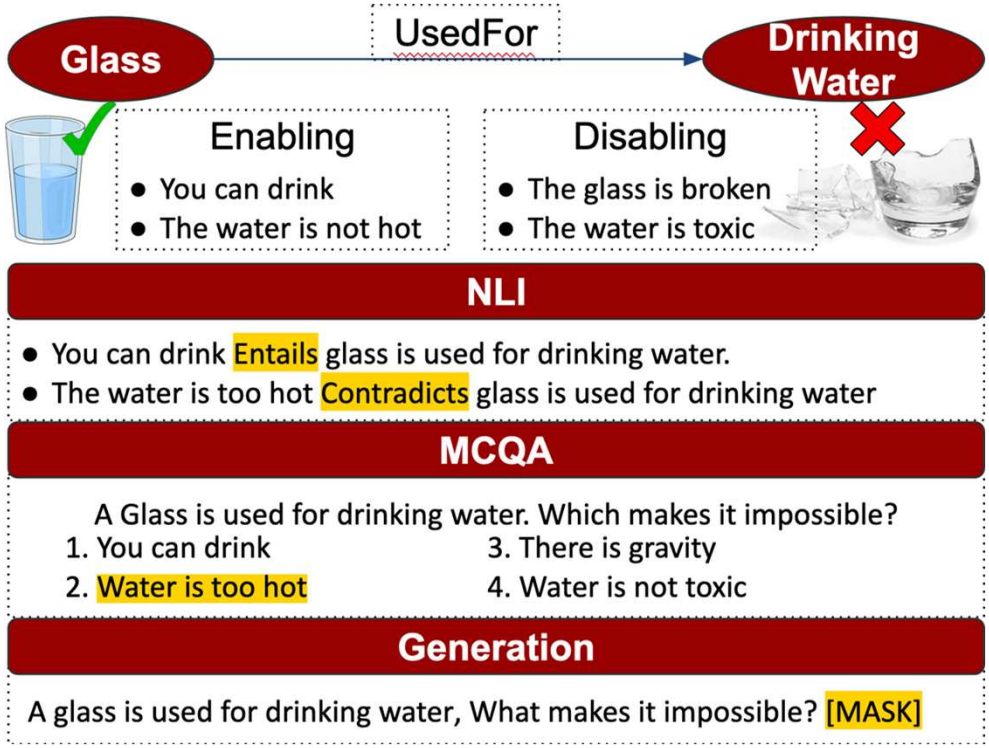
- Key Observations:
  - Fine-tuned models are underperforming
  - Automatic evaluation methods do not sufficiently distinguish between the models
    - Small number of reference responses
    - large space of correct responses
  - Models tend to generate simple answers
    - existence or availability of the subject.

Model	BLEU 0-Shot	BLEU Tuned	ROUGE Tuned	HUM Info.
UnifiedQA-small	0.007	0.157	0.064	0.12
UnifiedQA-base	0.006	0.303	0.115	0.28
UnifiedQA-large	0.029	0.330	0.128	0.48
BART-base	0.046	0.091	0.140	0.19
BART-large	0.041	0.058	0.117	0.11
GPT2	0.097	0.133	0.067	0.36
Expert Human	-	-	-	1.0

Table 5: BLEU-2, ROUGE-2, and human evaluation Information score for results of SOTA systems on the P-G task. Zero-shot ROUGE scores are omitted to save space as they are negligible and do not add additional insight beyond the zero-shot BLEU-2. Best values are highlighted.

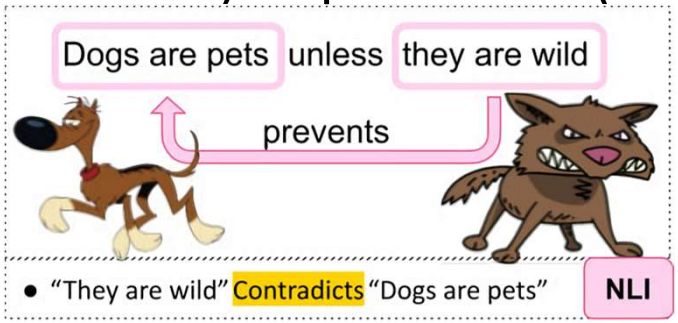
Conclusion of PaCo

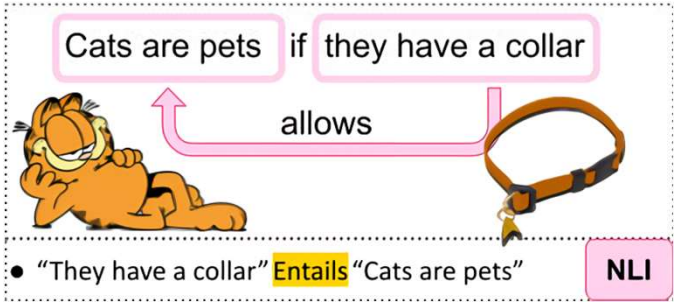
- We saw
  - Reasoning with preconditions
  - PaCo: Data to foster research
  - Evaluation of SOTA
- Challenges:
  - Insufficient annotated data
  - Need for improved LMs



Preconditioned Natural Language Inference

- Conventional NLI: Given a sentence pair composed of a *hypothesis* and a *premise*, the system has to decide whether the hypothesis is true (entailment), false (contradiction), or undetermined (neutral) given the premise
- P-NLI :Given a common sense statement (hypothesis) and an update sentence (premise) that serves as precondition, is the statement still allowed (entailment) or prevented (contradiction)?





Reviewed by ACL Rolling Review (ARR) with meta score of 4/5, committed to EMNLP 2022

Why LLMs are bad at Preconditioned Inference?

- Insufficient annotated data
  - We rarely write commonsense statements and their preconditions
- Need for improved LMs
  - SOTA is trained on raw text with no incentive to focus on preconditions

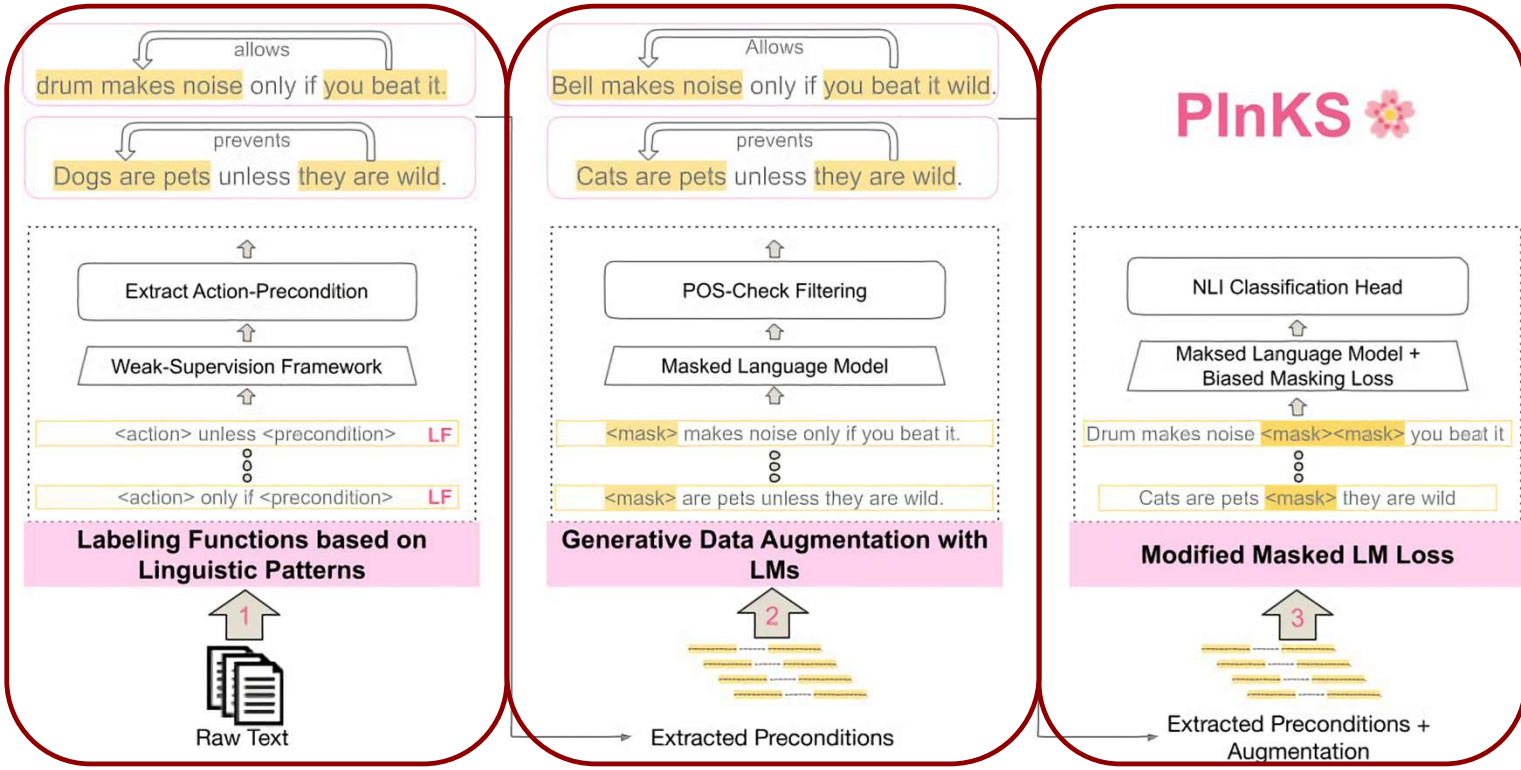
PInKS ✿ : Preconditioned Commonsense Inference with Minimal Supervision

1. Weak supervision for obtaining noisy preconditions data
2. Modify masked LM loss function to emphasize on preconditions
3. Empirical improvements and theoretical informativeness guarantees

In submission to AACL 2022

Improved Model + Weak Data

- Weak supervision for noisy training data
  - Linguistic pattern matching
  - Generative data augmentation
- Biased masking on conjunctions during fine-tuning

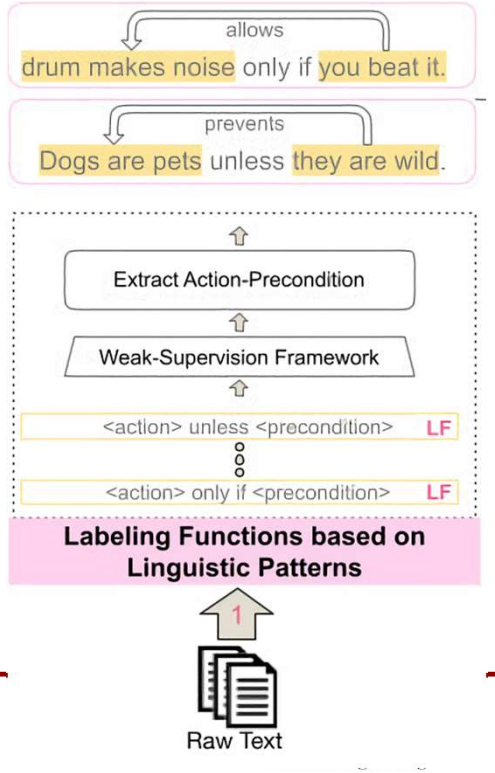


Weak Supervised Data from linguistic patterns

- Linguistic patterns based on conjunctions applied on raw corpora
  - 20+ Patterns (Labeling Functions)
    e.g. “<action> unless <precondition>”
  - OMCS and ASCENT for raw corpora

Text	Label	Action	Precondition
A drum makes noise only if you beat it.	Allow	A drum makes noise	you beat it.
Your feet might come into contact with something if it is on the floor.	Allow	Your feet might come into contact with something	it is on the floor.
Pears will rot if not refrigerated	Prevent	Pears will rot	refrigerated
Swimming pools have cold water in the winter unless they are heated.	Prevent	Swimming pools have cold water in the winter	they are heated.

Table 1: Examples from the collected dataset through linguistic patterns in §3.1.



Labeling Functions base on Conjuctions

Type	Conjunctions
Allowing	only if, subject to, in case, contingent upon, given, if, in the case that, in case, in the case that, in the event, on condition, on the assumption, only if, so, hence, consequently, on these terms, subject to, supposing, with the proviso, so, thus, accordingly, therefore, as a result, because of that, as a consequence, as a result
Preventing	but, except, except for, excepting that, if not, lest, saving, without, unless

Table 3. List of conjunctions used in modified masked loss function in section 3.3

Analysis on Weak Supervision

Conjunctions	Recall	Pattern
but	0.17	{action} but {negative_precondition}
contingent upon	0.6	{action} contingent upon {precondition}
except	0.7	{action} except {precondition}
except for	0.57	{action} except for {precondition}
if	0.52	{action} if {precondition}
if not	0.97	{action} if not {precondition}
in case	0.75	{action} in case {precondition}
in the case that	0.30	{action} in the case that {precondition}
in the event	0.3	{action} in the event {precondition}
lest	0.06	{action} lest {precondition}
makes possible	0.81	{precondition} makes {action} possible.
on condition	0.6	{action} on condition {precondition}
on the assumption	0.44	{action} on the assumption {precondition}
statement is true	1.0	The statement "{event}" is true because {precondition}.
supposing	0.07	{action} supposing {precondition}
to understand event	0.87	To understand the event "{event}", it is important to know that {precondition}.
unless	1.0	{action} unless {precondition}
with the proviso	-	{action} with the proviso {precondition}
on these terms	-	{action} on these terms {precondition}
only if	-	{action} only if {precondition}
make possible	-	{precondition} makes {action} possible.
without	-	{action} without {precondition}
excepting that	-	{action} excepting that {precondition}

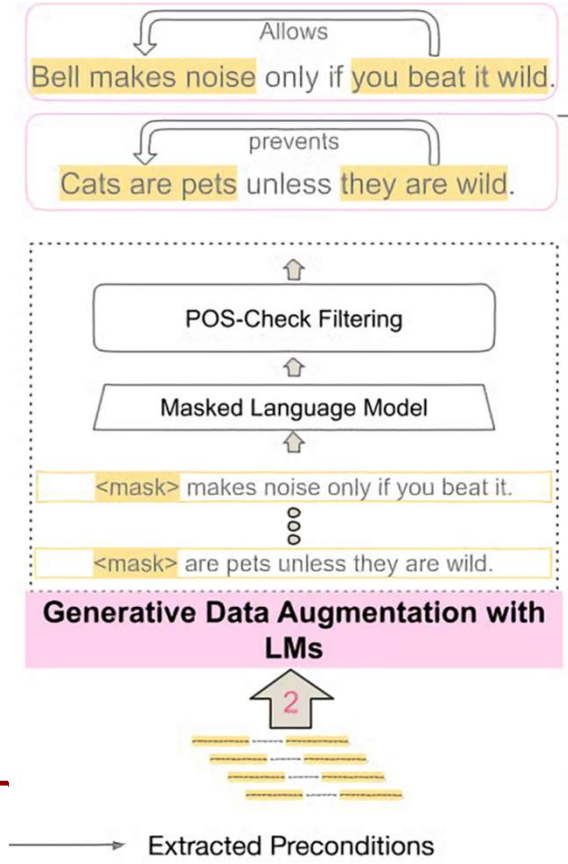
Table 2. Linguistic patterns in *PInKS* and their recall value. For patterns with not enough match in the corpora have empty recall values.

Analysis on Weak Supervision

- Weak Data Stats:
  - Linguistic Patterns: 102,474 Allow and 10,921 Prevent assertions
  - 7M labeled action-precondition pair with 11% prevent preconditions

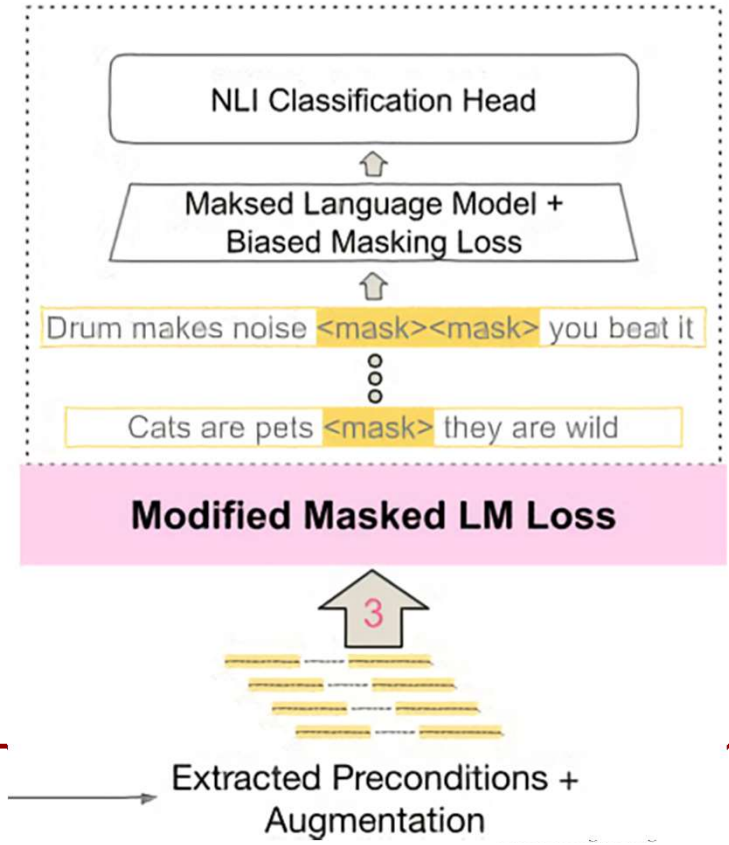
Weak Supervised Data from Generative Augmentation

- Generative data augmentation using masked language models
- POS Tag consistency as filtering mechanism
  - RoBERTa-base (<MASK> will rot if not refrigerated):
    - Food
    - ~~It (Filtered out based on POS tag)~~
    - Meat
    - Fish
    - Chicken



Modified Language Modeling

- Biased masking on conjunctions
  - Only mask conjunctions instead of random tokens
  - Additional fine-tuning on the weak supervised data (no labels) to fill mask
    - LM(Food will rot <mask> <mask> refrigerated)



Main Results: Experimental Setup

- Five target tasks in NLI canonical format derived from supervised resources on preconditions of commonsense
  - RoBERTa-Large-MNLI as model

Name	Original Data		Derived NLI	
Winoventi (Do and Pavlick, 2021)	**masked_prompt**:	Margaret smelled her bottle of maple syrup and it was sweet. The syrup is {MASK}.	**Hypothesis**:	Margaret smelled her bottle of maple syrup and it was sweet.
	target:	edible	**Premise**:	The syrup is edible/malodorous
	incorrect:	malodorous	**Label**:	ENTAILMENT/CONTRADICTION
ANION (Jiang et al., 2021)	**Orig_Head**:	PersonX expresses PersonX's delight.	**Hypothesis**:	Alice expresses Alice's delight/anger.
	Relation:	xEffect	**Premise**:	feel happy.
	Tail:	Alice feel happy	**Label**:	ENTAILMENT/CONTRADICTION
	Neg_Head:	PersonX expresses PersonX's anger.		
ATOMIC2020 (Hwang et al., 2020)	**Head**:	PersonX takes a long walk.	**Hypothesis**:	PersonX takes a long walk.
	Relation:	HinderedBy	**Premise**:	It is 10 degrees outside..
	Tail:	It is 10 degrees outside.	**Label**:	CONTRADICTION
δ-NLI (Rudinger et al., 2020)	**Hypothesis**:	PersonX takes a long walk.	**Hypothesis**:	PersonX takes a long walk.
	Premise:	HinderedBy	**Premise**:	It is 10 degrees outside..
	Update:	It is 10 degrees outside.	**Label**:	CONTRADICTION
	Label:	Weakener		
PaCo (Qasemi et al., 2022)	**Statement**:	A net is used for catching fish.	**Hypothesis**:	A net is used for catching fish.
	Precondition:	You are in a desert.	**Premise**:	You are in a desert.
	Label:	Disabling	**Label**:	CONTRADICTION

Table 8: Examples from target tasks in NLI format

Main Results

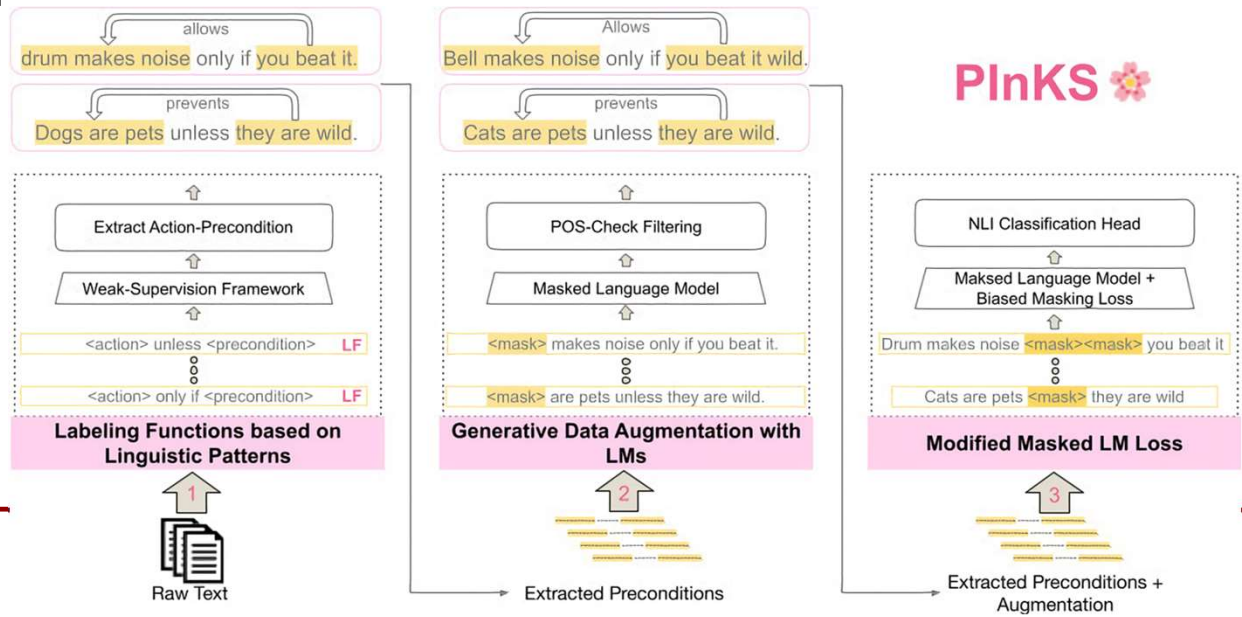
- Takeaway:
  - Exceeds the supervised results in zero-shot (Atomic, Winoventi)
  - Outperforms when combined with supervised signals (δ-NLI, ANION, ATOMIC)
  - No improvement on the hard preconditions (CoreQuisite/PaCo)
- Setups:
  - Orig.
    - tune on train, eval on test
  - PInKS
    - PINKS eval on test
  - Orig+PInKS
    - tune PInKS on train, eval on test

Target Task	Orig.	*PInKS*	Orig+*PInKS*
δ-NLI	83.4	60.3	**84.1**
CoreQuisite	77.1	69.5	68.0
ANION	81.1	52.9	**81.2**
ATOMIC	43.2	**48.0**	**88.6**
Winoventi	51.1	**52.4**	**51.3**

Table 2: Macro-F1 (%) results of *PInKS* on the target datasets: no *PInKS* (*Orig.*), with *PInKS* in zero-shot transfer learning setup (*PInKS*) and *PInKS* in addition to original task's data (*Orig.+PInKS*). **Bold** values are cases where *PInKS* is improving supervised results.

Summary

- We saw
  - Preconditions and Preconditioned Inference
  - Weak supervision methods to obtain data
  - Biased masking to improve LMs
  - Theoretical and Empirical improvements



Takeaways

- Preconditioned inference is inherently difficult tasks with vast applications
- Conjunctions can serve as proxy for models to comprehend preconditions
- Inference with hard preconditions tend to be more challenging
  - Enough weak supervision can help but needs huge computation and more fine-grained LFs

Visual Commonsense is the Viable Path

- Future/Ongoing Research Directions
  - Visual Commonsense motivation
  - VIPHY: Probing “Visible” Physical Commonsense Knowledge (Under Rev-EACL 2023)
  - Preconditioned Visual Language Inference with Weak Supervision (Under prep-ACL 2023)

References

- [ASCENT] Nguyen, Tuan-Phong, Simon Razniewski, and Gerhard Weikum. "Advanced semantics for commonsense knowledge extraction." Proceedings of the Web Conference 2021. 2021.
- [OMCS] Havasi, Catherine, et al. "Open mind common sense: Crowd-sourcing for common sense." Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.
- [Snorkel] Ratner, Alexander, et al. "Snorkel: Rapid training data creation with weak supervision." Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases. Vol. 11. No. 3. NIH Public Access, 2017.
- [PaCo] Ehsan Qasemi, Filip Ilievski, Muhao Chen, and Pedro Szekely. 2022. Paco: Circumstantial preconditions of common sense knowledge. arXiv preprint arXiv:2104.08712.
- [**δ**-NLI] Rudinger, Rachel, et al. "Thinking like a skeptic: Defeasible inference in natural language." Findings of the Association for Computational Linguistics: EMNLP 2020. 2020.
- [ANION] Jiang, Liwei, et al. "" I'm Not Mad": Commonsense Implications of Negation and Contradiction." arXiv preprint arXiv:2104.06511 (2021).
- [Winoventi] Do, Nam, and Ellie Pavlick. "Are Rotten Apples Edible? Challenging Commonsense Inference Ability with Exceptions." Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. 2021.
- [ATOMIC2020] Hwang, Jena D., et al. "(Comet-) Atomic 2020: On Symbolic and Neural Commonsense Knowledge Graphs." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 7. 2021.
- [PaBi] He, Hangfeng, et al. "Foreseeing the benefits of incidental supervision." arXiv preprint arXiv:2006.05500 (2020).

Thank you

Understanding and Generating Multimodal Feedback in Human-Machine Story-Telling

Setareh Nasihati Gilani

November 2022

USC University of Southern California

Outline

- Introduction
- Completed Work Part 1: Story-Telling Agent
- Completed Work Part 2 : RAVE
- Completed Work Part 3 : Identifying feedback Opportunities
- Proposed Work

Definitions-Terminology

- Feedback (in the context of interaction)
  - response of a receiver to sender's message
  - Can be verbal, nonverbal
- Story
  - A story or narrative is a connected series of events told through words (written or spoken), imagery (still and moving), body language, performance, music, or any other form of communication.
- Story-telling context
  - An interaction which involves a story told by an interlocuter

Motivating Examples

Outline

- Introduction
  - Motivation
  - Research Questions
  - Approach
  - Contributions

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context

Motivation

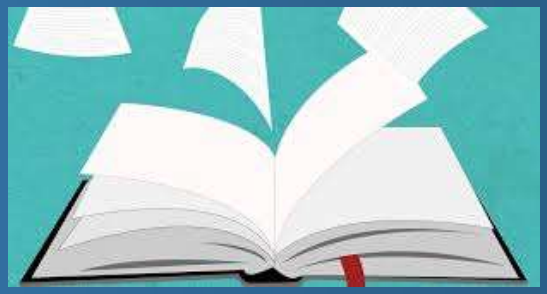
- Feedback is an important part of any interaction
  - Feedback can be multimodal (verbal, visual, audio, …)
  - Used by dialogue partners as evidence of their success
  - Can be positive / negative
  - Come naturally for humans

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
  - Used to establish identity
  - Pass on cultural heritage
  - Build Rapport



Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important

→ One might try to regain attention, or cut the story short upon receiving negative feedback

→ One might want to go into detail about one specific part upon receiving positive feedback

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context

Motivation

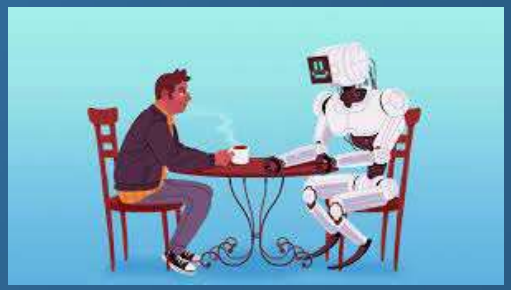
- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
  - Smart assistants on phones, laptops, …
  - Assistive robots for homes
  - Smart Devices, Internet of things
  - Companion robots for elderly

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context

Motivation

- Feedback is an important part of any interaction
- Telling Stories is important in interactions
- Feedback in story-telling context is important
- Human-Machine Interactions are rapidly increasing in different aspects of our lives
- Machines should be able to use feedback within story-telling context



Examples of Machines who use stories

- New Dimensions in Testimony
  - Story-teller agent
- Simsensei kiosk
  - Elicits and listens to stories

Outline

- Introduction
- Completed Work Part 1: Story-Telling Agent
- Completed Work Part 2: RAVE
- Completed Work Part 3: Identifying feedback Opportunities
- Proposed Work

Story-Telling Agent

RQ 1: What kinds of stories should virtual storytellers provide?

- Different story-designs
  - Context
  - Audience
  - Goals
- Investigated the answer in a specific scenario

What kind of stories?

- Goals
  - Building long-term rapport
  - Desire to keep interacting with the system
- Which types of stories will best achieve the goal?
- Ways that stories differ
  - Taxonomy of Collins & Traum includes 5 main dimensions with several sub-dimensions (LREC 2016)
    - Point of View/Perspective (first, second, third person)
    - Identity (explicit/unspecified, types of explicit)

Specifying types of stories

- Perspective: First person stories or third person?
- Identity: Human backstory or artificial?
  - Are human oriented stories plausible coming from a virtual human?
  - Is there a backfire effect when virtual humans claim to have human like experiences?

Related Work

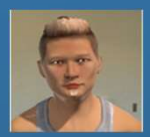
- Perspective
  - Virtual exercise counselor agent (Bickmore et al IVA 2009)
    - Tells inspiring stories about weight loss using either 1st person or 3rd person
    - Results: First person participants were likely to report greater enjoyment
- Identity:
  - Robot exercise buddy (Swift-Spong et al Ro-man 2016)
    - Robot or space alien identity backstory
    - Results: no significant difference

Experiment

- Engage with users in a simple ‘get to know you’ dialogue
- Swap stories for a set of ‘ice-breaker’ questions
- Independent Variables
  - Perspective (1st person vs 3rd person stories)
  - Identity (Human vs Virtual human)
  - Embodiment (Human video vs Virtual human)
- Investigated degree of reciprocity and compared rapport

Dialogue conditions

Identity

Embodiment	**Virtual (VH)**	**Human (H)**	
Human (H)		H – H – 1st	← 1st person
		H – H – 3rd	← 3rd person
Virtual (VH)	VH – VH – 1st	VH – H – 1st	← 1st person
	VH – VH – 3rd	 VH – H – 3rd	← 3rd person

Example story (VH – VH – 3rd)

- I'm a virtual human at ICT, which means that I get to talk and interact with people like you. It's a really great job. I get to talk with some intriguing people and hear about all kinds of interesting experiences. This woman the other day was telling me about how she had ruined her last 8 laptops in the same way; by spilling Greek yogurt all over the keyboard. To my relief, she said that she does not eat snacks around computers anymore, which I would probably say it's a good thing.

Participant interactions

- 60 participants assigned randomly to one of the 3 conditions of a between-subject factor
- VH-VH, VH-H, H-H
- They interacted with both perspective agents in that condition (within-subject factor): 1st person, 3rd person
- Counter balancing order
- Interaction episodes
  - Participant asks questions
  - Agent answers and follow up with a reciprocal question (Like "What about you?")
  - User answers the question himself

Video example of interaction

Experiment Metrics

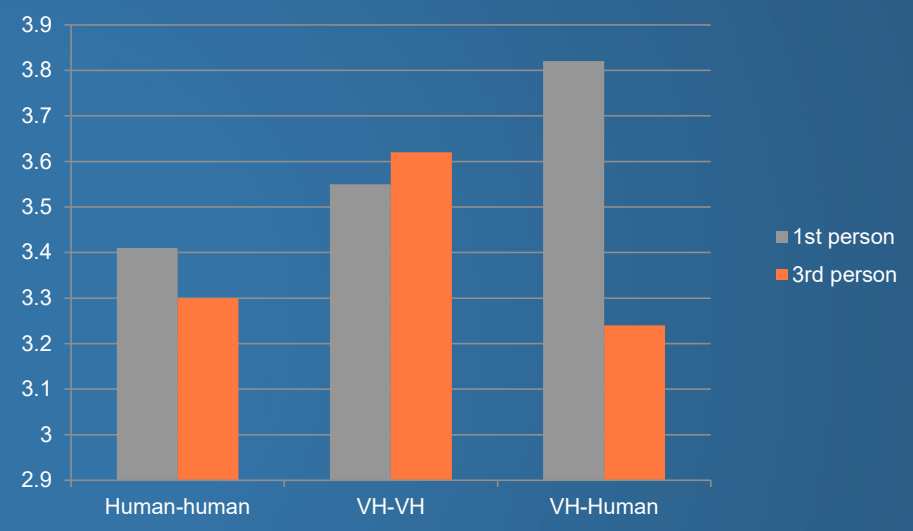
- 9-item rapport scale
  - Eg: I felt I had a connection with my partner.
- 6-item ancillary rapport scale
  - Eg: How much did you like the character?
- Two items on subjective sharing of personal information
  - How much did you reveal about yourself in your answers?
  - How personal were your answers?
- A set of 30 personality characteristics
  - Cheerful, threatening, aloof
- Length of participant responses
- Number of participant responses containing stories

Results

- Conducted a 2 (perspective) * 2 (order) * 3 (agent) mixed ANOVAs
  - Order and agent as between-subject factors
  - Perspective as within-subject factor

Results (continued)

- First Person: More Rapport
  - Marginally significant main effect of perspective exists
  - $F(1, 53) = 3.21$, $p = .08$.
  - 1st person agent ($M = 3.61$, $SE = 0.09$)
  - 3rd person agent ($M = 3.42$, $SE = 0.09$).



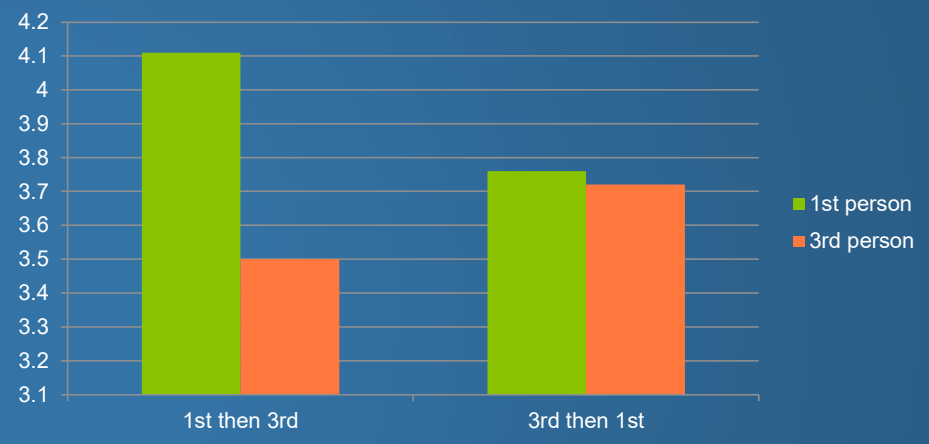
Interaction of perspective and agent on the Rapport Scale.

Results (continued)

- First Person: More Rapport
- First Person: More disclosure of personal information
  - significant main effect of perspective, $F(1, 53) = 6.88$
  - 1st person agent ($M = 3.92$, $SE = 0.12$)
  - 3rd person agent ($M = 3.65$, $SE = 0.14$).

Results (continued)

- First Person: More Rapport
- First Person: More disclosure of personal information
- First Person: More cheerful
  - Marginally significant main effect of perspective, $F(1, 53) = 3.33$, $p = 0.07$
  - 1st person agent ($M = 1.66$, $SE = 0.1$)
  - 3rd person agent ($M = 1.89$, $SE = 0.11$)



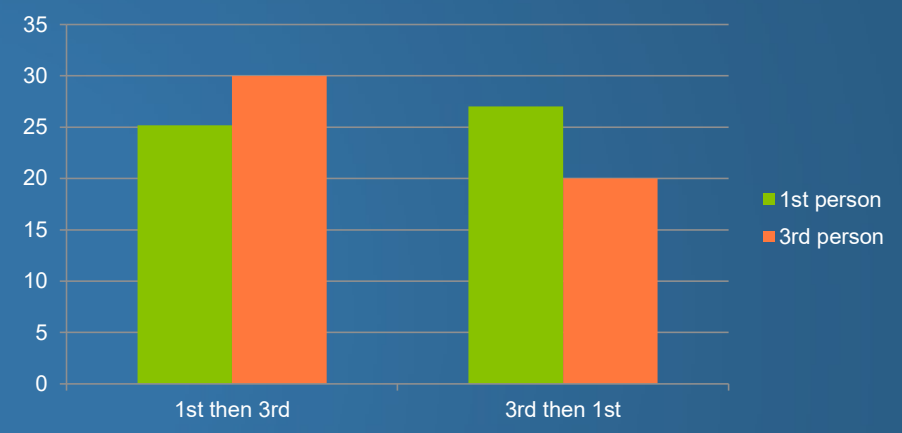
Interaction of perspective and order on the personality characteristic “cheerful"

Results (continued)

- First Person: More Rapport
- First Person: More disclosure of personal information
- First Person: More cheerful
- First Person: More trustworthy
  - significant main effect of perspective, $F(1, 52) = 5.67$; $p = 0.02$
  - 1st person agent: (M = 3.95; SE 0.13)
  - 3rd person agent: (M = 3:55; SE = 0:16).

Results (continued)

- First Person: More Rapport
- First Person: More disclosure of personal information
- First Person: More cheerful
- First Person: More trustworthy
- No difference in length of response



Interaction of perspective and order on the length participant talked

Results

- First person: More rapport
  - Marginally significant: p=0.08
- First Person: More disclosure of personal information
  - Significant: p=0.01
- First Person: More cheerful
  - Marginally significant: p=0.07
- First Person: More trustworthy
  - Significant: p=0.02

Takeaway w.r.t. Research question

RQ 1: What kinds of stories should virtual storytellers provide?

- We see a general preference for First person over third person stories
- Fine to tell 3rd person stories as well
- No significant difference for identity
- Also shows the necessity for having feedback in interaction

Outline

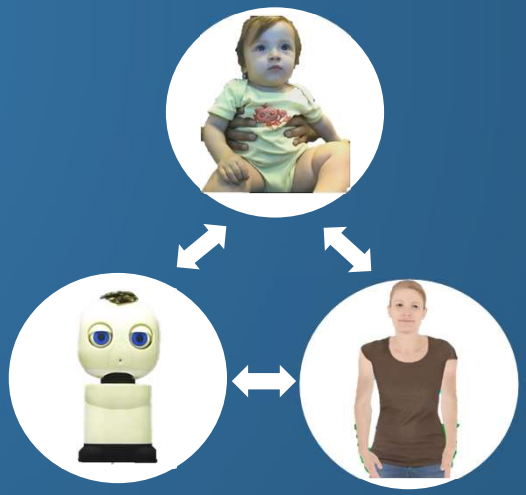
- Introduction
- Completed Work Part 1: Story-Telling Agent
- Completed Work Part 2: RAVE
- Completed Work Part 3: Identifying feedback Opportunities
- Proposed Work

Robot Avatar Thermal Enhanced System (RAVE)

RQ 2: What kinds of multimodal feedback can agents recognize and how can they be used to adapt the dialogue management policies of the agent?

→ Designed and implemented the dialogue management for a multimodal multiparty system called RAVE

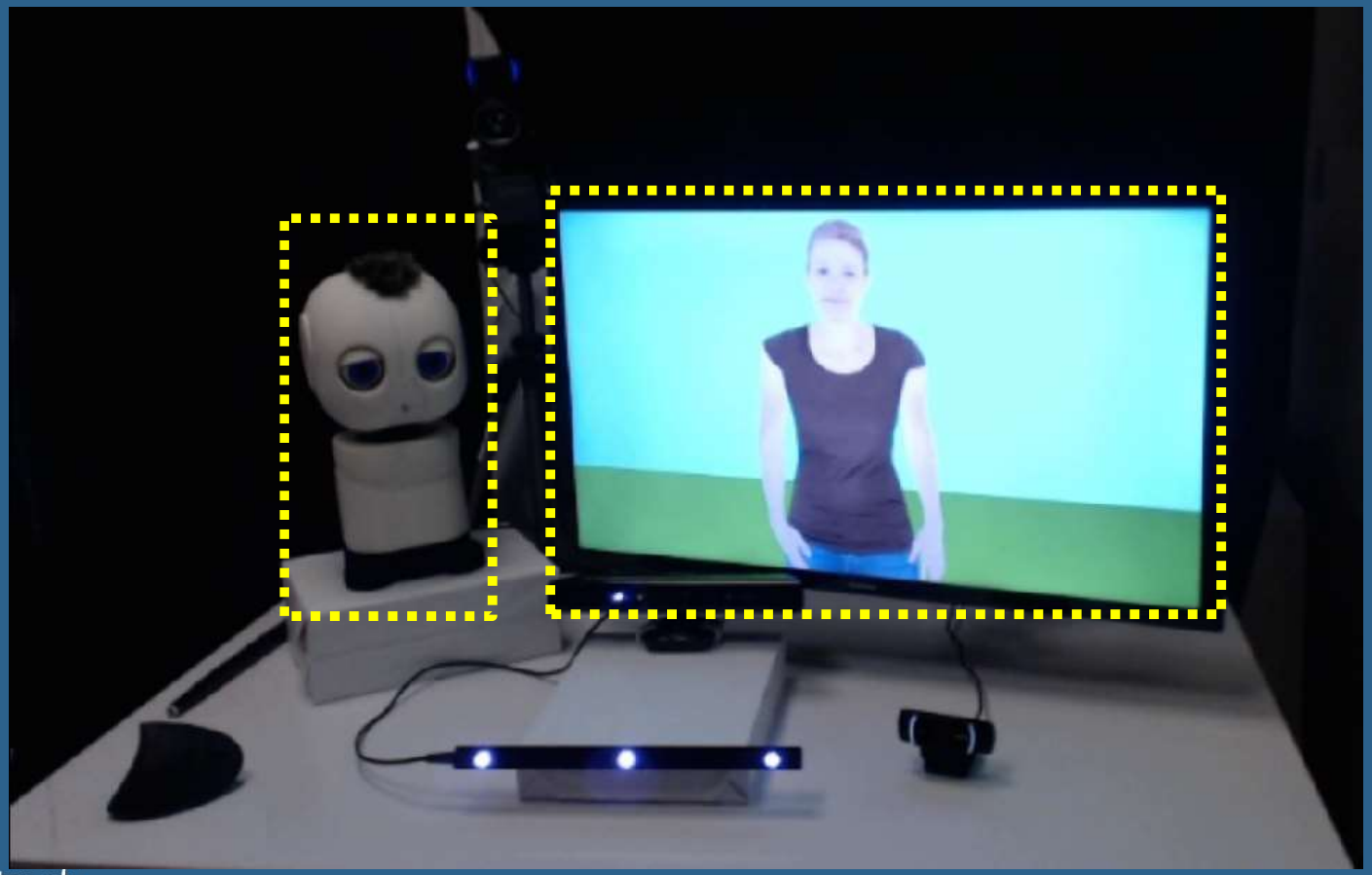
- → Builds a 3-way interaction between a robot, a virtual human and the baby
- → Provides Stories (nursery rhymes) and adapt the flow based on baby's feedback



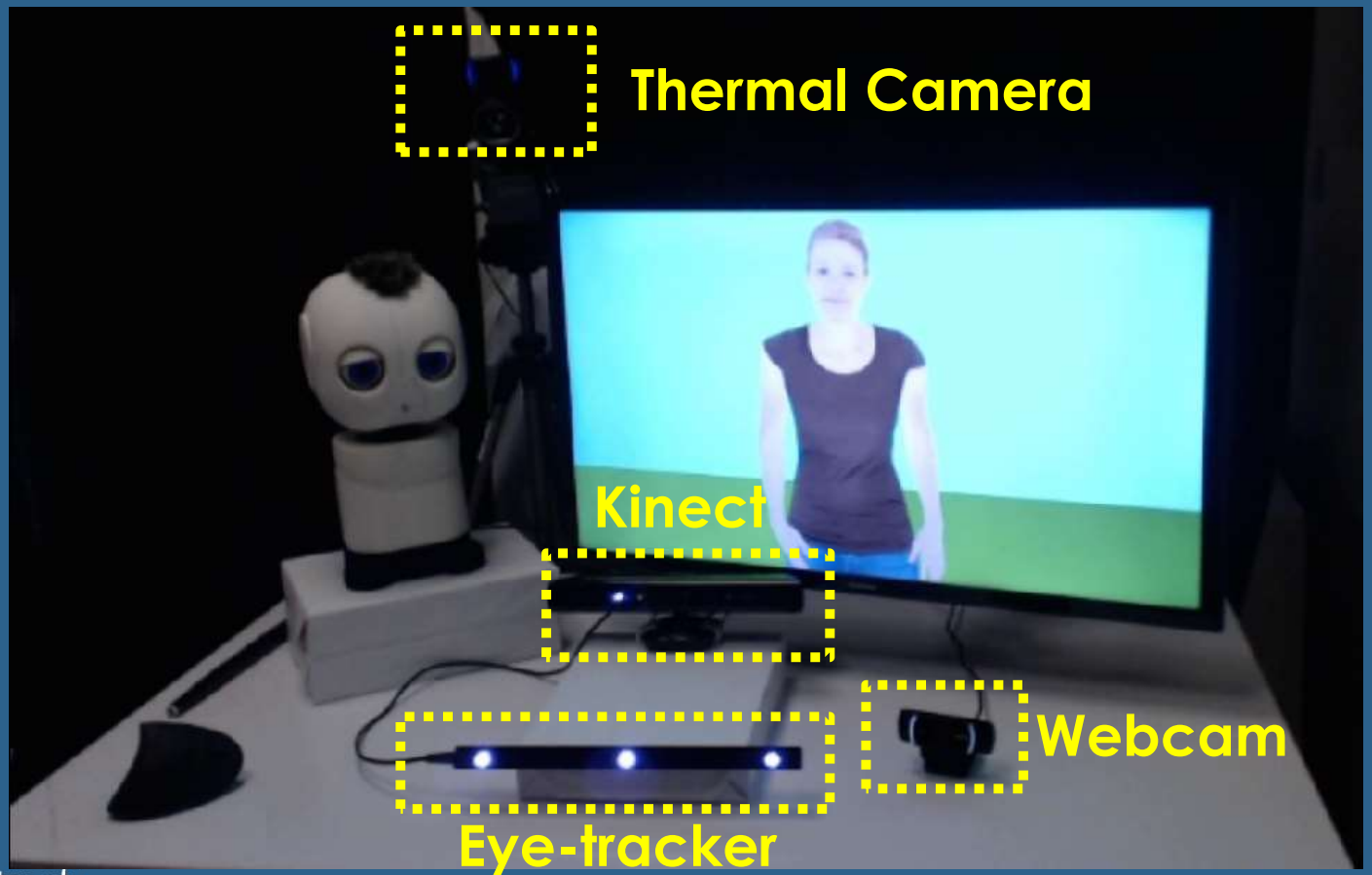
USC University of Southern California

RAVE - Robot, AVatar, thermal-Enhanced System

RAVE - Robot, AVatar, thermal-Enhanced System



RAVE - Robot, AVatar, thermal-Enhanced System

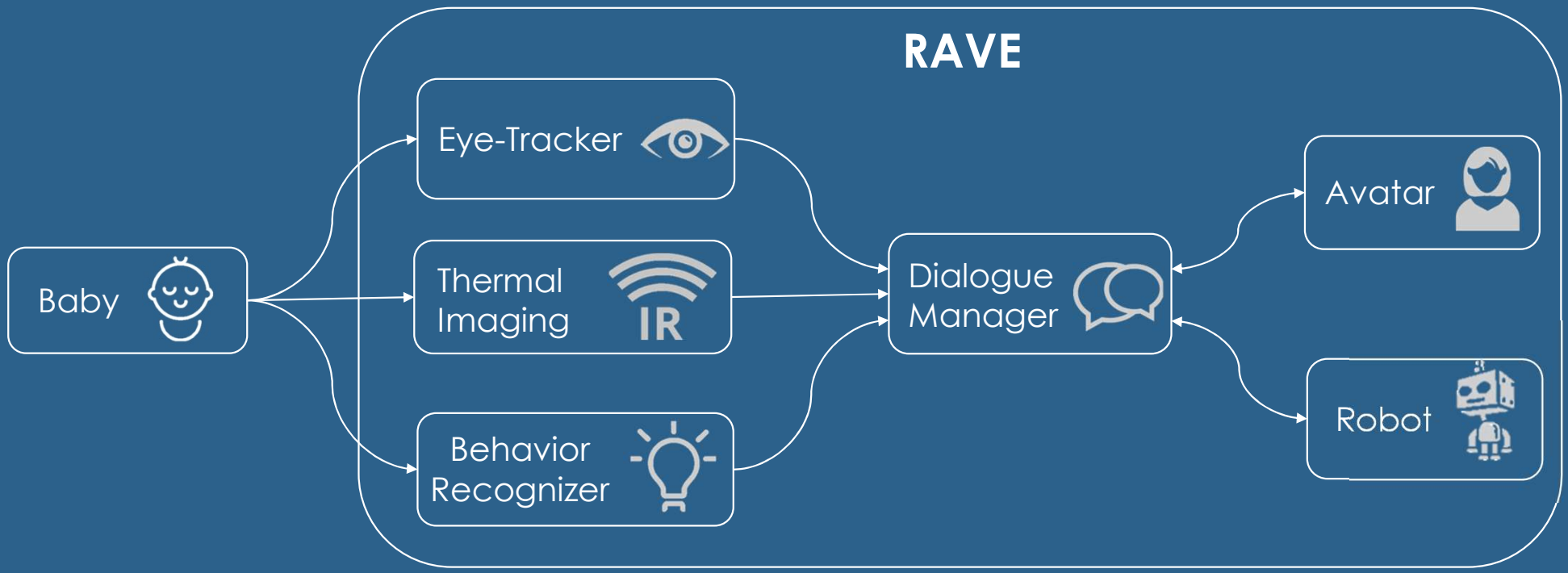


Why RAVE? Background and Motivation

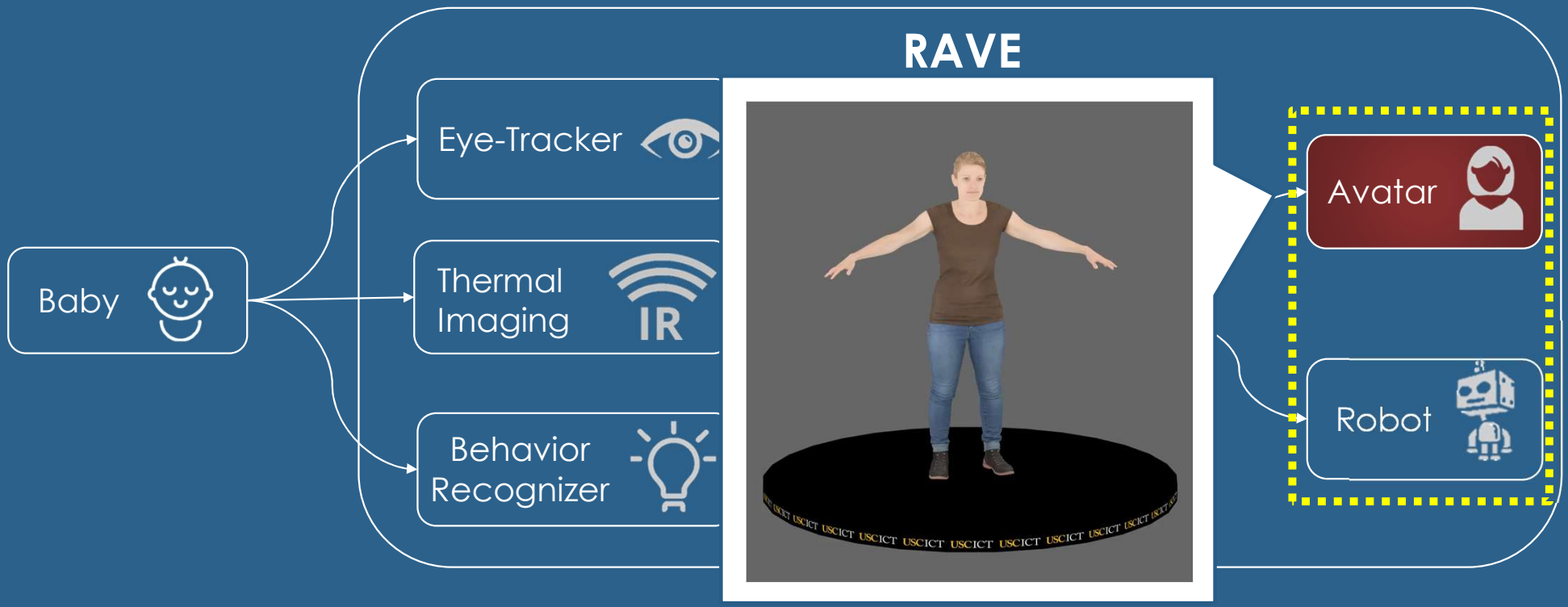
- Language Exposure during early life
- Learning the phonetic units of language by 6-10 months
- Children with minimal or delayed exposure are at risk
- Sign Language has also phonology!
  - Same patterns of phonological development for signed languages
- Deaf/Hard of hearing babies are at risk
- How can technology help?

Examples

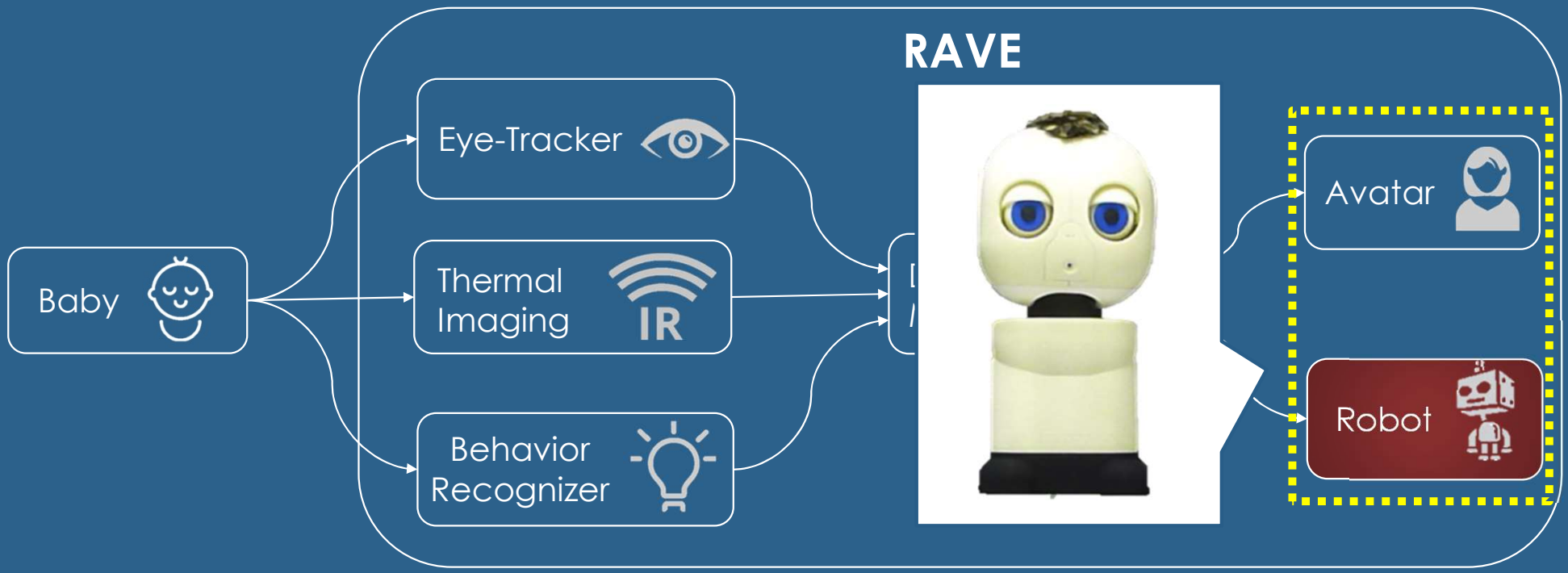
RAVE - Robot, AVatar, thermal-Enhanced System



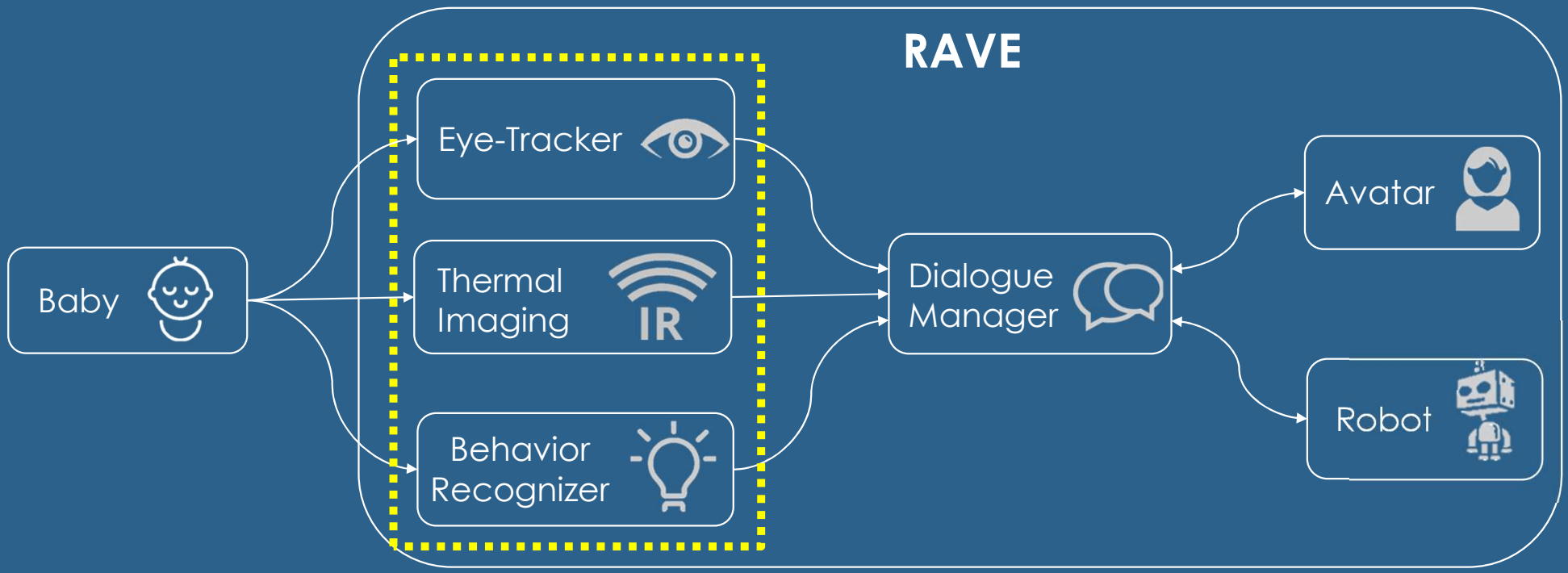
RAVE - Robot, AVatar, thermal-Enhanced System



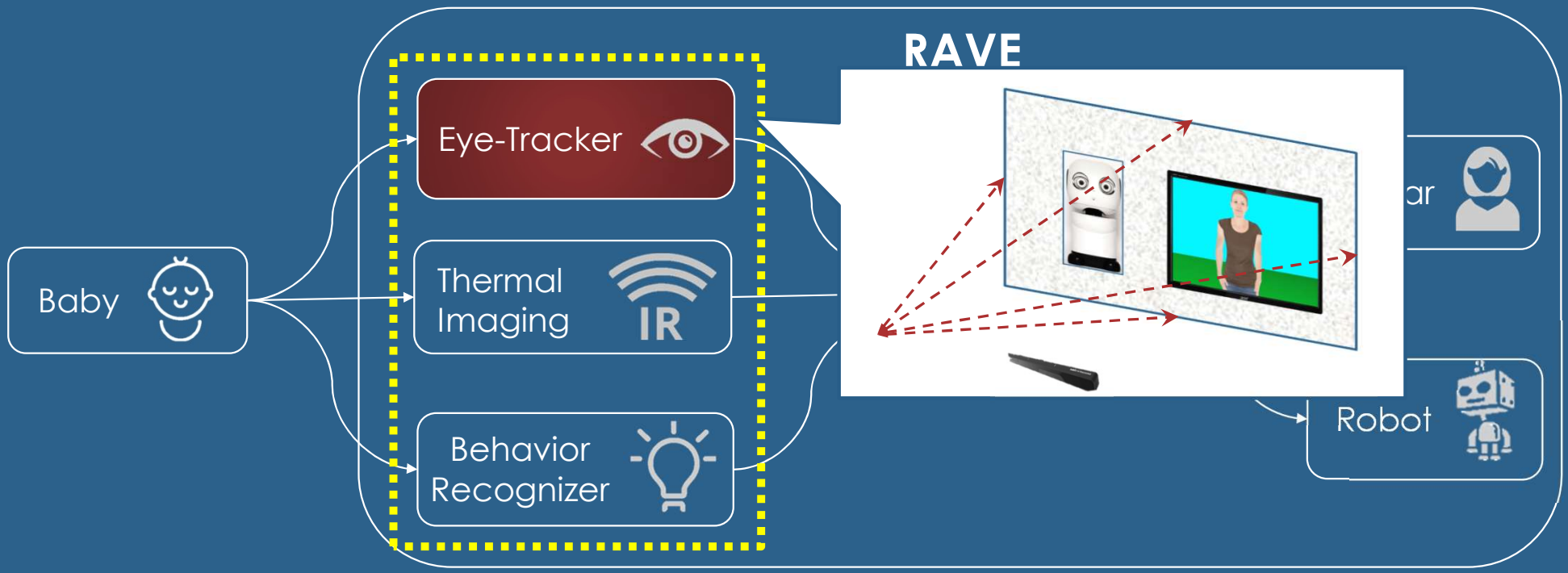
RAVE - Robot, AVatar, thermal-Enhanced System



RAVE - Robot, AVatar, thermal-Enhanced System

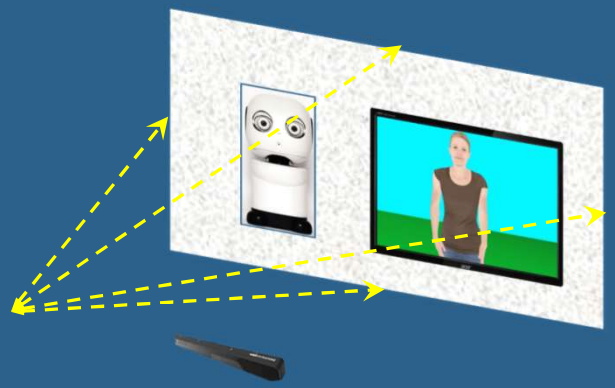


RAVE - Robot, AVatar, thermal-Enhanced System

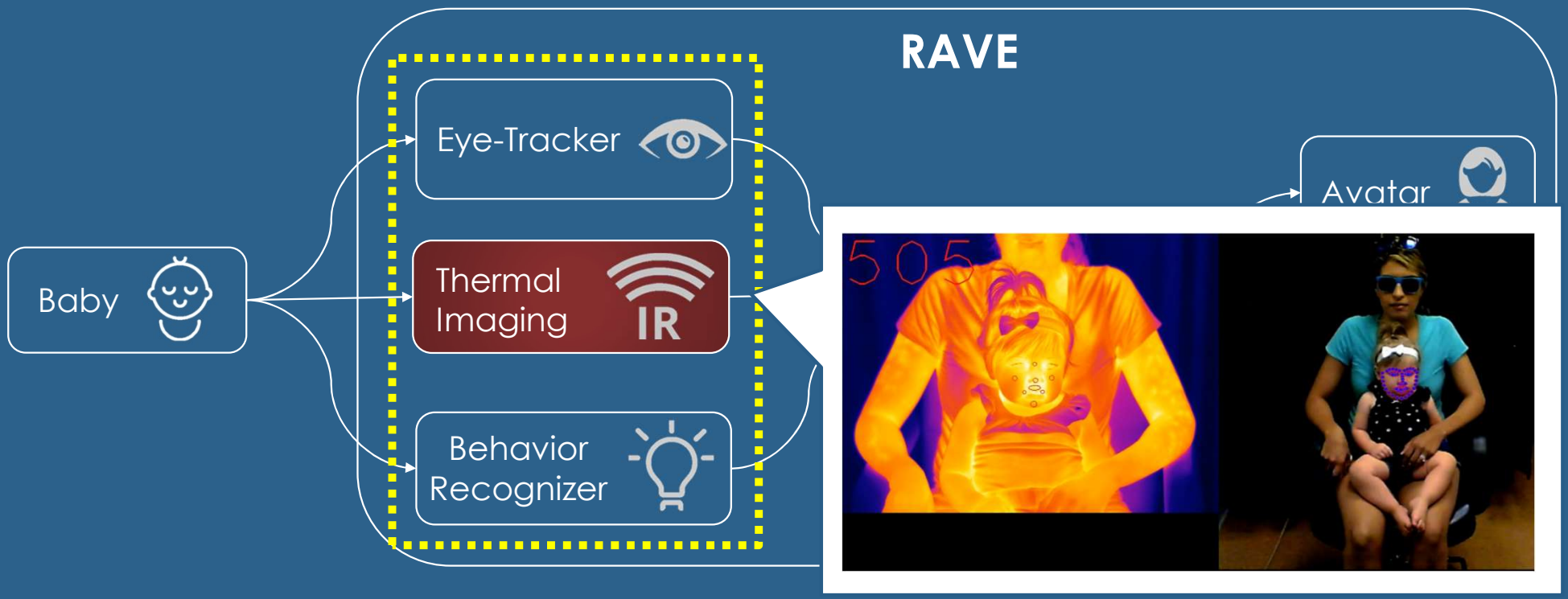


Eye-Tracker

- Tobii Pro X3-120
- Area of Interests (AOIs) for robot and Avatar
- Majority focus position of the baby's gaze every ¼ second



RAVE - Robot, AVatar, thermal-Enhanced System



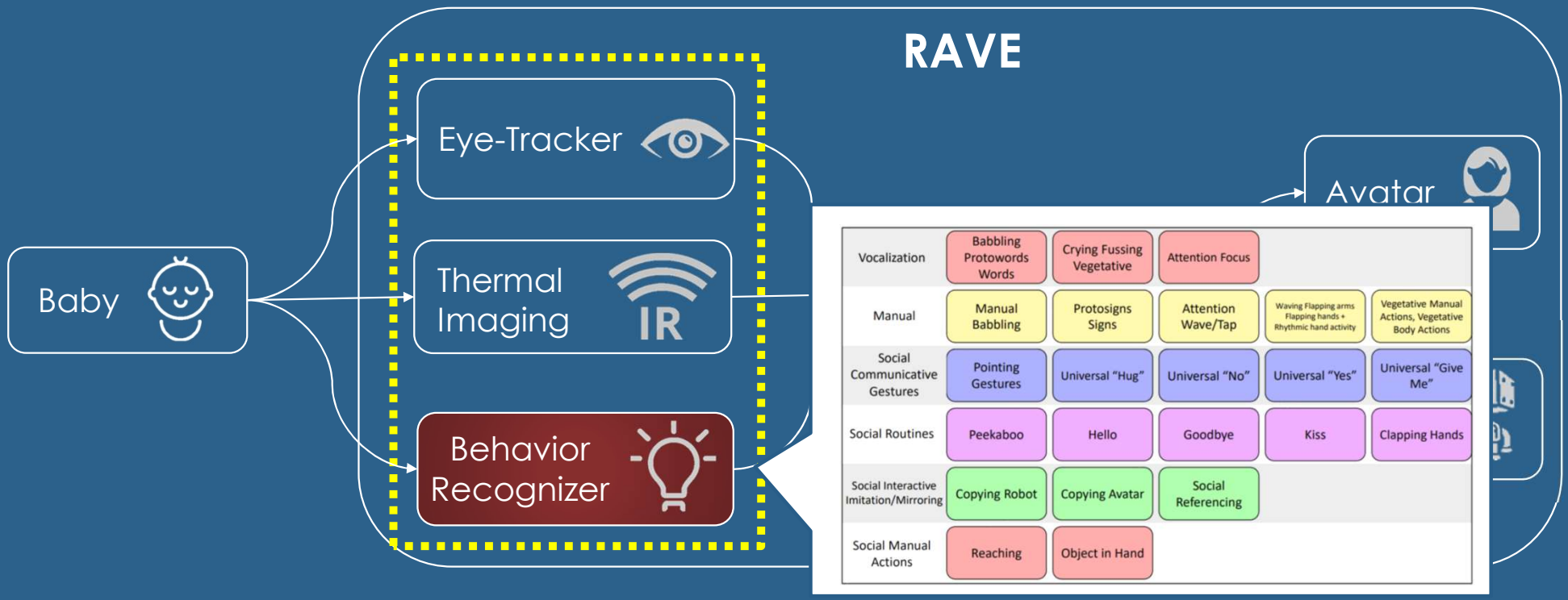
Thermal Imaging

- Digital IR thermal camera FLIR A655sc
- Using nose-tip temperature for classification*

  (Sympathetic vs Parasympathetic)
- Index of when young infants are "ready to learn"

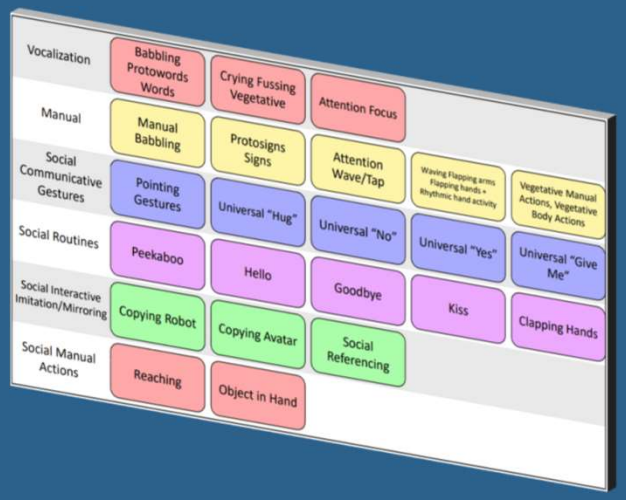
*Cardone, D., & Merla, A. (2017).

RAVE - Robot, AVatar, thermal-Enhanced System

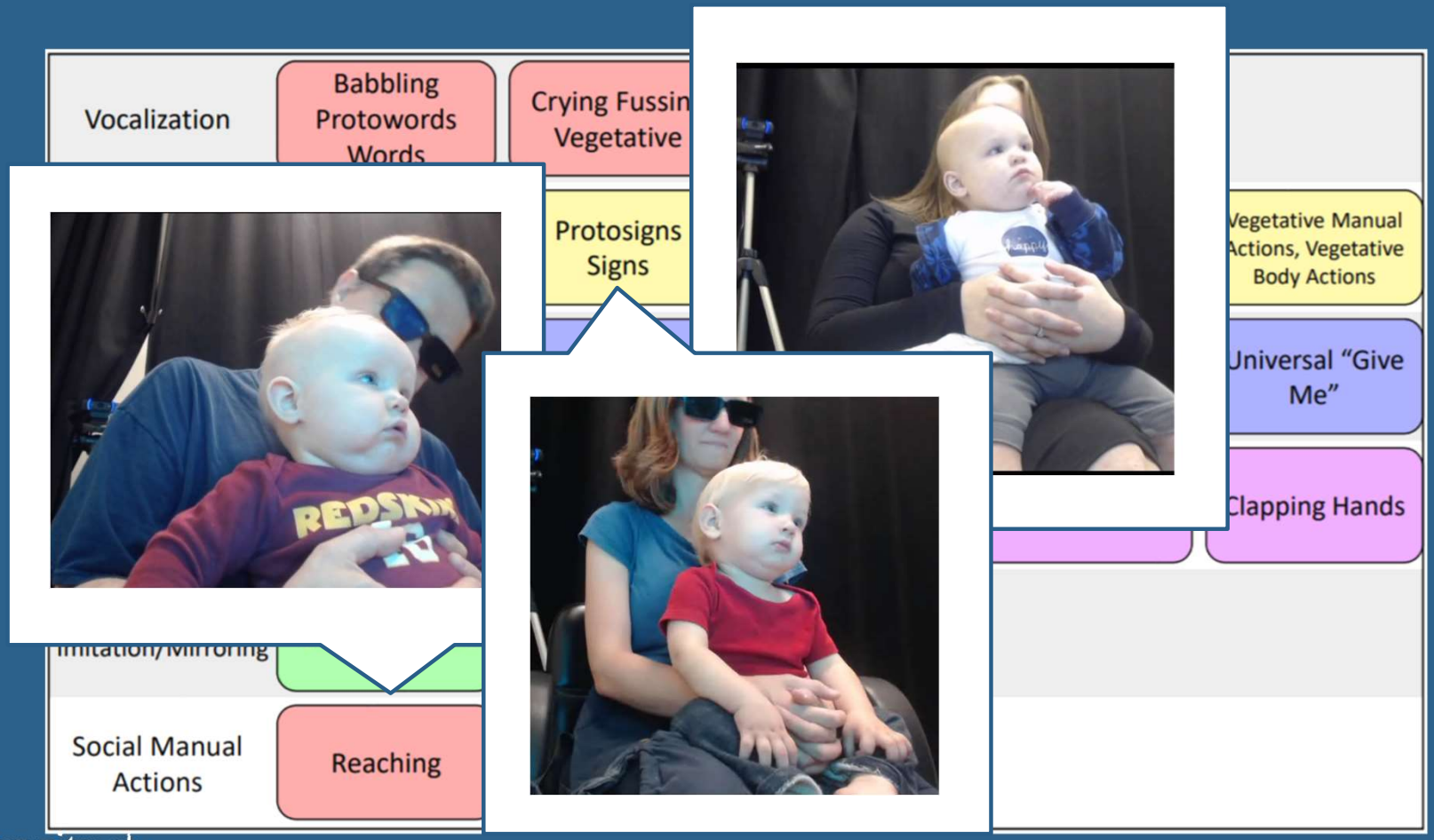


Behavior Recognizer

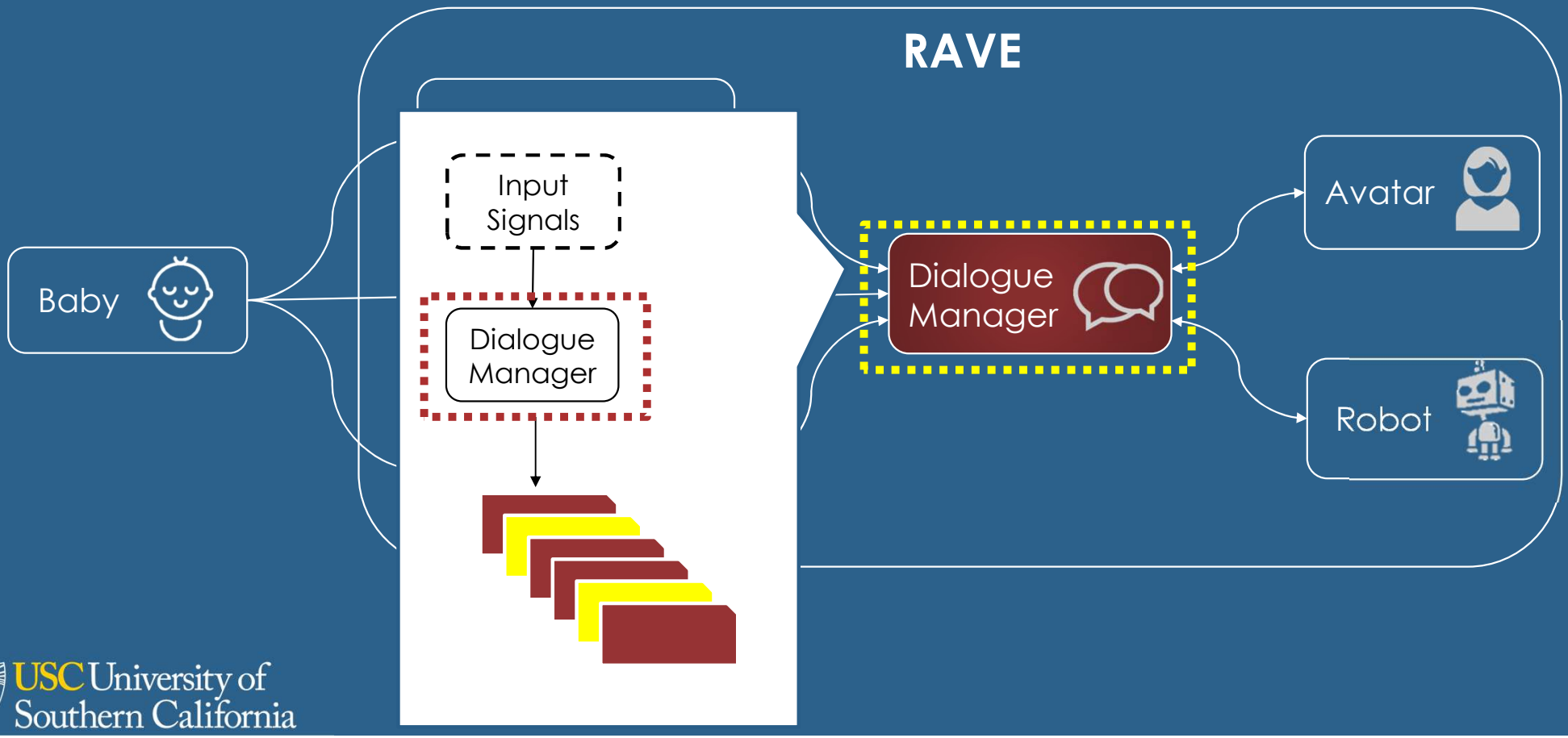
- We need to recognize communicative behaviors
- Off-the-shelf visual perception modules don't work for babies
  - Different posture and proportions
  - Babies are sitting on parents lap
- Human observer interface was used



Human Observer Interface

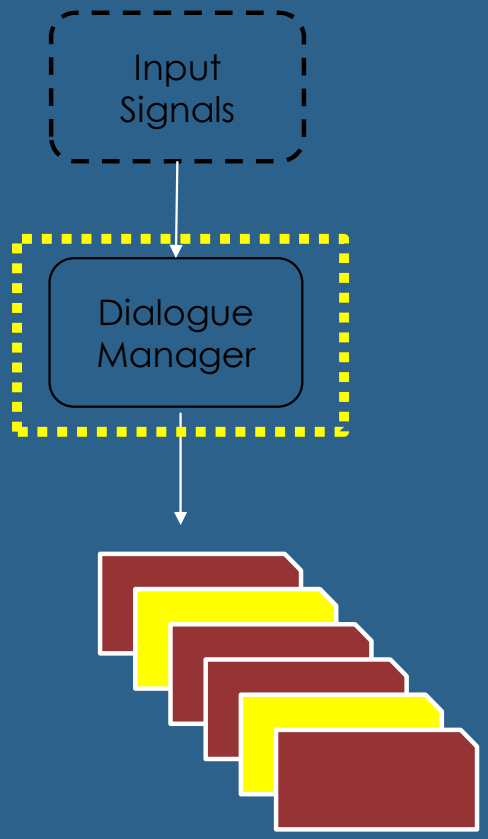


RAVE - Robot, AVatar, thermal-Enhanced System

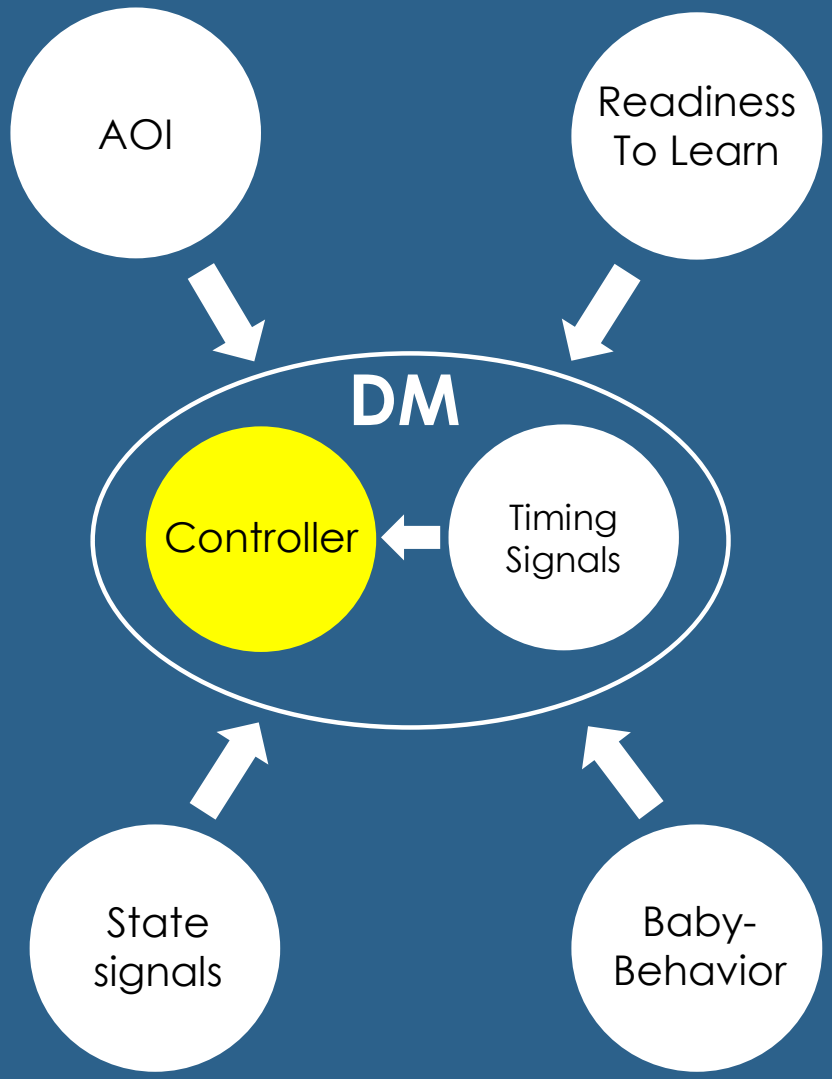


Dialogue Manager

- Rule based model
- Based on information-state architecture
- Plans ahead based on current observations
- Adjusts and re-plans as needed if the desired outcome is not observed

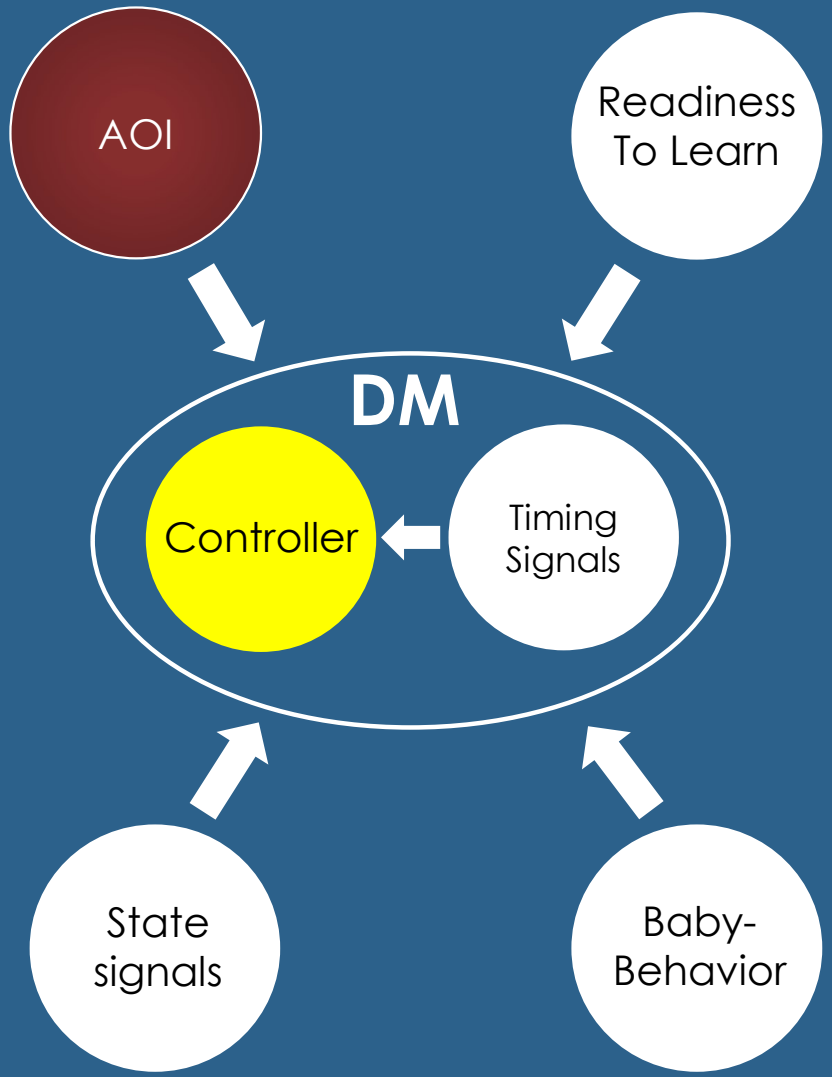


Dialogue Manager Inputs



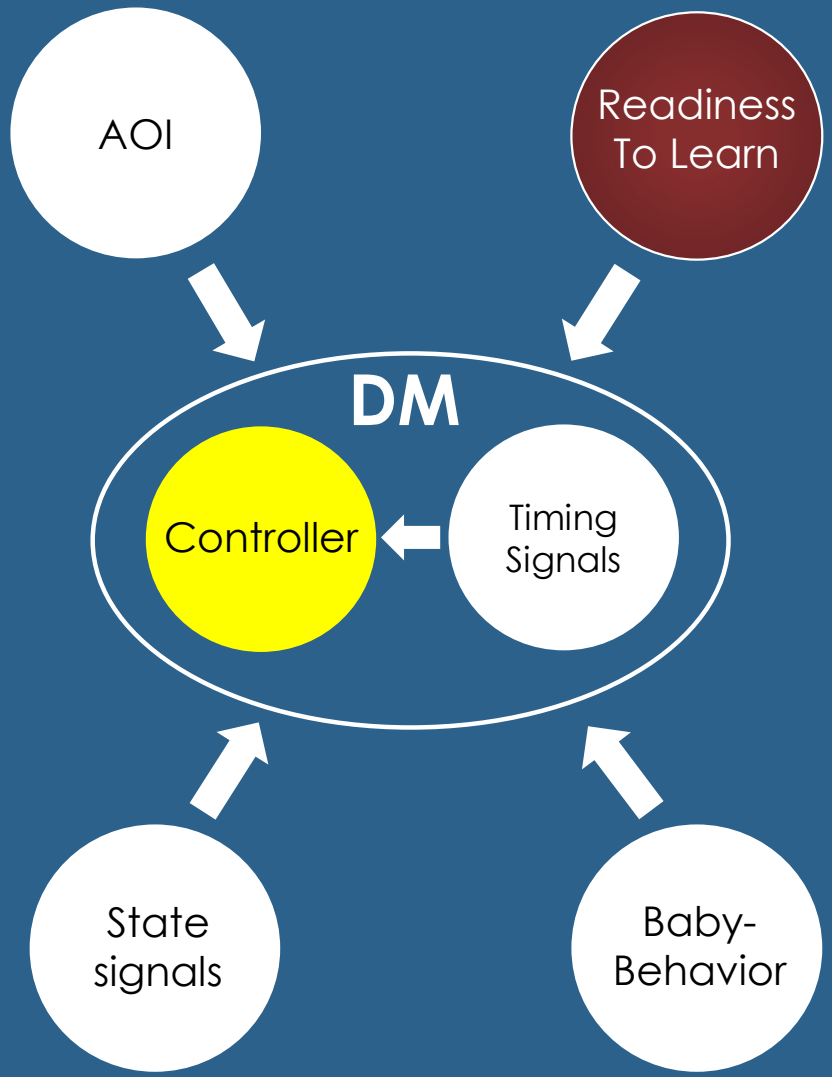
Dialogue Manager Inputs

- **Area of Interest (AOI): Avatar, Robot, Between**



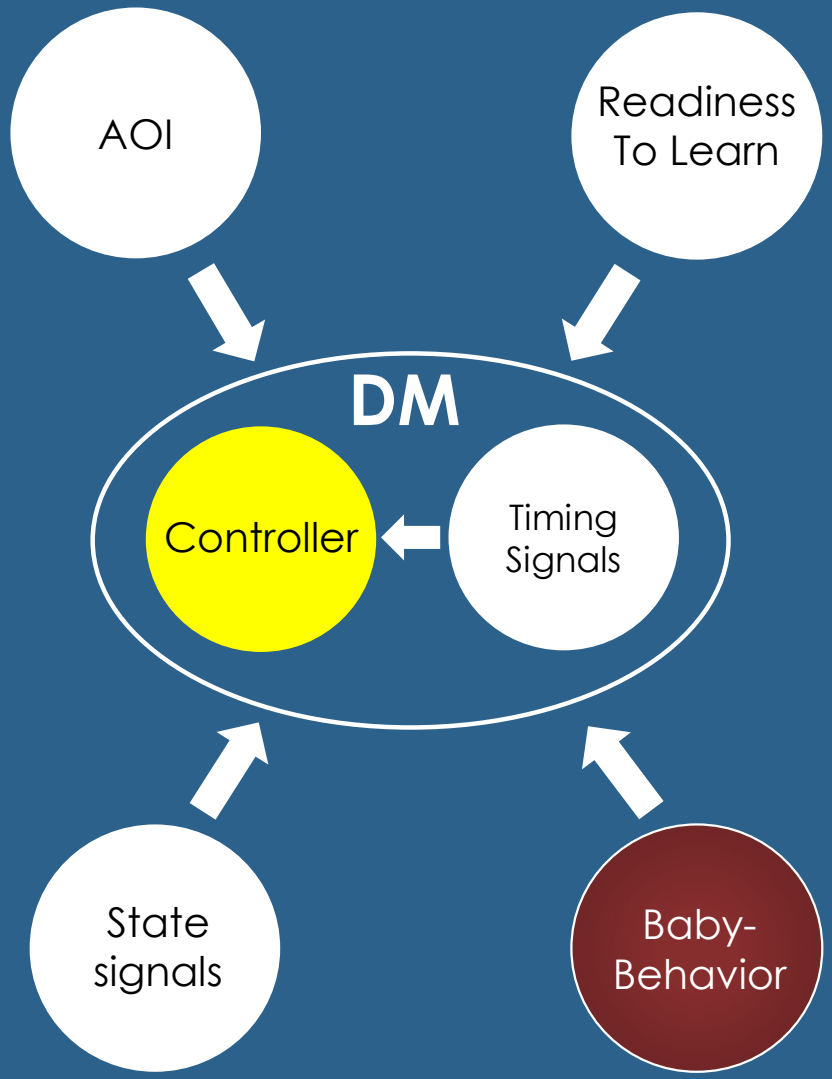
Dialogue Manager Inputs

- Area of Interest (AOI): Avatar, Robot, Between
- **Readiness To Learn: Sympathetic, Parasympathetic**



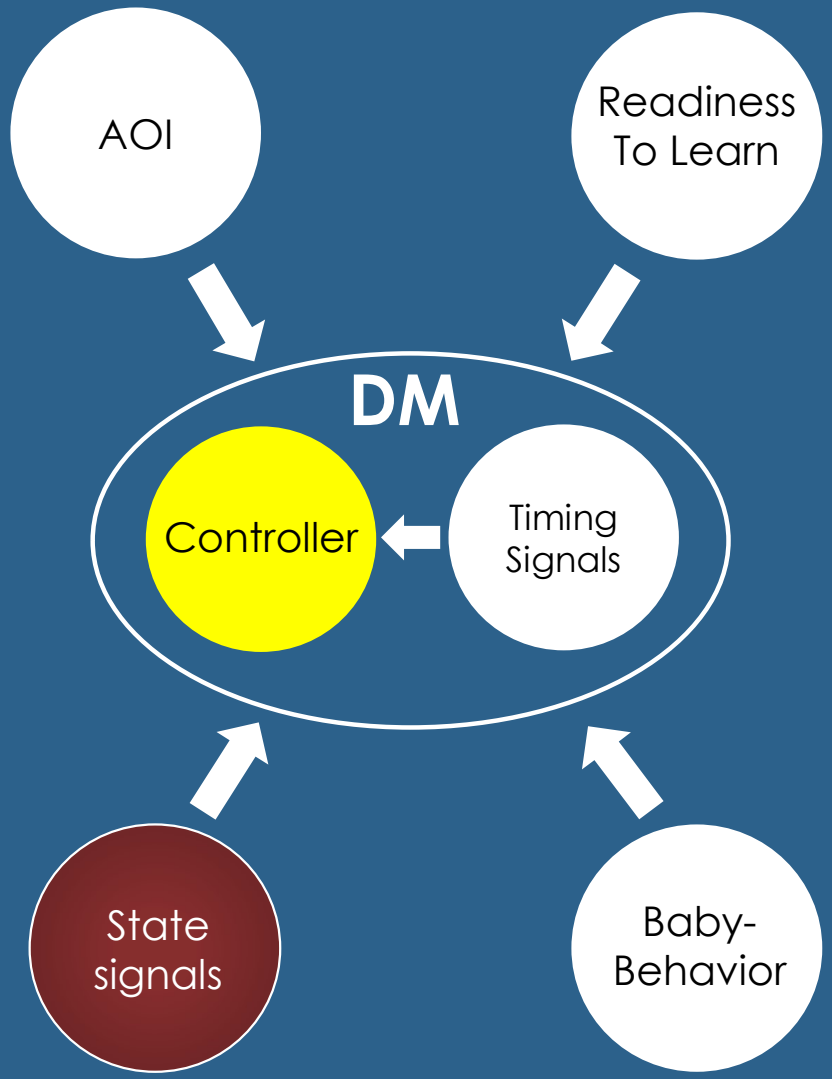
Dialogue Manager Inputs

- Area of Interest (AOI): Avatar, Robot, Between
- Readiness To Learn: Sympathetic, Parasympathetic
- **Baby-Behavior (BB): 23 Distinct States (e.g., Reaching)**



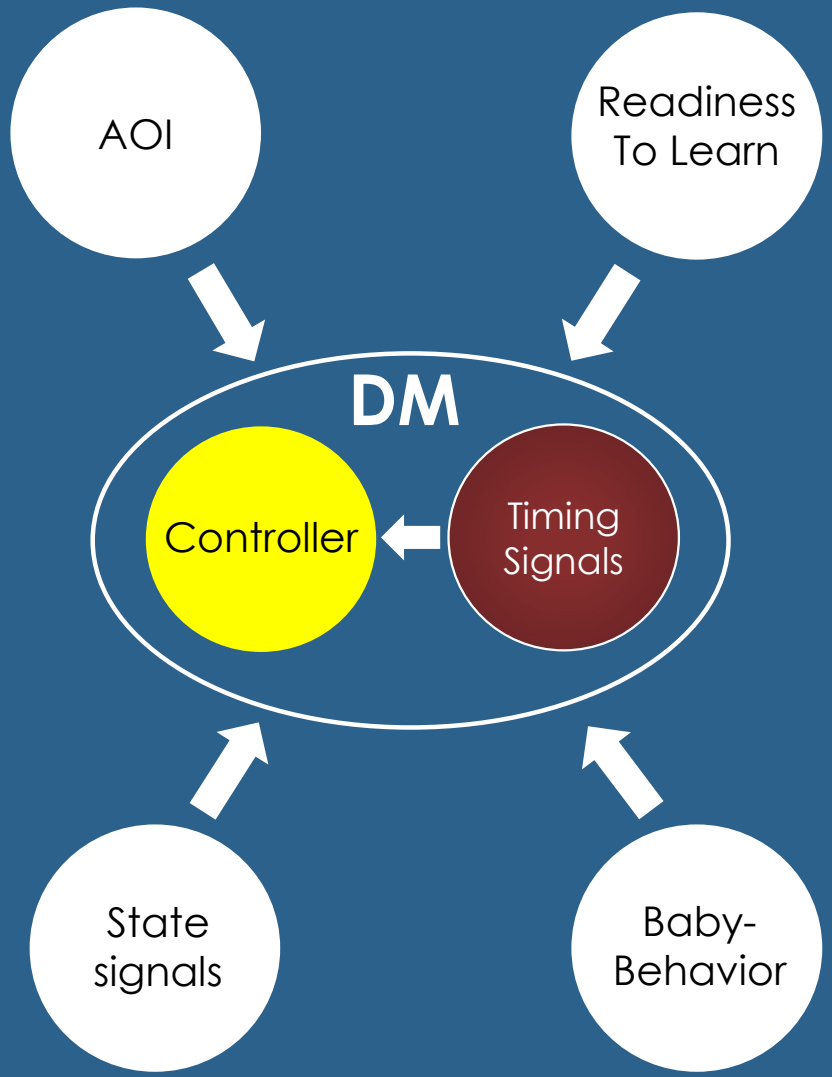
Dialogue Manager Inputs

- Area of Interest (AOI): Avatar, Robot, Between
- Readiness To Learn: Sympathetic, Parasympathetic
- Baby-Behavior (BB): 23 Distinct States (e.g., Reaching)
- **State Signals: (e.g., Robot Behavior Exception)**



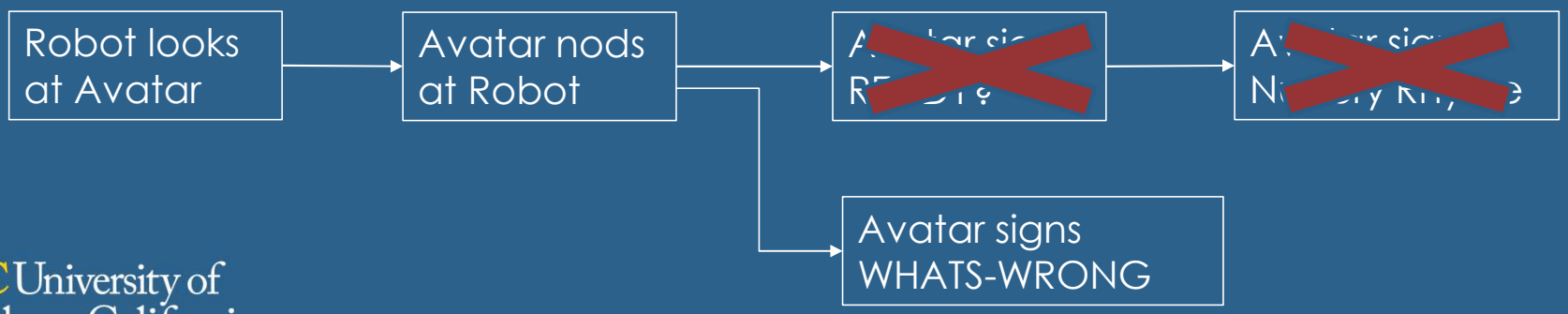
Dialogue Manager Inputs

- Area of Interest (AOI): Avatar, Robot, Between
- Readiness To Learn: Sympathetic, Parasympathetic
- Baby-Behavior (BB): 23 Distinct States (e.g., Reaching)
- State Signals: (e.g., Robot Behavior Exception)
- **Timing Signals: (e.g., Time-Out)**



Dialogue Manager Output Commands

- Primitive Behaviors
  - Atomic, cannot be interrupted
- Action Sequences
  - A sequence of primitive behaviors
  - Can be interrupted if needed at primitive behavior boundary



Avatar Primitive Behaviors

- Conversational Fillers
- Social Behaviors
- Question Solicitations
- Stories

Avatar Primitive Behaviors

- **Conversational Fillers**
- Social Behaviors
- Question Solicitations
- Stories

Avatar Primitive Behaviors

- Conversational Fillers
- **Social Behaviors**
- Question Solicitations
- Stories

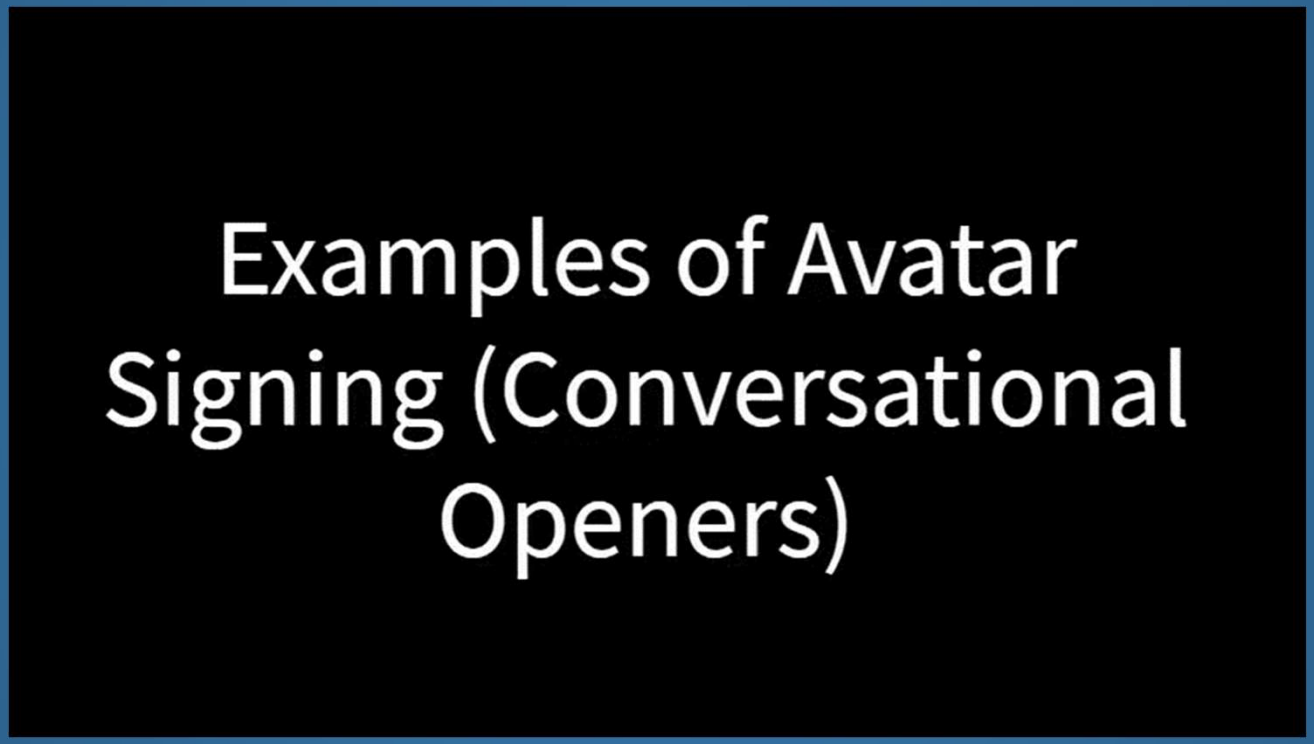
Avatar Primitive Behaviors

- Conversational Fillers
- Social Behaviors
- **Question Solicitations**
- Stories

Avatar Primitive Behaviors

- Conversational Fillers
- Social Behaviors
- Question Solicitations
- **Stories**

Examples of Avatar Signing

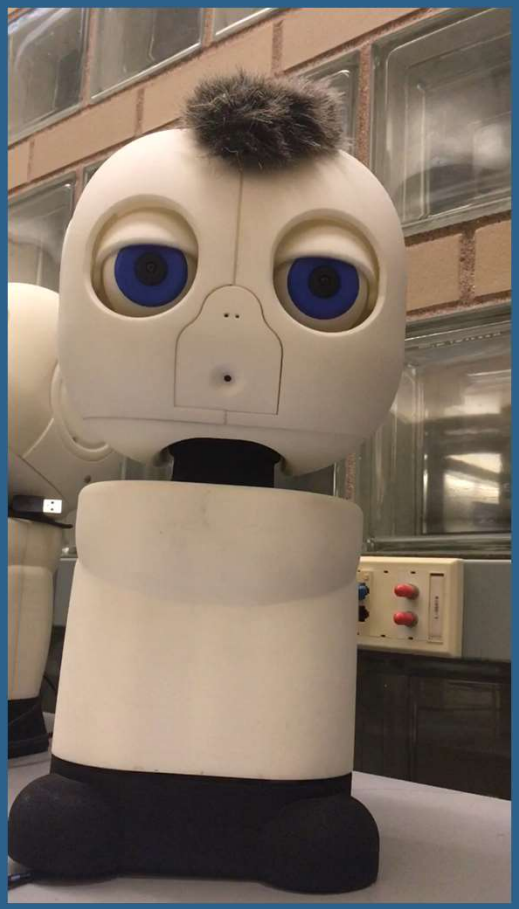


Robot Primitive Behaviors

- Head Turns
- Nodding
- Hide/Unhide

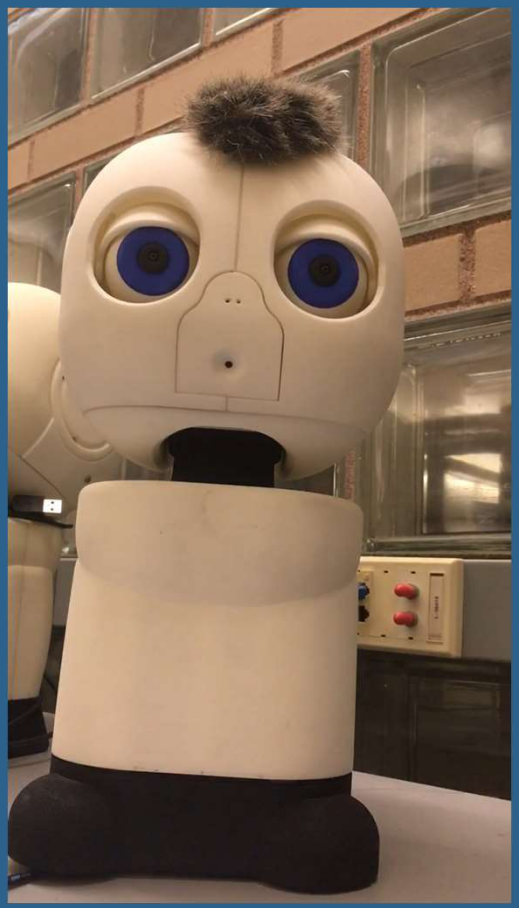
Robot Primitive Behaviors

- **Head Turns**
- Nodding
- Hide/Unhide



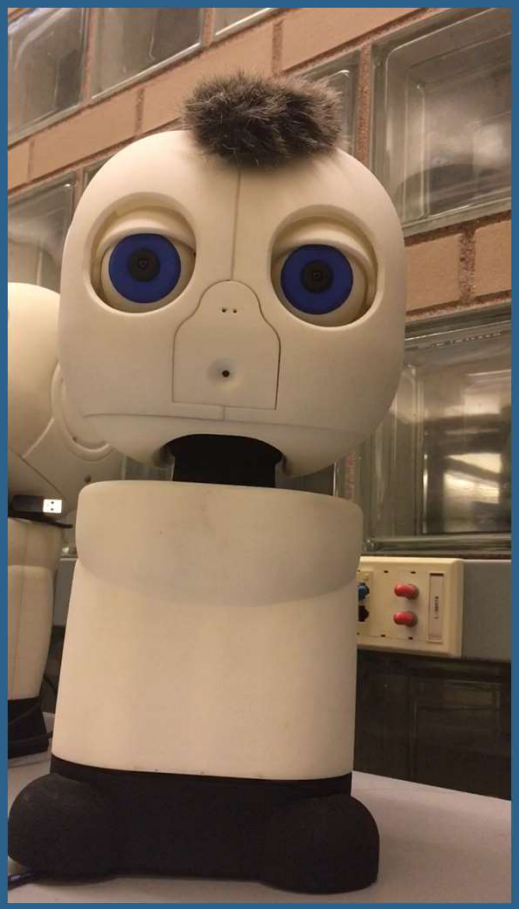
Robot Primitive Behaviors

- Head Turns
- **Nodding**
- Hide/Unhide



Robot Primitive Behaviors

- Head Turns
- Nodding
- **Hide/Unhide**



Action Sequence Example

Robot head turn

↓

Avatar signing HELLO

↓

Avatar nods

↓

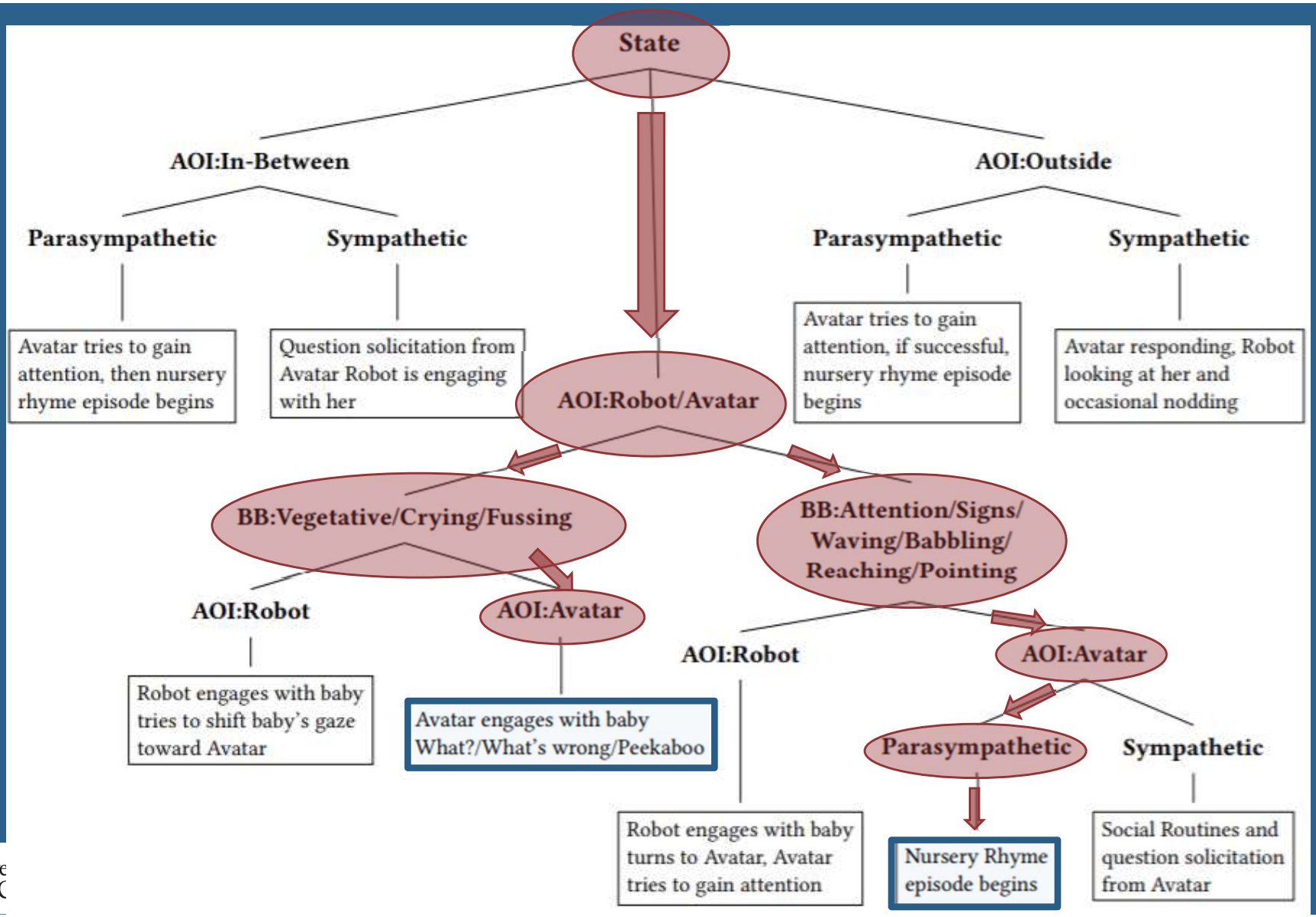
Avatar tells a story (PIG NR)

↓

Robot Nods

↓

Robot head turn


State
AOI:In-Between
AOI:Outside
Parasympathetic
Sympathetic
Parasympathetic
Sympathetic
Avatar tries to gain attention, then nursery rhyme episode begins
Question solicitation from Avatar Robot is engaging with her
AOI:Robot/Avatar
Avatar tries to gain attention, if successful, nursery rhyme episode begins
Avatar responding, Robot looking at her and occasional nodding
BB:Vegetative/Crying/Fussing
BB:Attention/Signs/ Waving/Babbling/ Reaching/Pointing
AOI:Robot
AOI:Avatar
AOI:Robot
AOI:Avatar
Robot engages with baby tries to shift baby's gaze toward Avatar
Avatar engages with baby What?/What's wrong/Peekaboo
Parasympathetic
Sympathetic
Robot engages with baby turns to Avatar, Avatar tries to gain attention
Nursery Rhyme episode begins
Social Routines and question solicitation from Avatar

Evaluation

Can Rave engage the infant's attention?

Can agents elicit socially contingent interaction?

Experiment Subjects

- 8 infant participants
  - 2 females and 6 males
  - average age of 9 months and 20 days (range 7-13 months)
  - Different levels of sign-exposure
    - 5 not sign-exposed
    - 2 hearing sign-exposed
    - 1 deaf sign-exposed

Evaluation

Can Rave engage the infant's attention?

Can agents elicit socially contingent interaction?

Evaluation

Can Rave engage the infant's attention?

Can agents elicit socially contingent interaction?

Evaluation – Engagement

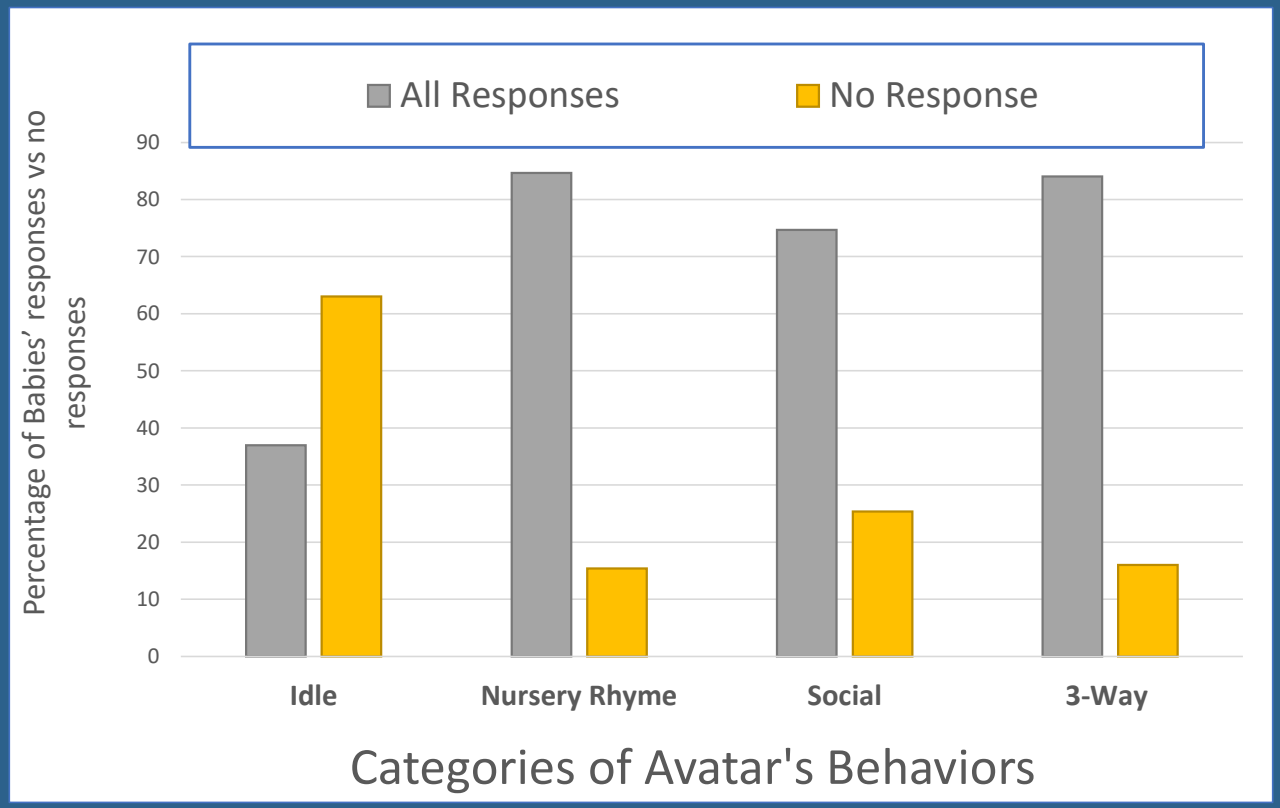
- Concern:
  - Babies would see the agents as boring and not social
  - Stranger Anxiety
- Method:
  - Overall response rate to different avatar behaviors
  - Percentage of different baby's behaviors
  - Amount of tracking the agents

Evaluation – Engagement

- All babies exhibited positive engagement behaviors
  - Immediate visual engagement (locked attention) with the agents
  - Sustained attention (persisting over time)
  - Visually tracked (gaze following) the Avatar and Robot

Response rate across Avatar's behaviors

Overall response rate : 60% (M = 61.8, SD = 6.9)



Examples of engagement

Evaluation

Can Rave engage the infant's attention?

Can agents elicit socially contingent interaction?

Evaluation

Can Rave engage the infant's attention?

Can agents elicit socially contingent interaction?

Evaluation – Contingent Interaction

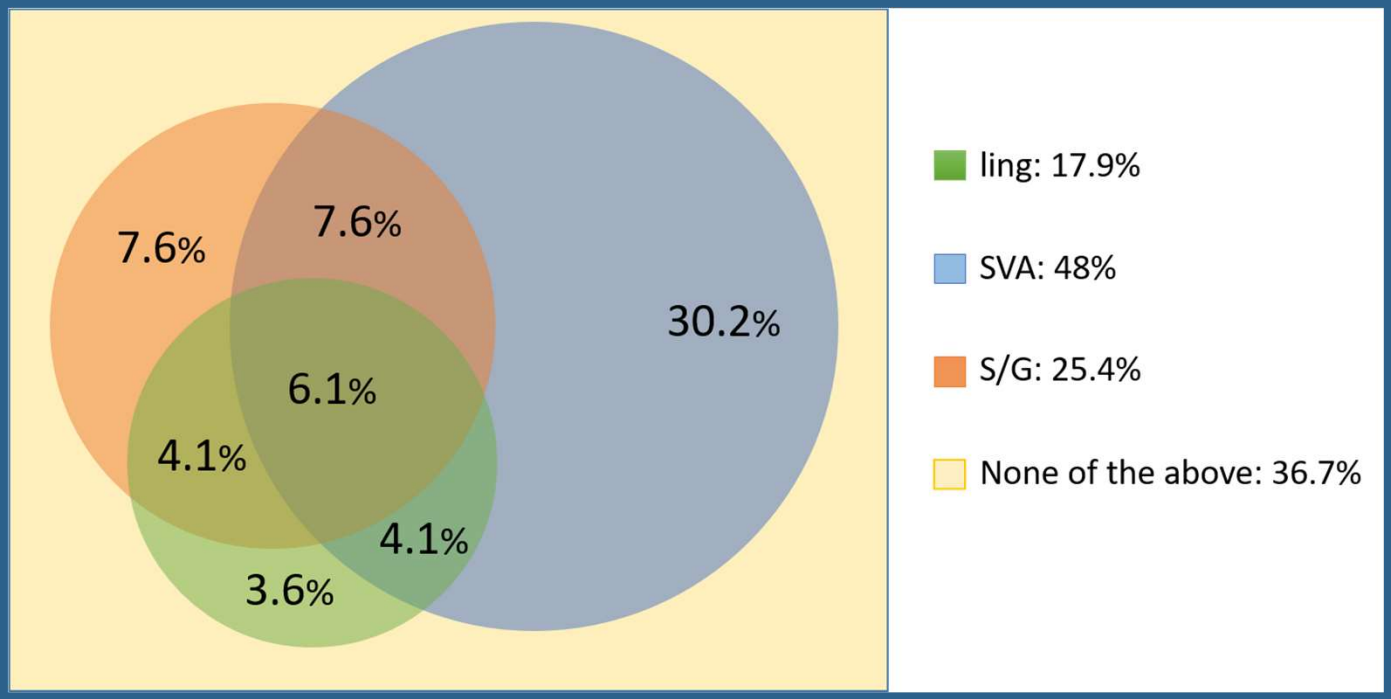
- Babies attempted to copy the avatar
- Babies performed age-appropriate proto-linguistic behavior
- Even though they were not understanding the meanings

Results - Babies' behavioral distribution

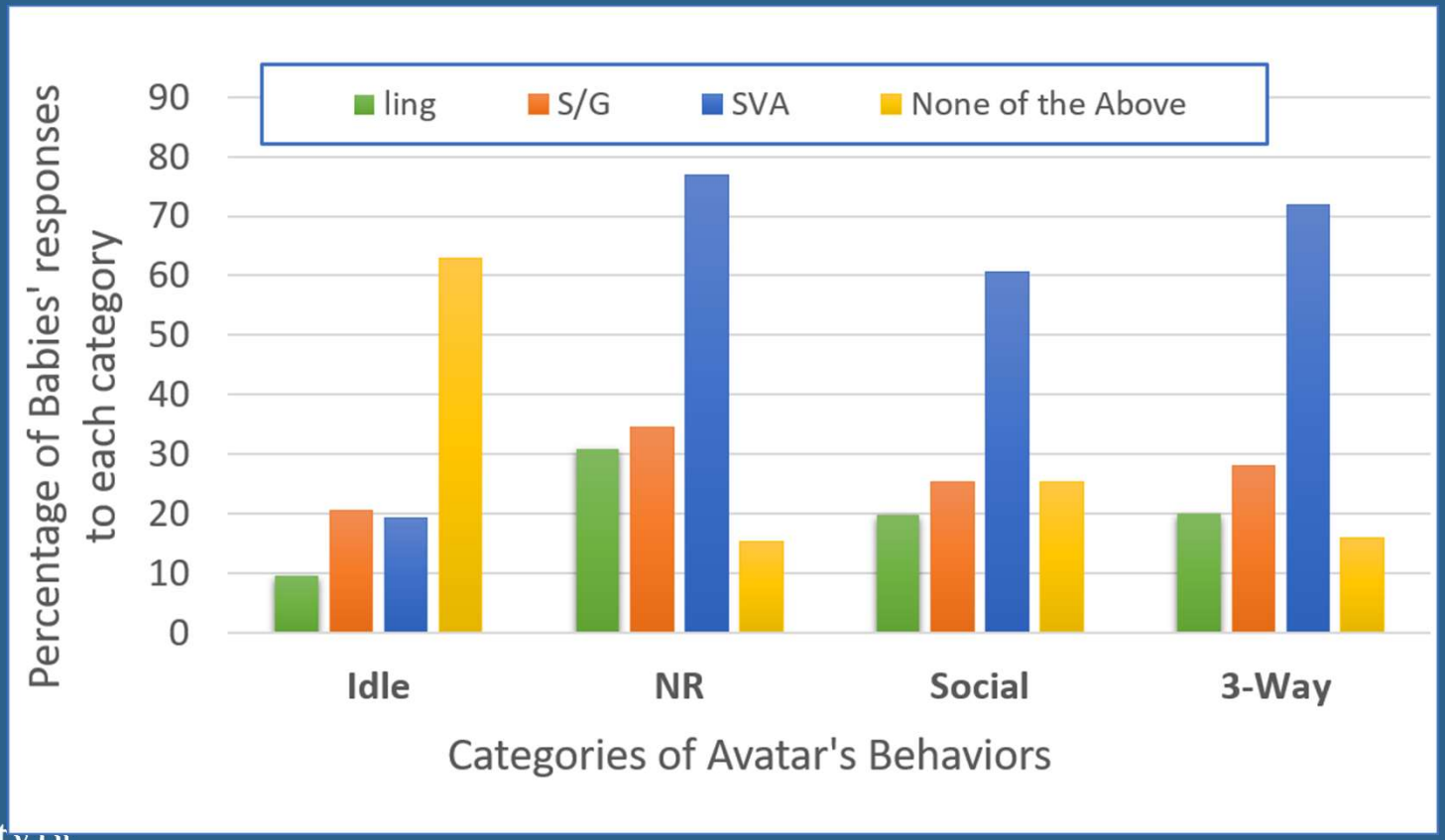
Ling (linguistic): production of manual proto-signs/sings

SVA (Sustained visual attention: baby being visually transfixed on the agents

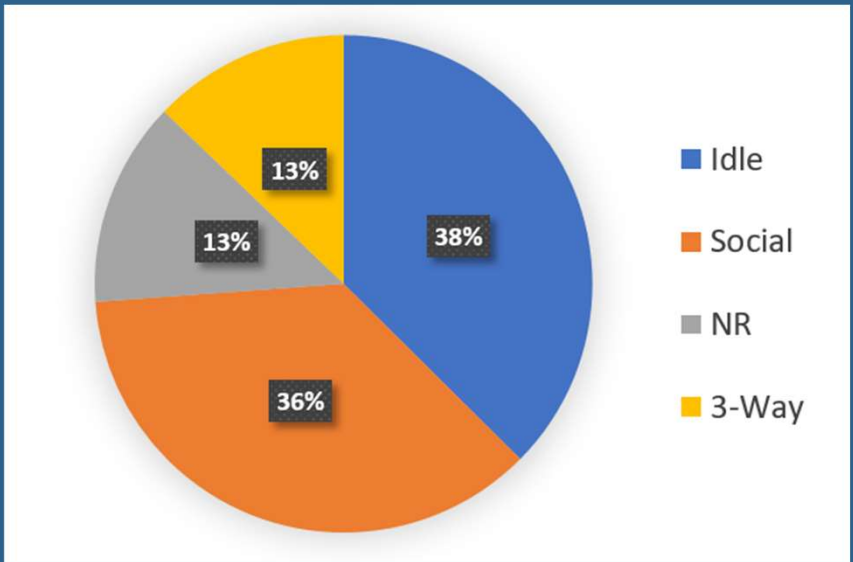
S/G (Social/Gestural): pointing, waving, social referencing



Baby Behavior in response to different Avatar's behaviors



Avatar's behavioral distribution



Example: Contingent Interaction

Example: Parents Interacting

Summary

- Agents performing dialogue routines were successful in eliciting socially contingent conversation from the infant
- Potential viability for using this kind of system for language learning in young infants

Takeaway w.r.t. Research Question

RQ 2: What kinds of multimodal feedback can agents recognize and how can they be used to adapt the dialogue management policies of the agent?

- RAVE is able to pick up multimodal input (feedback) from babies
- It can successfully use this feedback to adapt the dialogue management policy to maintain a contingent interaction

Experiment Protocol

- Introduced to Avatar and Robot
- Calibration Process
- Familiarization Episode
- Interaction session
  - Condition 1: parents were not participating
  - Condition 2: parents intervene in the interaction as they naturally do

Q&A

Mulţumesc
有り難う
Obrigado
Sağol
Thank you
謝謝
Kiitos
Grazie
شكرا لك
Hvala

Natural Language Processing and Robotics

Jesse Thomason

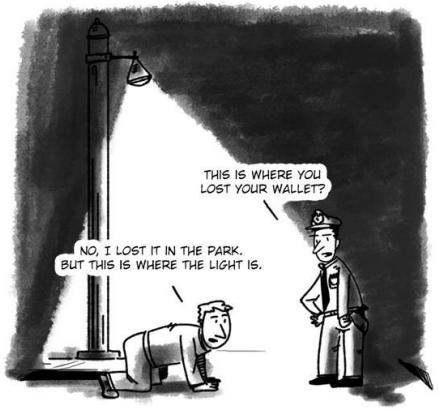
[http://glamor.rocks/]

USC Viterbi School of Engineering

✨GLAMOR

Streetlight Effect

A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, "this is where the light is".

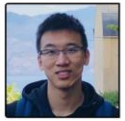


https://en.wikipedia.org/wiki/Streetlight_effect

GLAMOR Lab

- **G**rounding **L**anguage in **A**ctions, **M**ultimodal **O**bservations, and **R**obots

PhD Students



Bill Zhu
2nd year PhD student (co-Robin Jia)
Compositional Semantics
Vision-Language Navigation

Tejas Srinivasan
2nd year PhD student
Vision-Language Alignment
VL Continual Learning

Anthony Liang
2nd year PhD student
Language-guided RL
RL for Dialogue

Ting-Yun Chang
2nd year PhD student (co-Robin Jia)
Pre-trained Model Understanding

Ishika Singh
2nd year PhD student
VLN and VL Interaction

Leticia Pinto-Alva
2nd year PhD student
Vision-and-Language

Lee Kezar
3rd year PhD student
VL for ASL

Abrar Anwar
2nd year PhD student
Speech and Gesture
Assistive Robotics

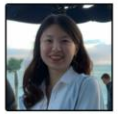
Undergraduate and Masters Students

Elle Szabo
Senior Undergraduate @ USC
Computer Science

Flora Jia
Junior Undergraduate @ USC
Computer Science and Applied Mathematics
CURVE Fellowship

Yuan Huang
2nd year Masters Student @ USC
Electrical and Computer Engineering
Specializing in Machine Learning and Data Science

Julie Kim
Sophomore Undergraduate @ USC
Computer Engineering and Computer Science
CURVE Fellowship (co-Maja Mataric)



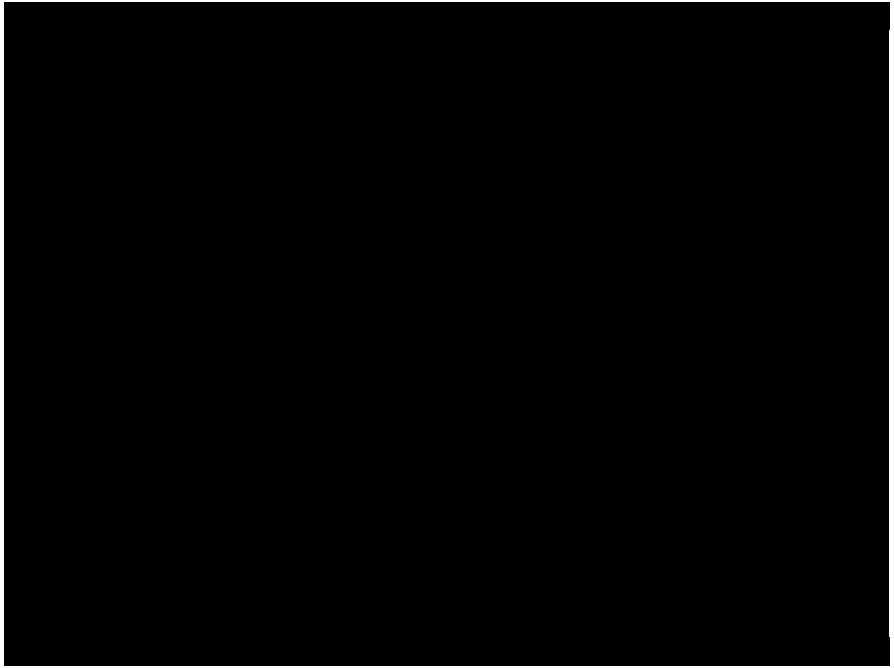
Chu Fang
Senior Undergraduate @ USC
Computer Science (Games)
URAP

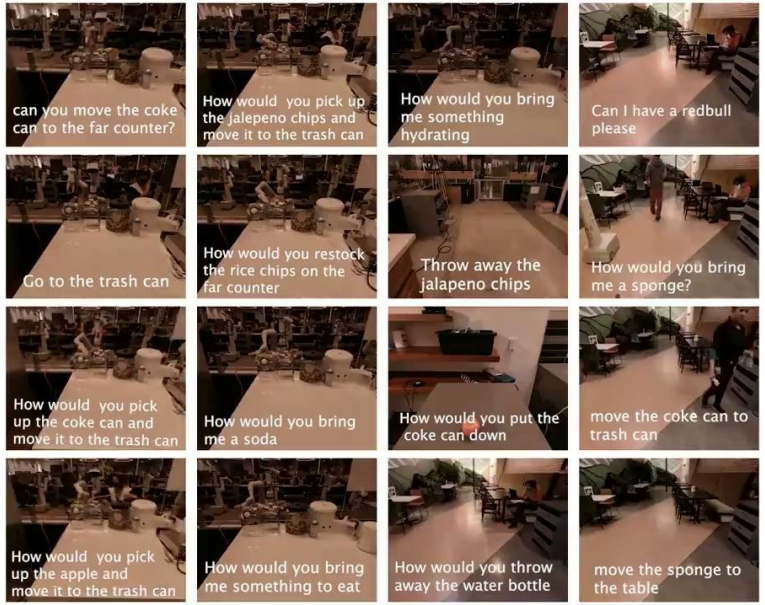
Junu Song
Junior Undergraduate @ USC
Computer Science
CURVE Fellowship

Leslie Moreno
Sophomore Undergraduate @ USC
Computer Engineering and Computer Science
CURVE Fellowship (co-Maja Mataric)

http://glamor.rocks/

Why Would a Robot Need Language?



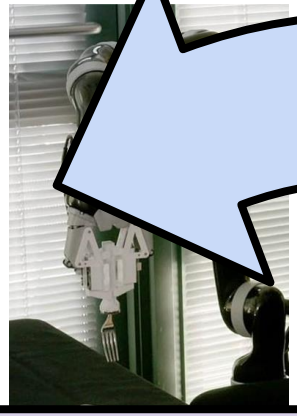


https://emprise.cs.cornell.edu/hrcom/ ; https://say-can.github.io/

Why Aren't Our Robots Language-Guided Already?

Require visual *grounding*.

Boston Dynamics



???

No visual connections.

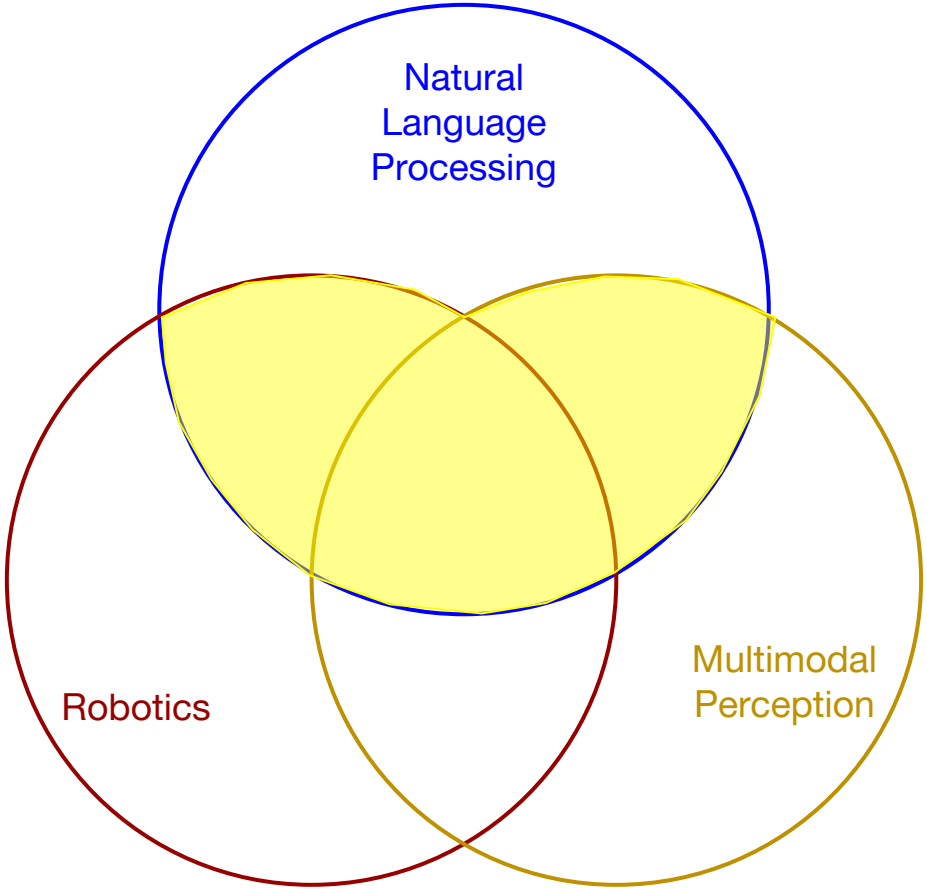
Amazon

Google

iRobot

Approach Summary:
Enable robots to perform language grounding via interaction with humans.

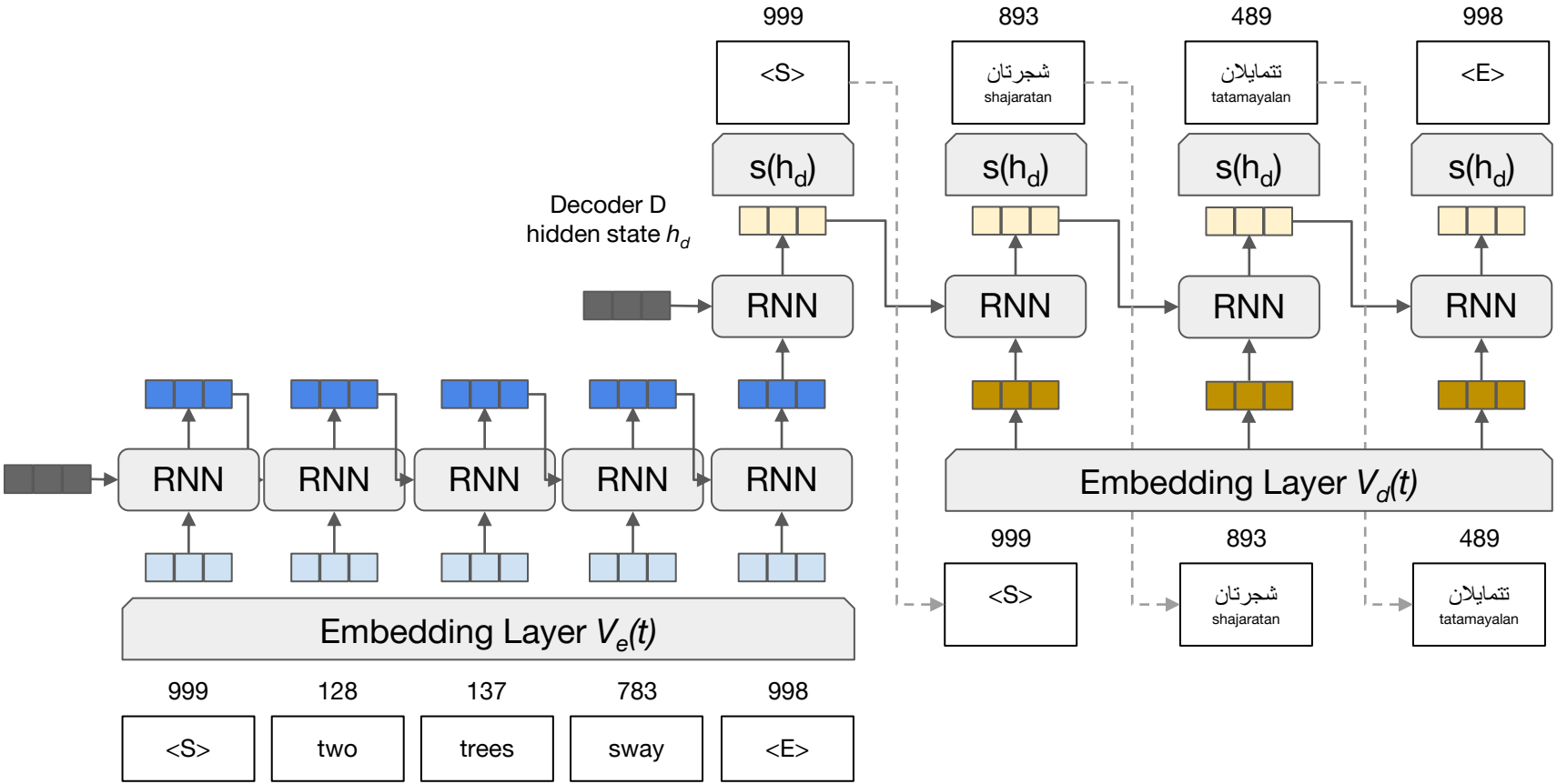
GLAMOR Lab Work



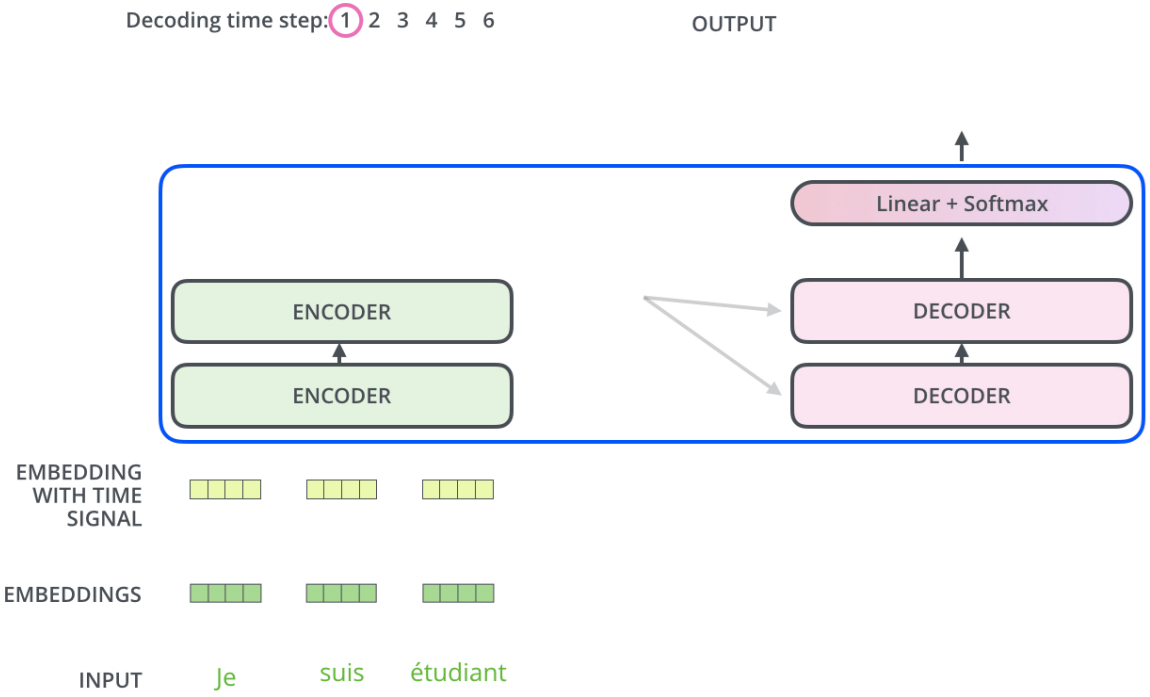
Natural Language Processing

- Concerned with turning “in the wild” human language into something machine readable for downstream tasks like
  - Text classification
    - Sentiment analysis, fake news detection, hate speech detection, spam filters, …
  - Textual language modeling
    - Text generation (e.g., GPT-3), autocomplete on your phone, spelling correction suggestions, machine translation, …

Machine Translation: Basic Idea

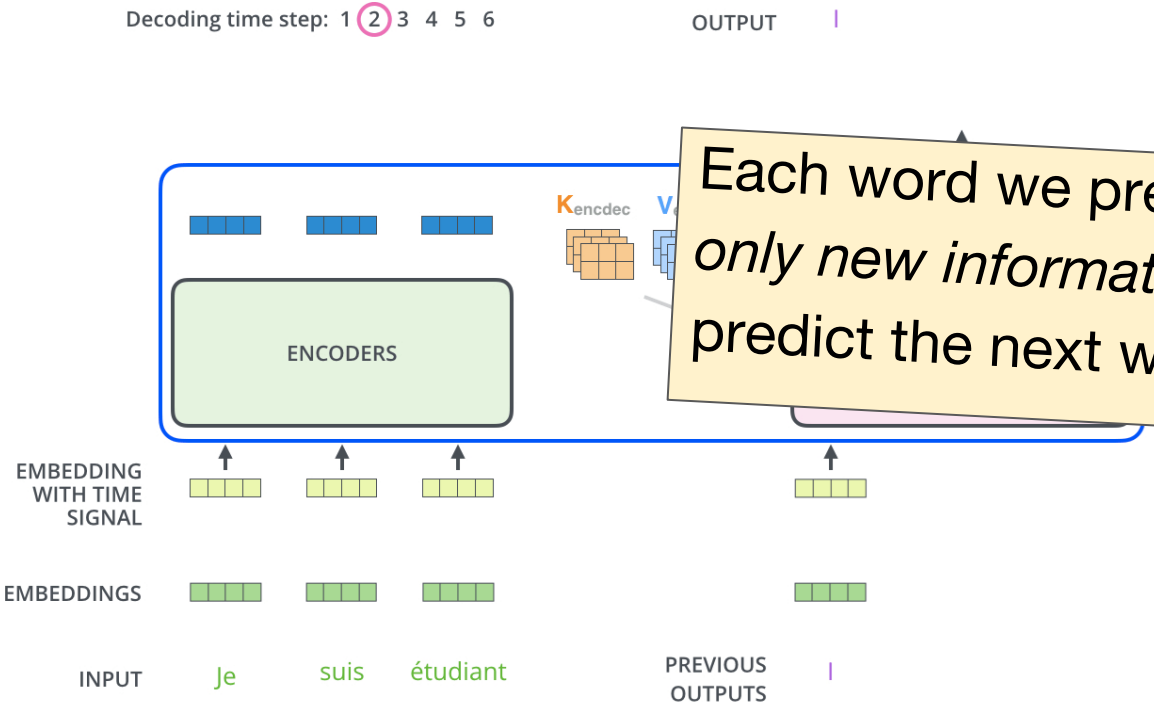


Machine Translation: Transformer Models



https://jalammar.github.io/illustrated-transformer/

Machine Translation: Transformer Models

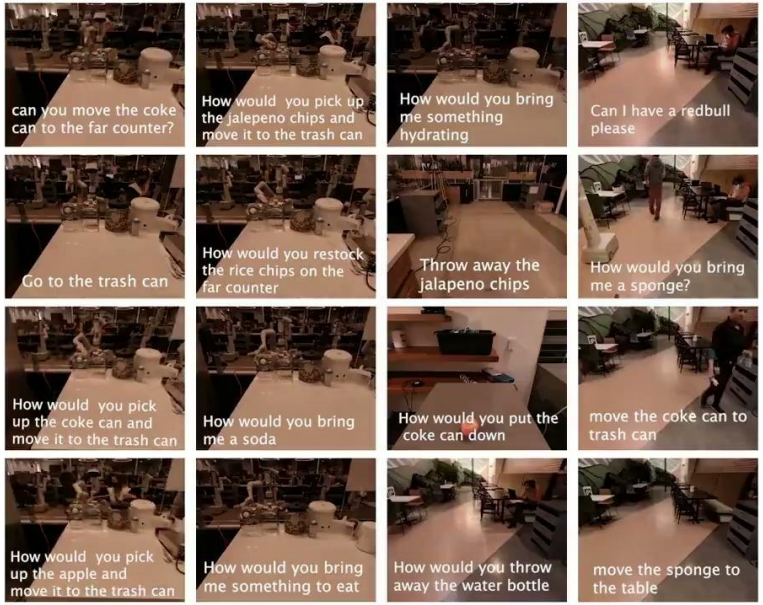


Each word we predict is the *only new information we get* to predict the next word.

https://jalammar.github.io/illustrated-transformer/

Machine Translation for Robots

- Source language input:
  - "Can you move the coke can to the far counter?"
- Target language output:
  - `look down, rotate right, move forward, move forward, rotate right, grasp coke can, rotate right, move forward, move forward, rotate right, place coke can`



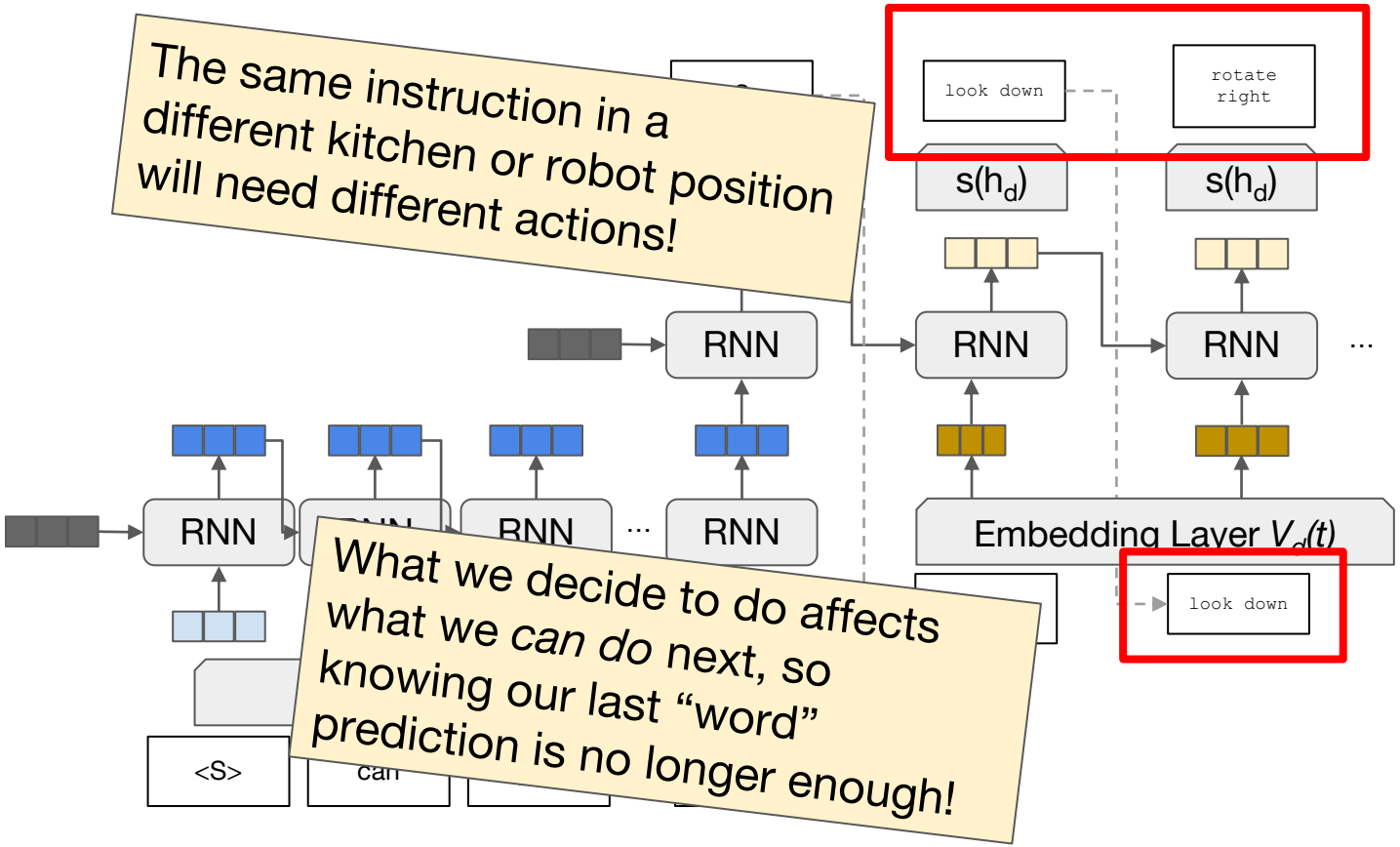
https://say-can.github.io/

Machine Translation for Robots

- Source language input:
  - "Can you move the coke can to the far counter?"
- Target language output:
  - `look down, rotate right, move forward, move forward, rotate right, grasp coke can, rotate right, move forward, move forward, rotate right, place coke can`

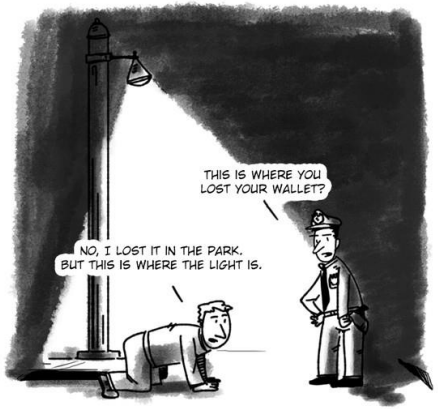
- How many input/output examples would you need to learn a *policy* that reliably translates English to sequences of robot actions?

Machine Translation for Robots



Streetlight Effect

A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, "this is where the light is".



https://en.wikipedia.org/wiki/Streetlight_effect

[Bisk, Holtzman, **Thomason**, et al., EMNLP'20]

Experience Grounds Language - World Scopes

Upshot:
Most current
operates at th

Upshot:
Most L folks playing
and vice versa are

Upshot:
Maybe this is AI Complete;
but so is "language"

WS1
Carefully Annotated
Penn Treebank
Brown Corpus
WordNet

WS2
Unstructured, Unlimited Text
Common Crawl
Word2Vec
ELMo, BERT, GPT*

WS3
Text Paired with Sensory Data
ImageNet, VQA,
ViLBERT,
Video Captioning

WS4
Language and an Embodied Agent
VLN*, IQA,
NL+Games,
RoboNLP

WS5
Social Embodiment
Human-robot
collaboration
and dialog

Claim:
You can't learn
from the radio

Claim:
You can't learn lan
from a television.

Claim:
You can't learn language
by yourself.

[Bisk, Holtzman, **Thomason**, et al., EMNLP'20]

Experience Grounds Language - World Scopes

Common Crawl

WS1

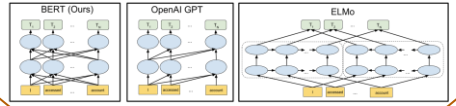
Carefully Annotated

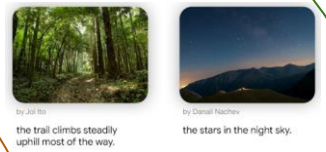
Penn Treebank
Brown Corpus
WordNet

WS2

Unstructured Unlimited Text

Common Crawl
Word2Vec
ELMo, BERT, GPT*

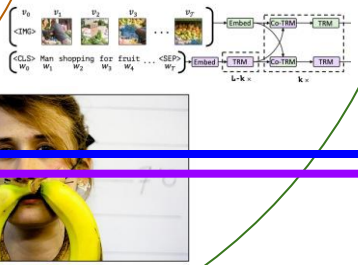




WS3

Text Paired with Sensory Data

ImageNet, VQA,
ViLBERT,
Video Captioning



WS4

Language and an Embodied Agent

VLN*, IQA,
NL+Games,
RoboNLP

WS5

Social Embodiment

Human-robot collaboration and dialog

Images and Text for Retrieval

Text Query

"A tropical bird perches in the jungle."

Candidate Images

Images and Text for Retrieval

Image Query

Candidate Captions

"A rabbit sits in the palm of a hand."

"Men are talking on a basketball court."

"A tropical bird perches in the jungle."

"Children play soccer in a field."

"A white fox is looking at the camera."

"Sports equipment is staged for a photo."

Formulating the Retrieval Problem

Images	$\mathcal{I}$
Captions	$\mathcal{L}$
Scoring Function	$F : \mathcal{I} \times \mathcal{L} \to \mathbb{R}$

Formulating the Retrieval Problem

$$F : \mathcal{I} \times \mathcal{L} \rightarrow \mathbb{R}$$

F(# red pixels, # word “red”) = 1 if (a > 0, b > 0) else 0

F(RGB bins, color word counts) =
sum of #(color word, bin)/(#color word + #bin) in training data

F(detected objects, word counts) =
sum of #(object, word)/(#object + #word) in training data

F(image pixels, token sequence) =
NN trained with contrastive matching loss

Formulating the Retrieval Problem

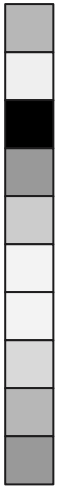
Image Features	$\psi : \mathcal{I} \rightarrow \mathbb{R}^n$
Caption Features	$\omega : \mathcal{L} \rightarrow \mathbb{R}^m$
Scoring Function	$F_{(\psi,\omega)} : \psi(\mathcal{I}) \times \omega(\mathcal{L}) \rightarrow \mathbb{R}$

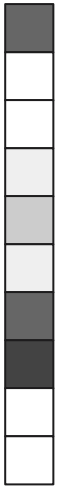
Feature Extraction for Language

BOW	"A rabbit sits in the palm of a hand."	"A tropical bird perches in the jungle."	"Men are talking on a basketball court."
a	2	1	1
bird	0	1	0
hand	1	0	0
in	1	1	0
men	0	0	1
on	0	0	1
rabbit	1	0	0
the	0	1	0
⋮	⋮	⋮	⋮

Feature Extraction for Vision

RGB
kNN bins



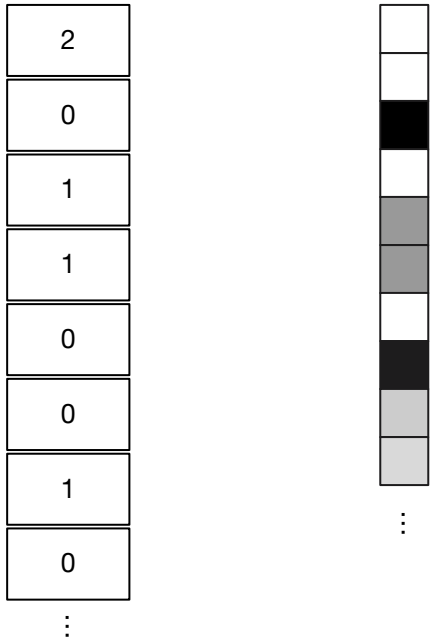


A Simple Retrieval Solution

$$\psi : \mathcal{I} \rightarrow \mathbb{R}^n$$

$$\omega : \mathcal{L} \rightarrow \mathbb{R}^m$$

$$F_{(\psi,\omega)} : \psi(\mathcal{I}) \times \omega(\mathcal{L}) \rightarrow \mathbb{R}$$



Contextual Information

You shall know a word by the company it keeps (Firth, J. R. 1957:11)

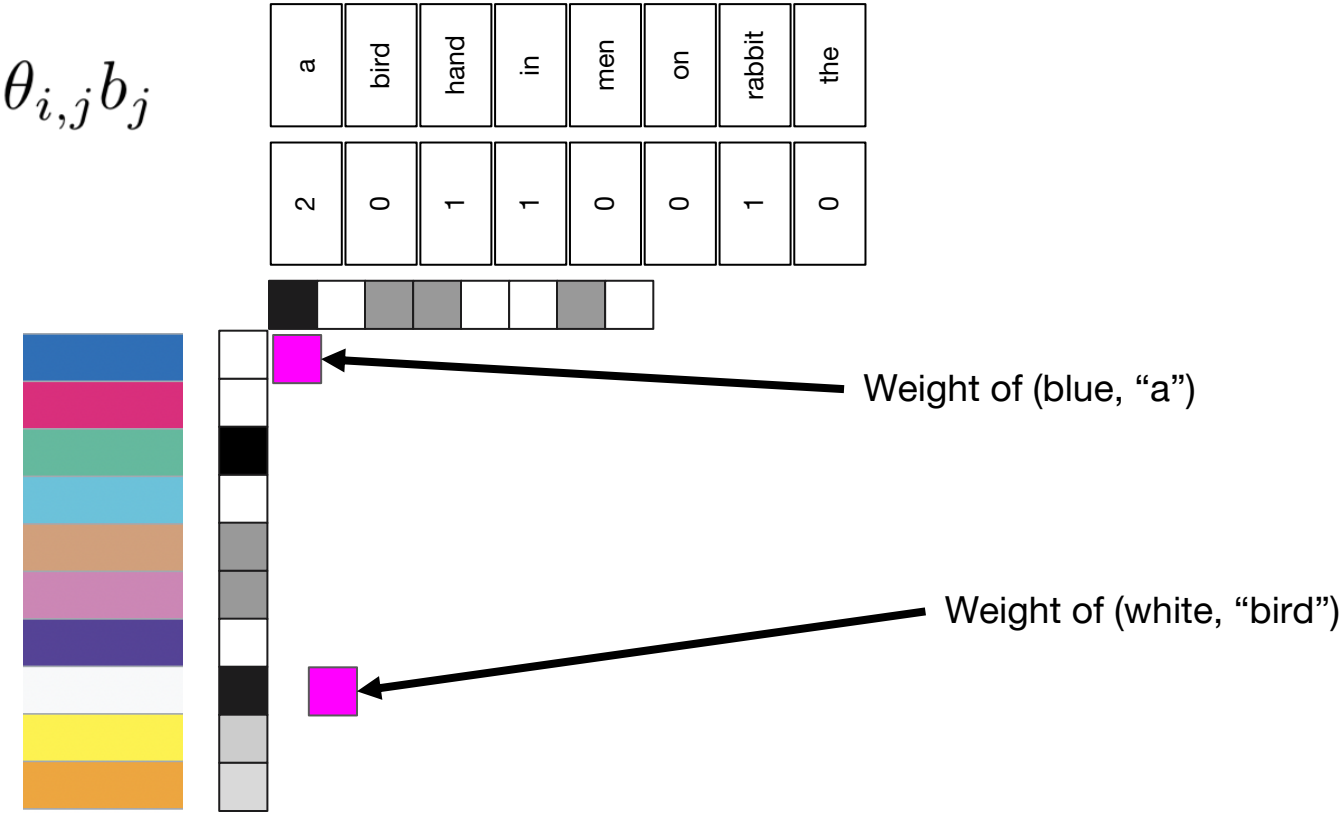
You shall know the features of modality A by the company they keep in the features of modality B. And vice versa.

Formulating the Retrieval Problem as a Linear Model

$$F_{(\psi,\omega)} : \psi(\mathcal{I}) \times \omega(\mathcal{L}) \rightarrow \mathbb{R}$$

Formulating the Retrieval Problem as a Linear Model

$$\sum_{i=1,j=1}^{n,m} a_i \theta_{i,j} b_j$$



Pointwise Mutual Information

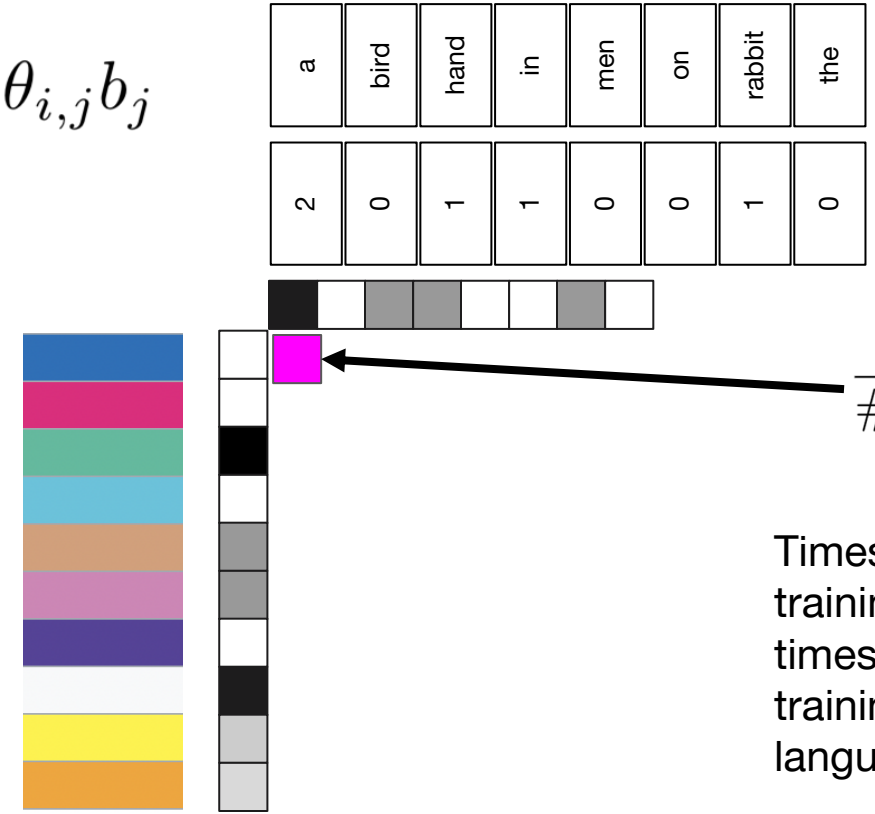
$$\mathrm{pmi}(x; y) = \log \frac{p(x, y)}{p(x)p(y)}$$

$$F(A, B) = \mathrm{pmi}(A; B); A \in \mathcal{I}, B \in \mathcal{L}$$

$$
\begin{aligned}
F(\phi(A), \omega(B)) &= \mathrm{pmi}(\vec{a}; \vec{b}) \\
&= \log \frac{p(\vec{a}, \vec{b})}{p(\vec{a})p(\vec{b})}
\end{aligned}
$$

Formulating the Retrieval Problem as a Linear Model

$$\sum_{i=1,j=1}^{n,m} a_i \theta_{i,j} b_j$$

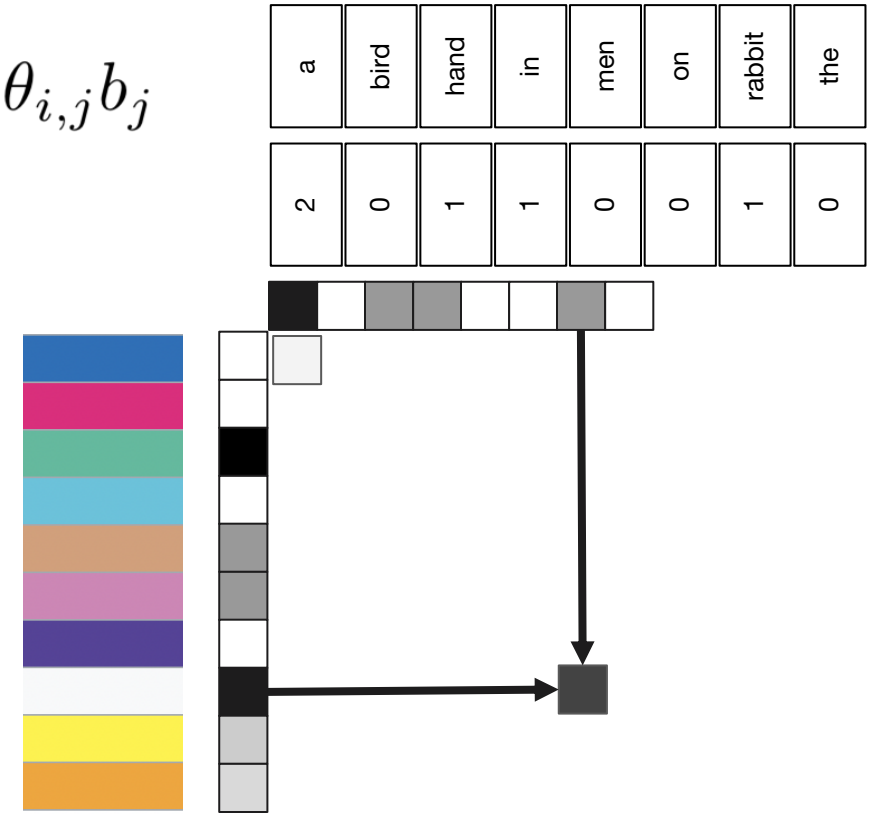


$$\frac{\#(a_i, b_i; \mathcal{D})}{\#(a_i; \mathcal{D}) + \#(b_j; \mathcal{D})}$$

Times (blue, “a”) co-occurred in training data divided by sum of times blue occurred in visual training data and “a” occurred in language training data.

Formulating the Retrieval Problem as a Linear Model

$$\sum_{i=1,j=1}^{n,m} a_i \theta_{i,j} b_j$$

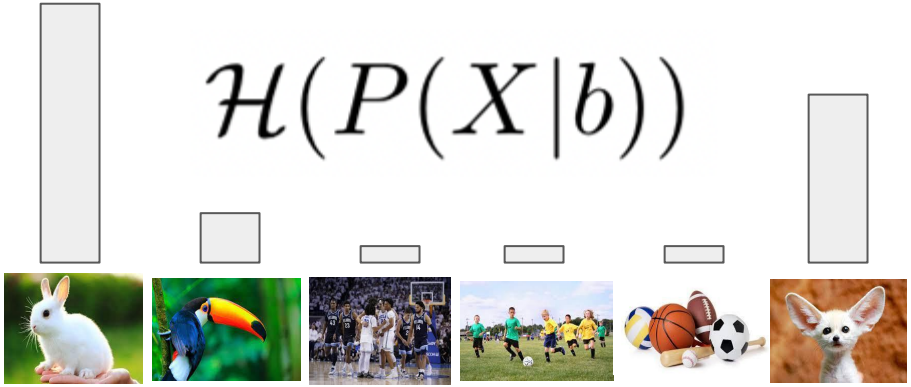


Feature Extraction and Representation Learning

$$X \subset \mathcal{I}, Y \subset \mathcal{L}$$

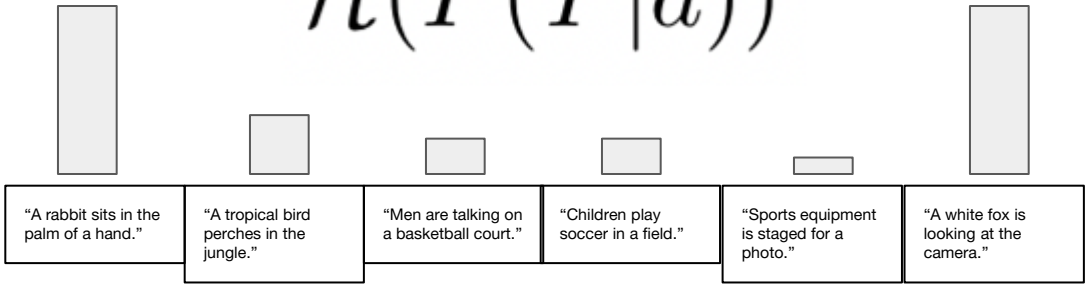
$$a \in \psi(\mathcal{I}), b \in \omega(\mathcal{L})$$

$$\mathcal{H}(P(X|b))$$



b indicates “rabbit”

$$\mathcal{H}(P(Y|a))$$



a indicates detected furry ear

Tokenization Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”

“Perching parakeet in a wire frame cage.”

“A snow owl lands on a wooden perch.”

$$\mathcal{H}(P(X|b))$$

Tokenization Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”

“Perching parakeet in a wire frame cage.”

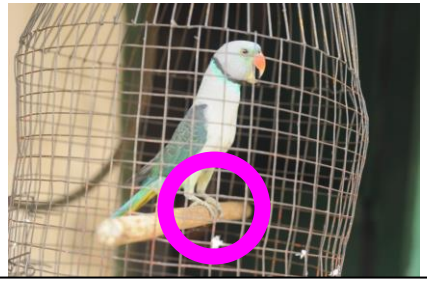
“A snow owl lands on a wooden perch.”

Stemming

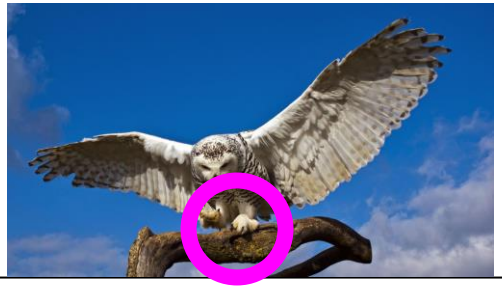
a	tropic	bird	perch	in	the	jungl	.
perch	parakeet	in	a	wire	frame	cage	.
a	snow	owl	land	on	wood	perch	.

Tokenization Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”



“Perching parakeet in a wire frame cage.”



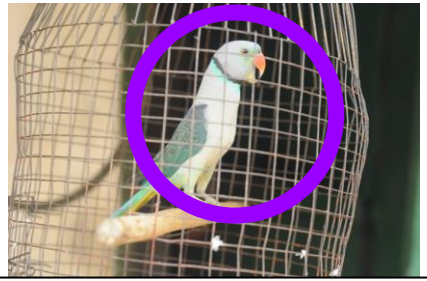
“A snow owl lands on a wooden perch.”

Stemming

a	tropic	bird	perch	in	the	jungl	.
perch	parakeet	in	a	wire	frame	cage	.
a	snow	owl	land	on	wood	perch	.

Tokenization Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”



“Perching parakeet in a wire frame cage.”

“A snow owl lands on a wooden perch.”

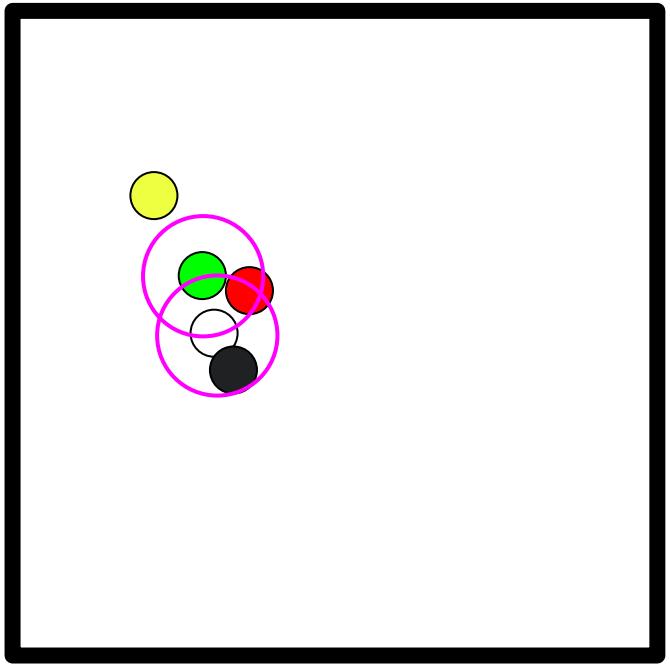
a	tropic	bird	perch	in	the	jungl	.
perch	parakeet	in	a	wire	frame	cage	.
a	snow	owl	land	on	wood	perch	.

Pretrained Language Token Embeddings

- Cosine similarity of “bird”, “owl”, “parakeet” helps share information across training data
- Taking a guess: what are the nearest neighbors of “green” in word2vec embedding space?
  - Blue, white, red, yellow, black, grey, purple, pink, light, gray
- What can we learn for highly polysemous words like “play”?
  - “play guitar”, “play piano”, “play basketball”, “play tag”
- Text-based embeddings help us share information ***only to the extent that words used in a similar context share a visual representation***
  - which is true for, say, birds or trees, but not colors

Pretrained Language Token Embeddings

$\omega(\mathcal{L})$



Training

"A white bunny rabbit held in a green field."

Inference

"A black rabbit watches a red sunset."

Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”

“Perching parakeet in a wire frame cage.”

“A snow owl lands on a wooden perch.”

$$\mathcal{H}(P(Y|a))$$

Image Feats. Considering Language and Vision Retrieval

“A tropical bird perches in the jungle.”

“Perching parakeet in a wire frame cage.”

“A snow owl lands on a wooden perch.”

Object Classification

toucan

cockatoo

great grey owl

https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/

Image Feats. Considering Language and Vision Retrieval

"A tropical bird perches in the jungle."

"Perching parakeet in a wire frame cage."

"A snow owl lands on a wooden perch."

Object Detection

bird	potted plant
bird	boat
bird	aeroplane

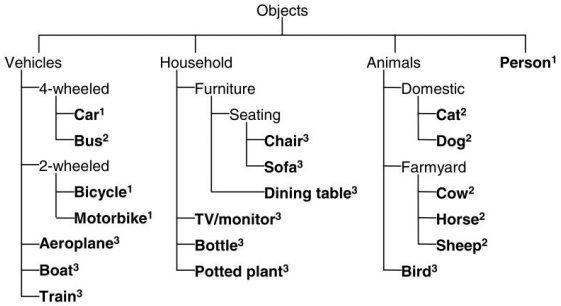
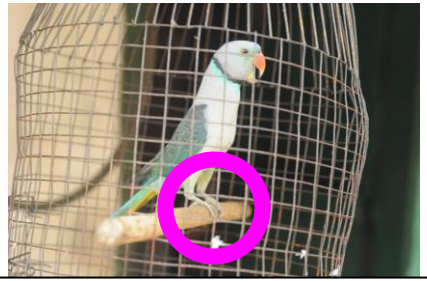
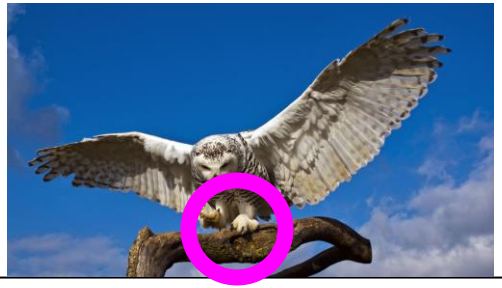


Image Feats. Considering Language and Vision Retrieval

"A tropical bird perches in the jungle."



"Perching parakeet in a wire frame cage."



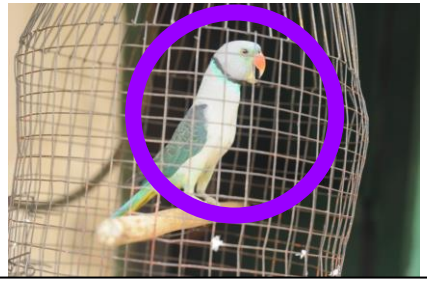
"A snow owl lands on a wooden perch."

bird	potted plant
bird	boat
bird	aeroplane

a	tropic	bird	perch	in	the	jungl	.
perch	parakeet	in	a	wire	frame	cage	.
a	snow	owl	land	on	wood	perch	.

Image Feats. Considering Language and Vision Retrieval

"A tropical bird perches in the jungle."



"Perching parakeet in a wire frame cage."

"A snow owl lands on a wooden perch."

bird	potted plant
bird	boat
bird	aeroplane

a	tropic	bird	perch	in	the	jungl	.
perch	parakeet	in	a	wire	frame	cage	.
a	snow	owl	land	on	wood	perch	.

"A rabbit sits in the palm of a hand."

"A white fox is looking at the camera."

"Children play soccer in a field."

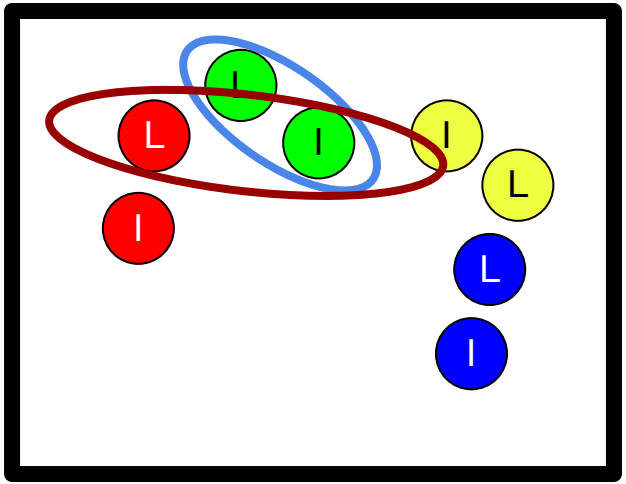
"Men are talking on a basketball court."

Image Embedder $\psi(\mathcal{I})$

Caption Embedder $\omega(\mathcal{L})$

Pull matching imag embeddings together.

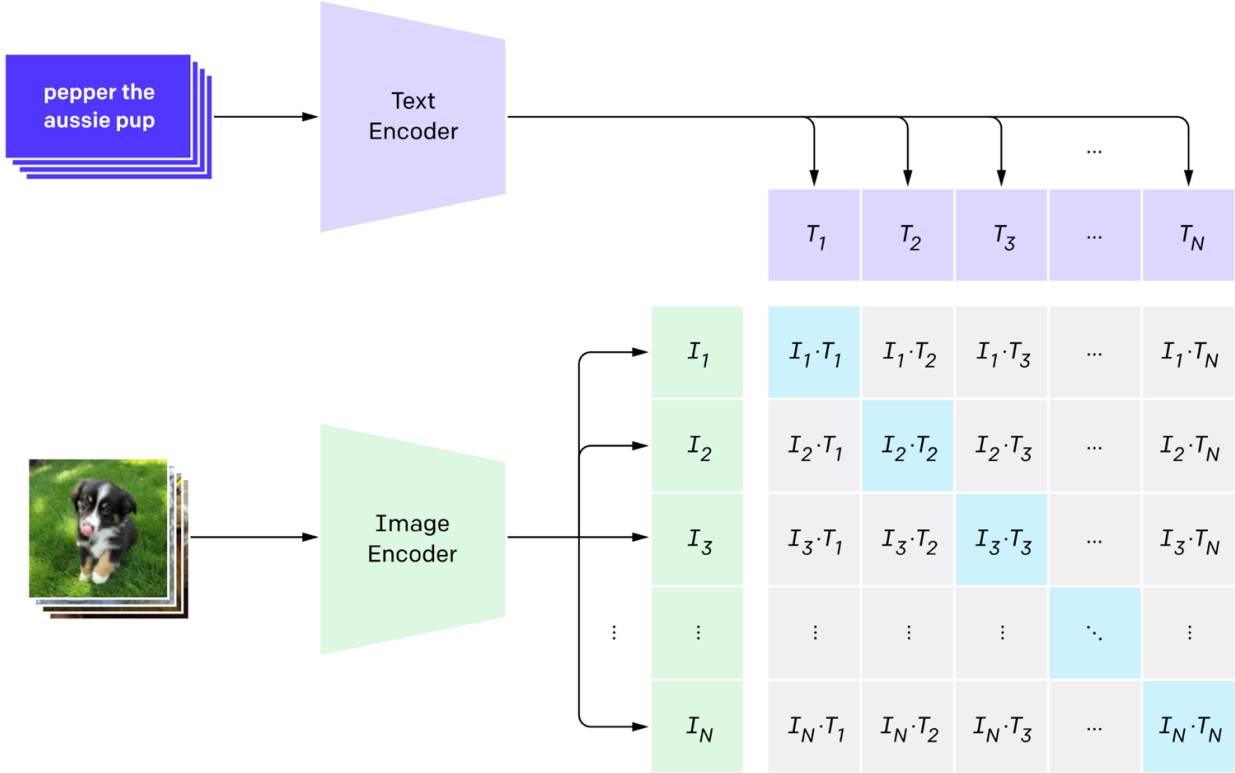
Learned Projection



Learned Projection

Push distractor on embeddings apart.

Contrastive Language–Image Pre-training (CLIP)



https://openai.com/blog/clip/

[Bisk, Holtzman, **Thomason**, et al., EMNLP'20]

Experience Grounds Language - World Scopes

Common Crawl

WS1

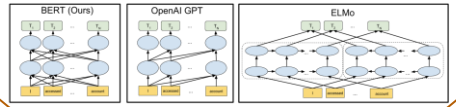
Carefully Annotated

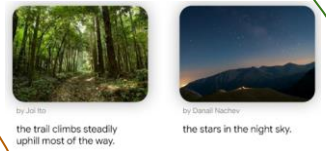
Penn Treebank
Brown Corpus
WordNet

WS2

Unstructured Unlimited Text

Common Crawl
Word2Vec
ELMo, BERT, GPT*

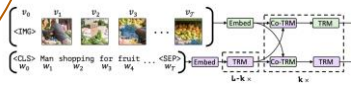




WS3

Text Paired with Sensory Data

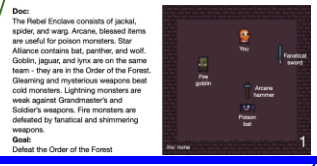
ImageNet, VQA,
ViLBERT,
Video Captioning



WS4

Language and an Embodied Agent

VLN*, IQA,
NL+Games,
RoboNLP



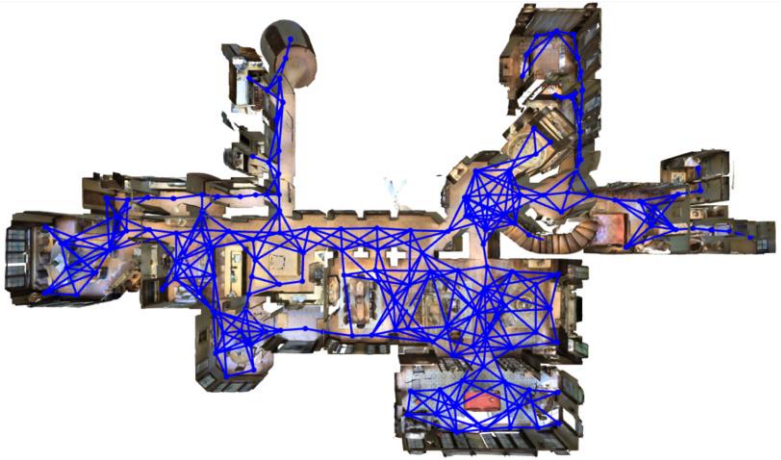
WS5

Social Embodiment

Human-robot collaboration and dialog

Vision-and-Language Navigation (VLN)

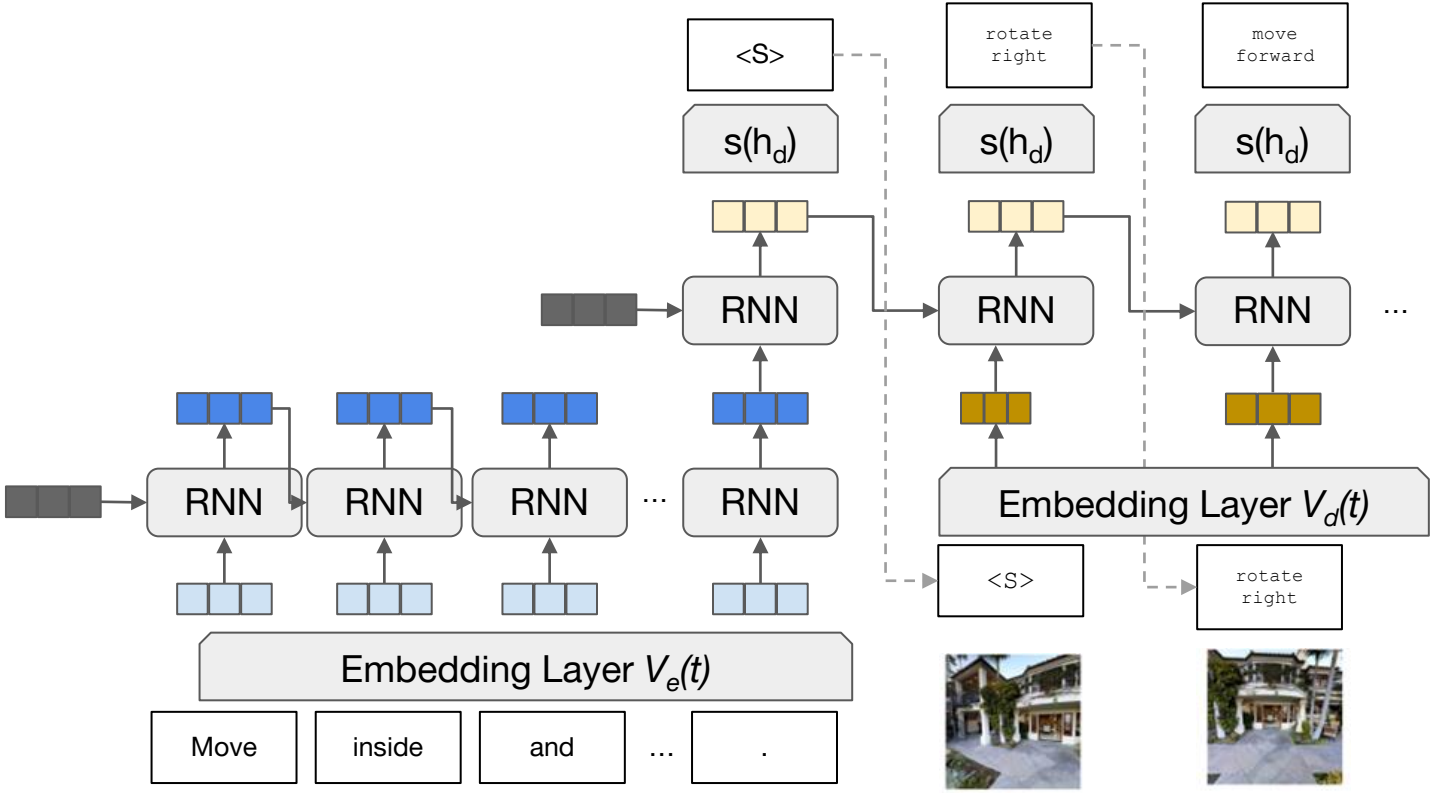
Leave the bedroom, and enter the kitchen. Walk forward, and take a left at the couch. Stop in front of the window.

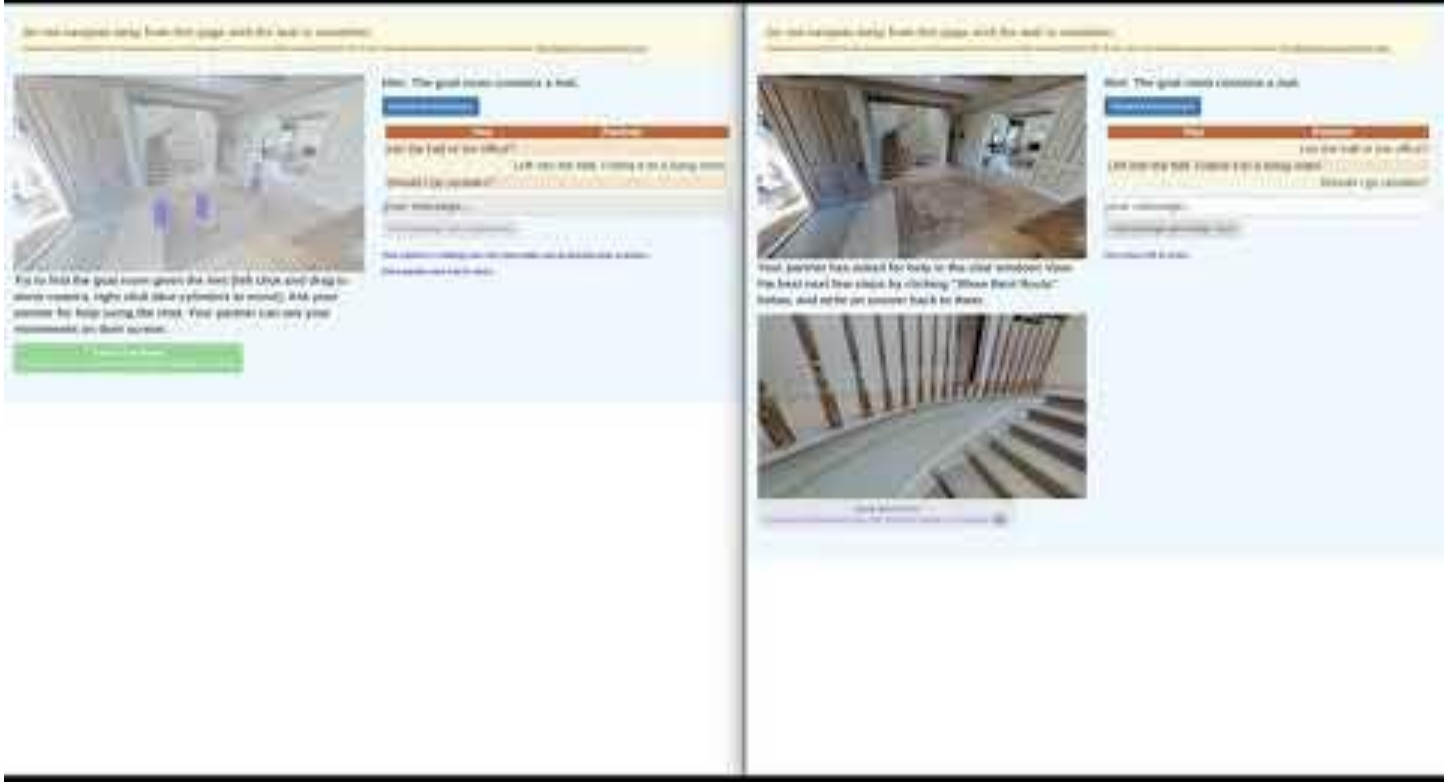


- Source language input: instructions in English
- Target language output: sequence of actions like “forward”, “rotate left”, “rotate right”, and “stop”

https://bringmeaspoon.org/

Machine Translation for Robots

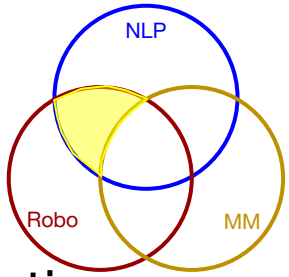




Activity: Cooperative Vision-and-Dialogue Navigation

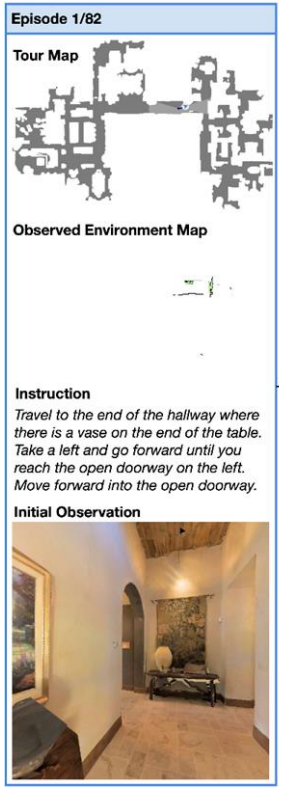
- If we ask a computer to do it, we have to be able to do it too!
- https://cvdn.dev/
  - “Two-player Demo”
  - Scroll to the bottom, click “Start Task”

Language-guided Robots with Vision



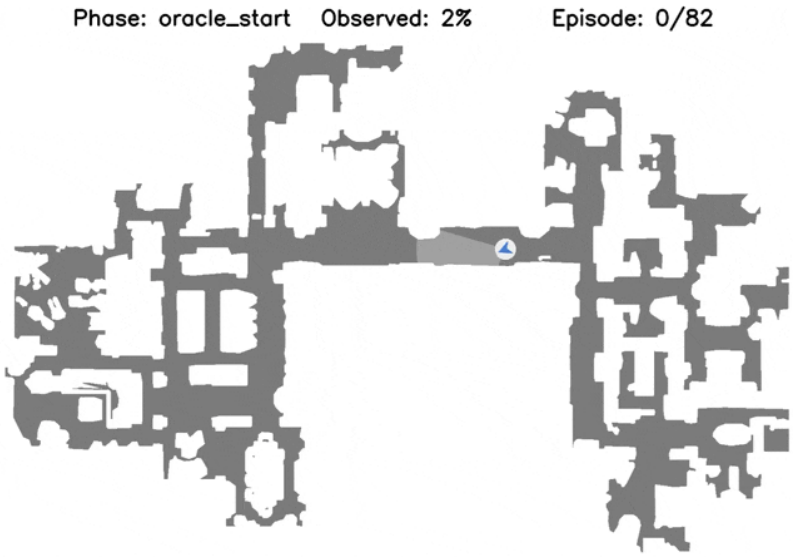
- We are exploring three directions in the space of translating language instructions to agent movement plans:
  - *Memory*: following instructions and building maps together.
  - *Physical Control*: overcoming the gap between simulation and real robot control.
  - *Planning*: translating underspecified language into coherent plans to get a job done.

Memory: Iterative Vision-and-Language Navigation [*in submission*]



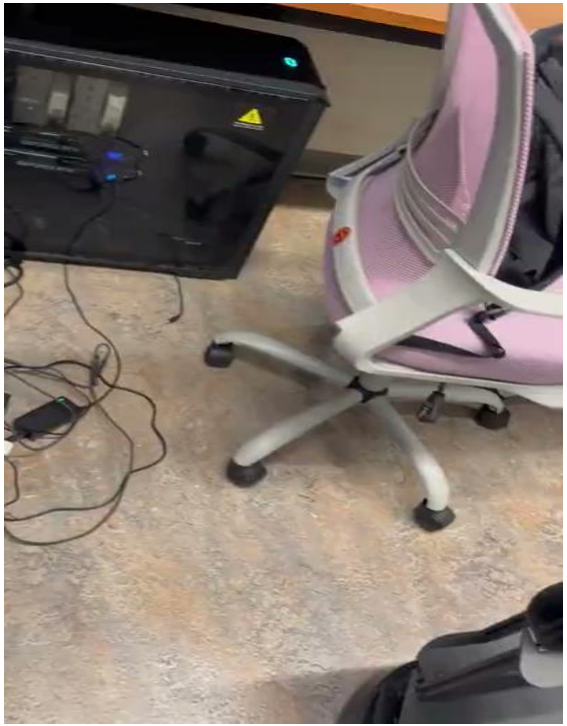
https://jacobkrantz.github.io/ivln

Memory: Iterative Vision-and-Language Navigation



https://jacobkrantz.github.io/ivln

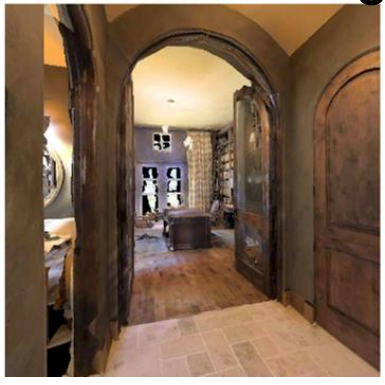
Physical Control: VLN With A Physical Robot



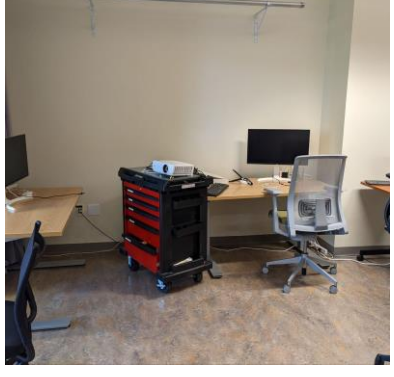
“Go to the couch and stop.”

Physical Control: VLN With A Physical Robot

Simulation Training

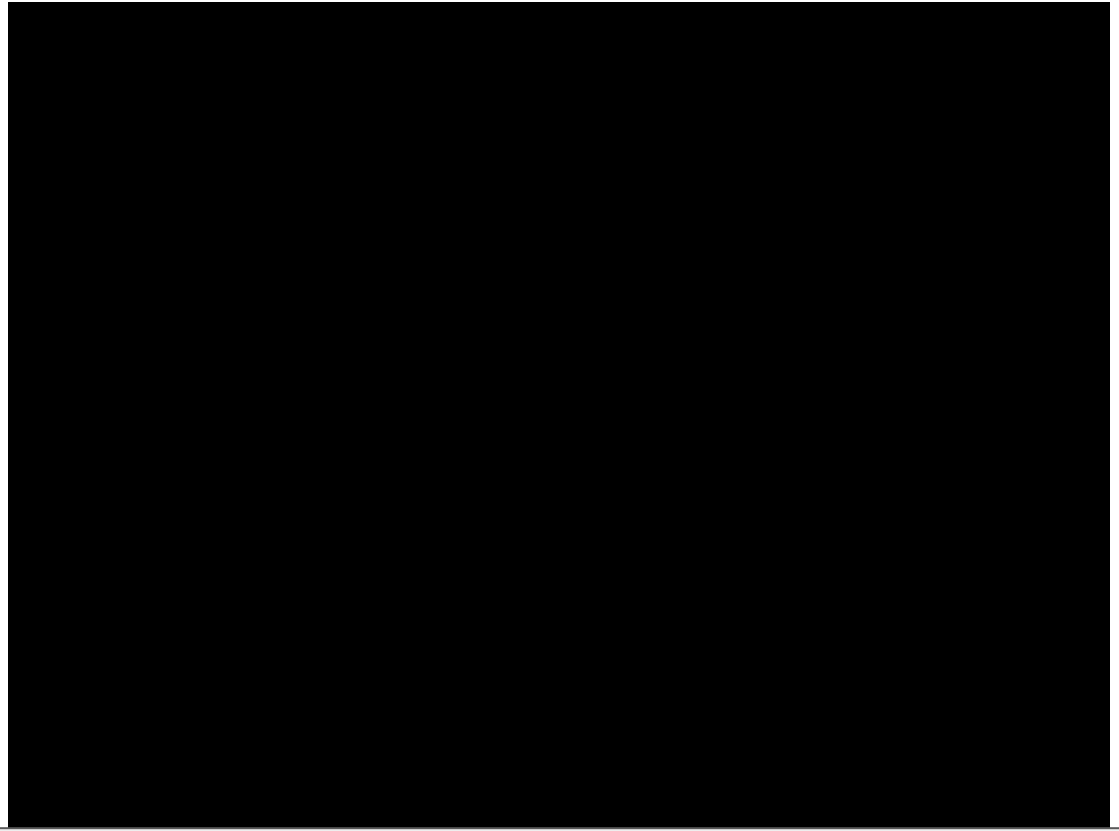


My Lab Space



- Use *mid-level* vision to align virtual and real world spaces

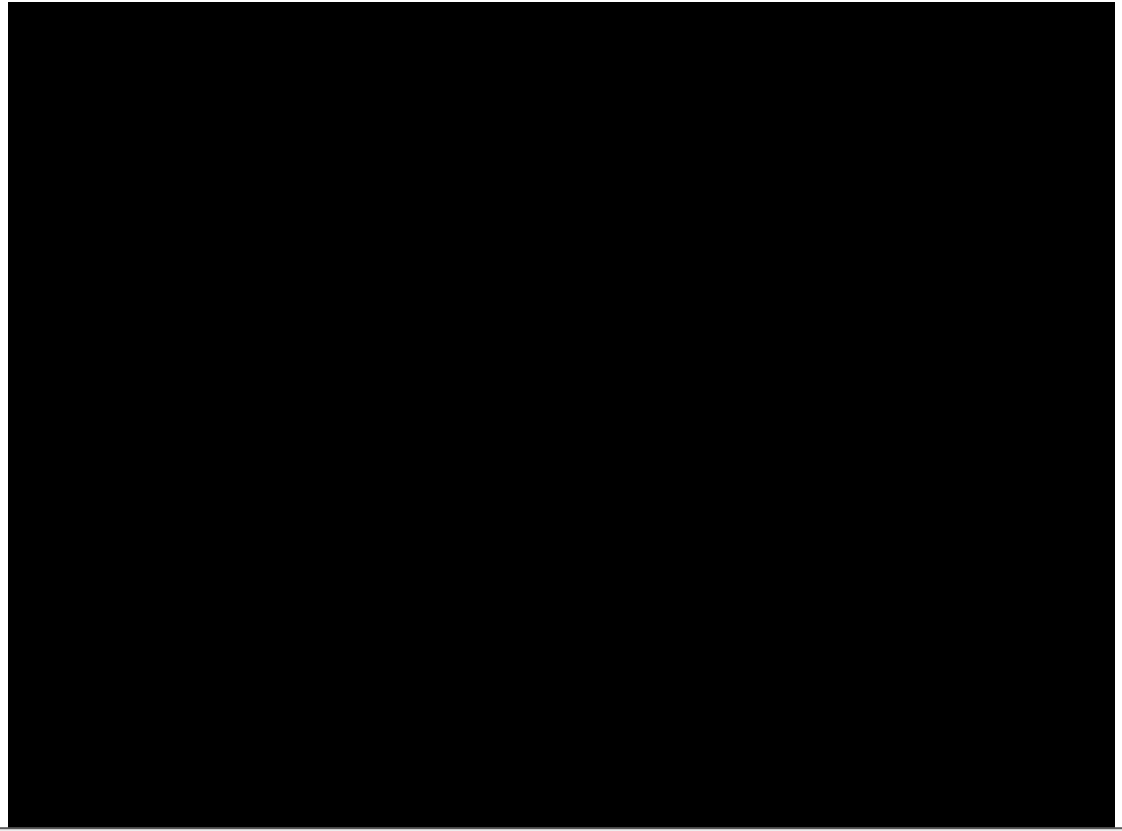
Planning: Prompting for Robot Control [*in submission*]



- LLMs generate actions robots can't do with objects that aren't around
- Pythonic *prompts* can specify robot actions and world objects

https://progprompt.github.io/

Planning: Prompting for Robot Control



- Used to control an agent in a virtual environment
- Perform new tasks in a zero-shot setting

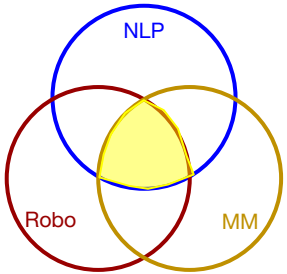
https://progprompt.github.io/

Planning: Prompting for Robot Control

- Also enables generating pick-and-place robot plans!

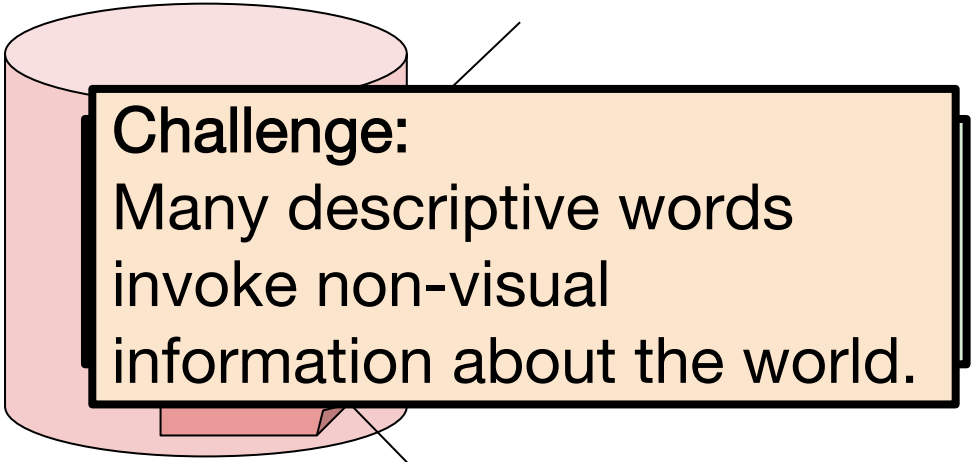
https://progprompt.github.io/

Language-guided Embodied, Multimodal Agents



- Sometimes robot *vision* is not enough.
- A robot is more than a mounted camera; it exists in the physical world and enables *physical sensory experience*
- We can combine language instructions and dialogue for learning with machine vision and perception for robots that learn from people and all their senses

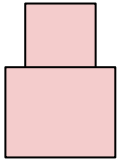
Language-guided Embodied, Multimodal Agents



Challenge:
Many descriptive words invoke non-visual information about the world.

Structured Lang+Viz Data

"The **full** water bottle."



Physical Robot Agent

Sensory Perception Beyond Vision

Look

color, ***shape***, and deep ***visual*** features extracted from images of the object.

Push

Press

Grasp

Lift / Lower

Drop

haptic and ***audio*** features captured from arm and microphone.

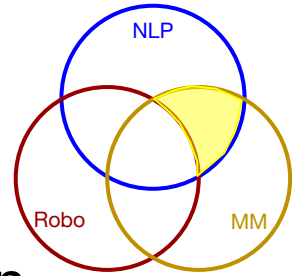
Grounding Multimodal Sensory Perception with Dialogue

Jointly Improving Parsing and Perception for Natural Language Commands through Human-Robot Dialog

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, Ray Mooney

Jointly Improving Parsing and Perception for Natural Language Commands through Human-Robot Dialog

Multimodal NLP: Other Times Text is not Enough



- We are exploring the integration of language-and-vision representations with other sensory and linguistic signals for:
  - *Continual Learning*: creating systems that learn from multiple tasks that involve combinations of language and vision input [*to appear* @ Neurips'22!]
  - *Sign Recognition*: using linguistic priors to better identify ASL signs from videos [*in submission*]
  - *Dementia Detection*: using both language transcriptions and speech signals to detect disfluencies indicating early stage dementia.

Natural Language Processing and Robotics

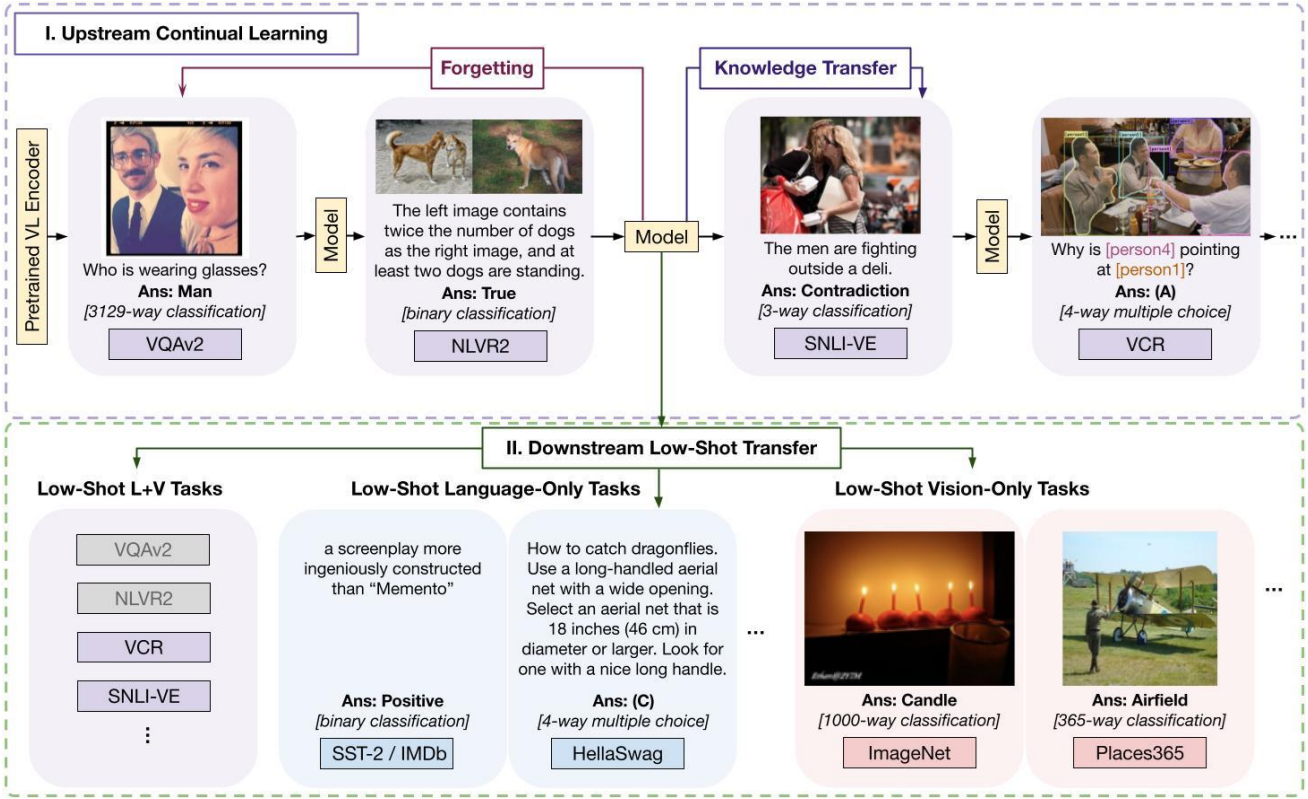
Jesse Thomason

[http://glamor.rocks/]

USC Viterbi School of Engineering

✨GLAMOR

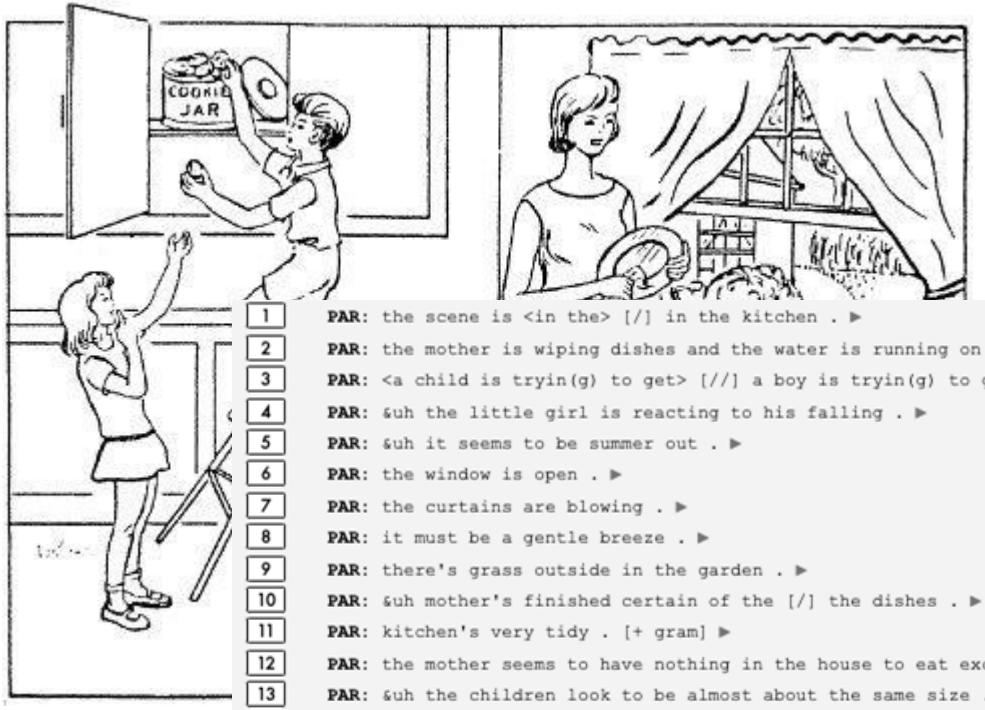
Multimodal Continual Learning



- Explore *adapter fusion* and *split adapters* to learn future tasks, even when a modality disappears

https://github.com/GLAMOR-USC/CLiMB

Early Detection of Dementia




```
1   PAR: the scene is <in the> [/] in the kitchen . ▶
2   PAR: the mother is wiping dishes and the water is running on the floor . ▶
3   PAR: <a child is tryin(g) to get> [//] a boy is tryin(g) to get cookies outta [: out of] a jar and he's about to tip over on a stool . ▶
4   PAR: &uh the little girl is reacting to his falling . ▶
5   PAR: &uh it seems to be summer out . ▶
6   PAR: the window is open . ▶
7   PAR: the curtains are blowing . ▶
8   PAR: it must be a gentle breeze . ▶
9   PAR: there's grass outside in the garden . ▶
10  PAR: &uh mother's finished certain of the [/] the dishes . ▶
11  PAR: kitchen's very tidy . [+ gram] ▶
12  PAR: the mother seems to have nothing in the house to eat except cookies in the cookie jar . ▶
13  PAR: &uh the children look to be almost about the same size . ▶
14  PAR: perhaps they're twins . ▶
15  PAR: they're dressed for summer warm weather . ▶
```

- Predict based on both text *and* audio signals

https://dementia.talkbank.org/

Isolated Sign Language Recognition

- Predict signed word from video frames
- Adding auxiliary predictions can help!
- Handshape
- Location

USC Viterbi School of Engineering

Information Extraction

Muhao Chen
USC
muhaoche@usc.edu

CSCI 544: Applied NLP

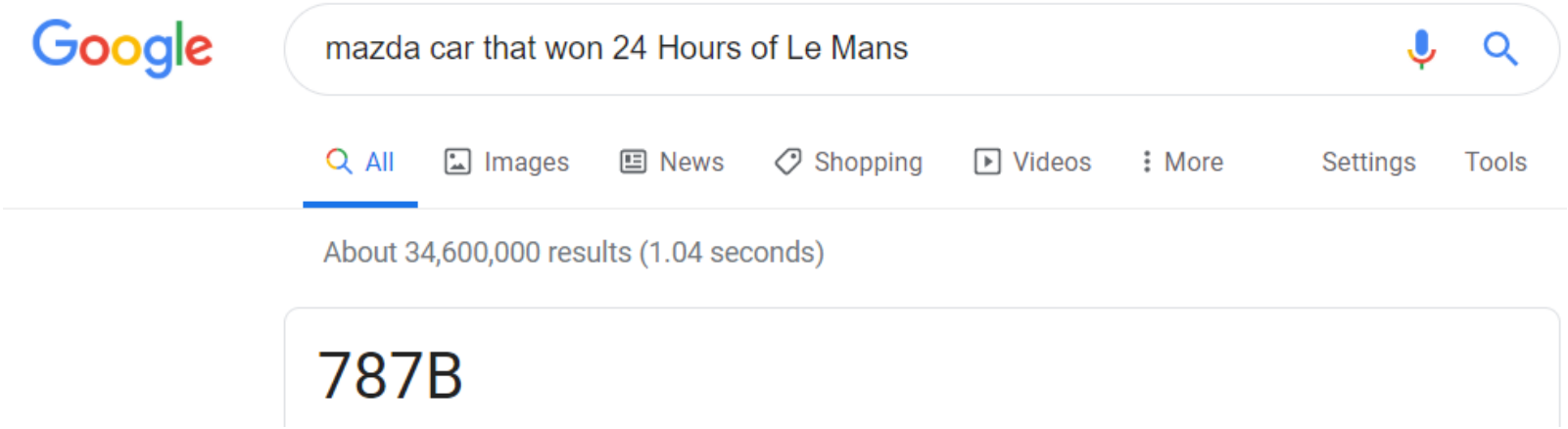
Outline

- Background and Motivation
- Lexical IE
  - Named Entity Recognition
  - Typing
  - Linking
  - Event Extraction
- Relational IE
  - Relation Extraction
  - Coreference Resolution
- Research Directions

AI Needs to Understand the Relations of Concepts

USC Viterbi School of Engineering

QA and Semantic Search:



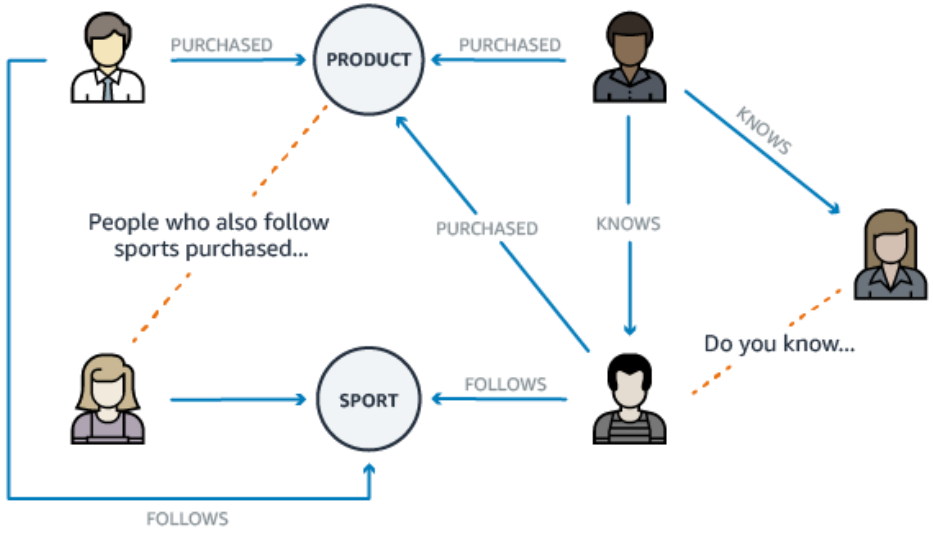
(?car, produced by, Mazda)
(?car, won, 24 Hours of Le Mans)

Google Knowledge Graph

AI Needs to Understand the Relations of Concepts

USC Viterbi School of Engineering

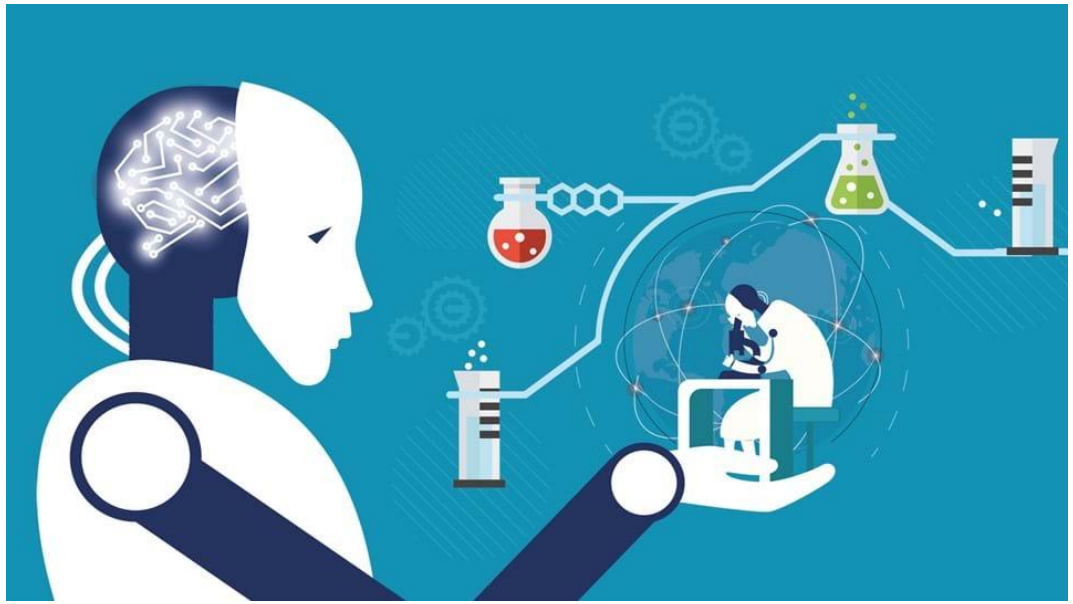
E-Commerce



Co-purchase relations of products

Social relations of users

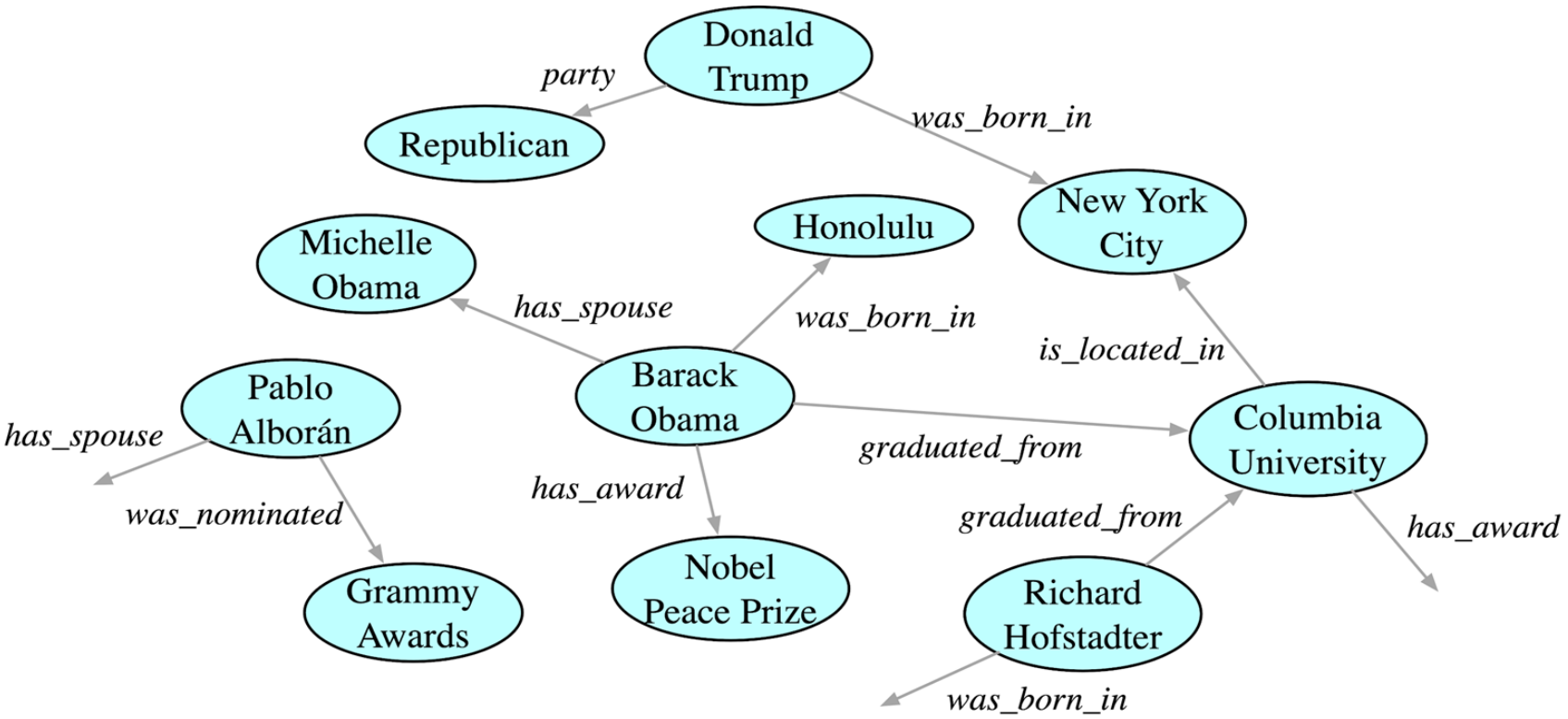
Computational Biology Research



Interactions of molecules and biomolecules.

Structured Knowledge Representation

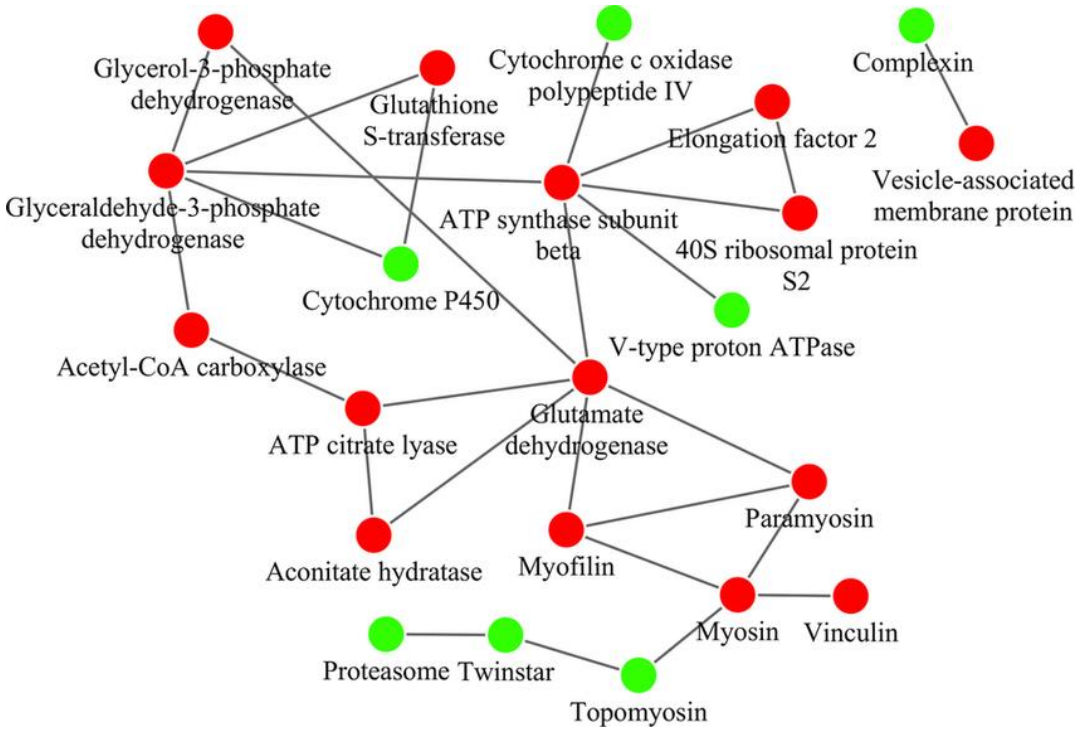
- Knowledge Graphs:
  - Representations for *how concepts are related*



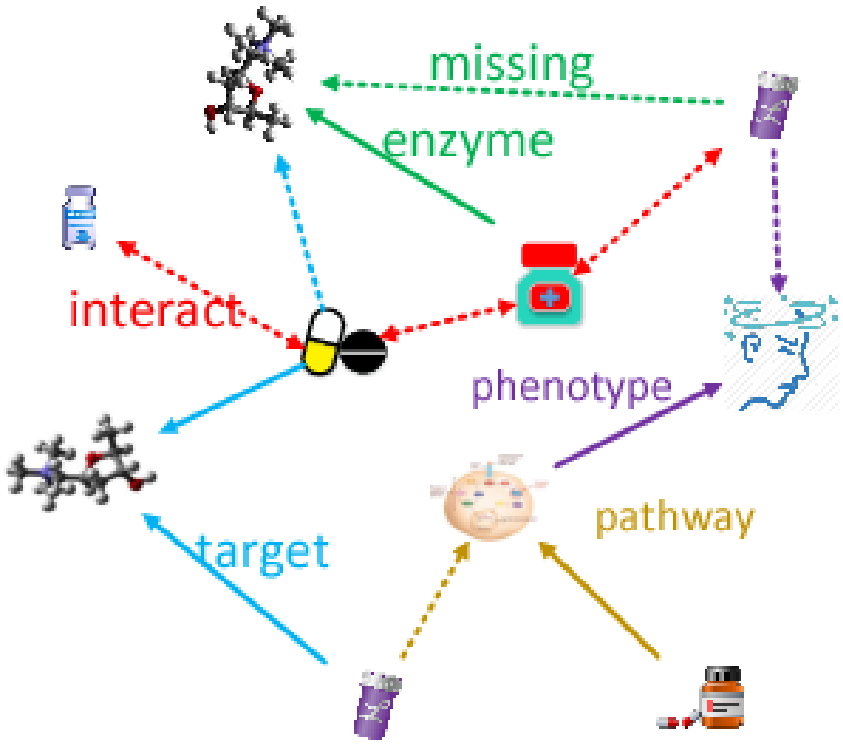
Entities and Relations

Structured Knowledge Representation

- Knowledge Graphs:
  - Representations for *how concepts are related*



Proteins and Interactions



Drug-Target Interactions

Information Extraction (IE)

The process of automatically extracting structural information from unstructured data (e.g. natural language text)

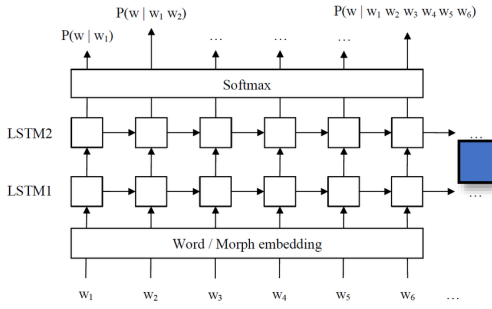
Honolulu

From Wikipedia, the free encyclopedia

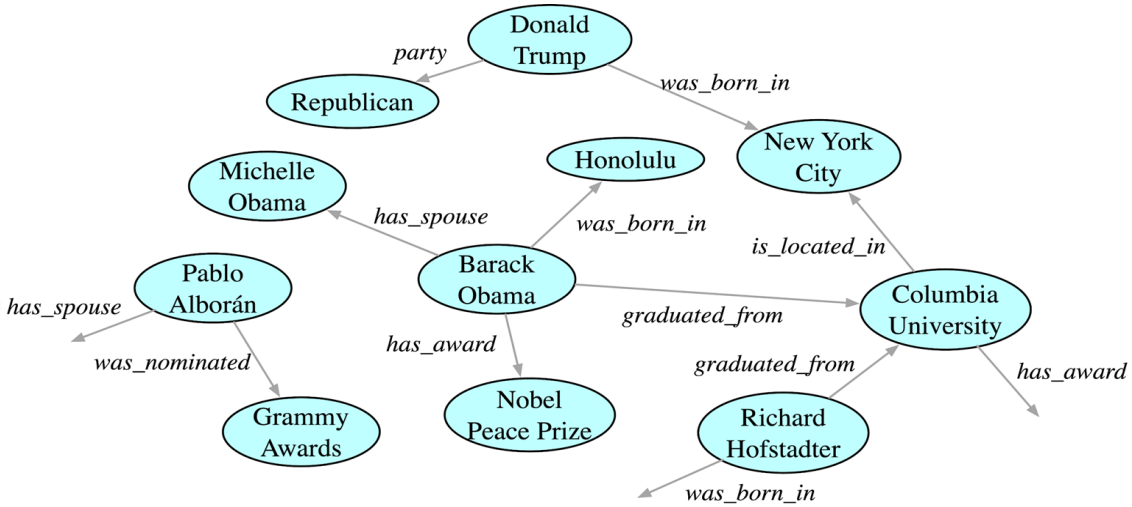
This article is about the largest city and state capital city of Hawai
Honolulu itself, see Honolulu County, Hawaii. For other uses, see

Honolulu (/ˌhɑːnəˈluːluː/;[7] Hawaiian: [honoˈlulu]) is the capital and largest city of the U.S. state of Hawaii, which is located in the Pacific Ocean. It is an unincorporated county seat of the consolidated City and County of Honolulu, situated along the southeast coast of the island of O'ahu,[a] and is the westernmost and southernmost major U.S. city. Honolulu is Hawaii's main gateway to the world. It is also a major hub for international business, finance, hospitality, and military defense in both the state and Oceania. The city is characterized by a mix of various Asian, Western, and Pacific cultures, as reflected in its diverse demography, cuisine, and traditions.

Honolulu means "sheltered harbor"[9] or "calm port" in Hawaiian;[10] its old name, ***Kou***, roughly encompasses the area from Nu'uanu Avenue to Alakea Street and from Hotel Street to Queen Street, which is the heart of the present downtown district.[11] The city's desirability as a port accounts for its historical growth and importance in the Hawaiian archipelago and the broader Pacific region. Honolulu has been the capital of the Hawaiian Islands since 1845, first of the independent Hawaiian Kingdom, and after 1898 of the U.S. territory and state of Hawaii. The city gained worldwide recognition following Japan's attack on nearby Pearl Harbor on December 7, 1941, which prompted decisive entry of the U.S. into World War II; the harbor remains a major naval base, hosting the U.S. Pacific Fleet, the world's largest naval command.[12]

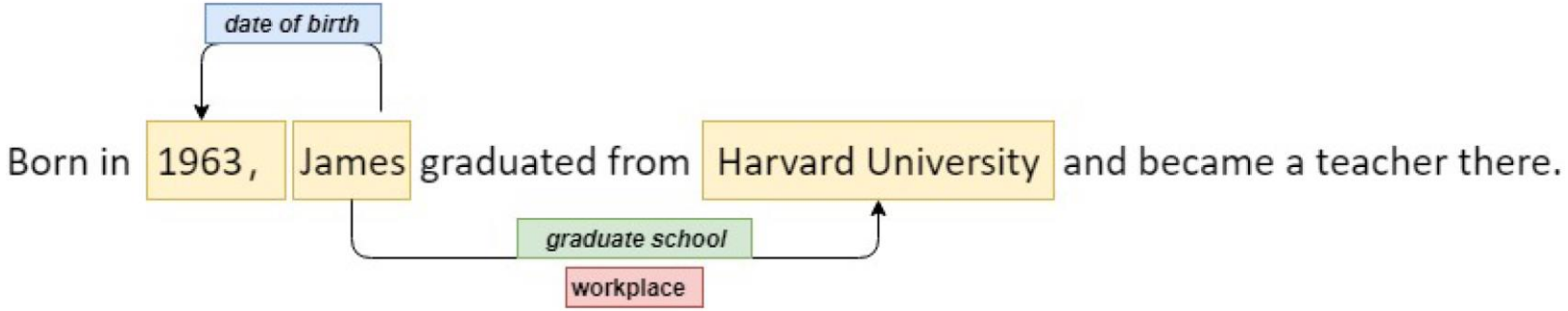


IE Model/System



Two Goals of IE

From the text:



1. Identify the **concepts** (entities, events, terms, etc.)

2. Identify the **relations** (perhaps also properties) of concepts

IE Tasks

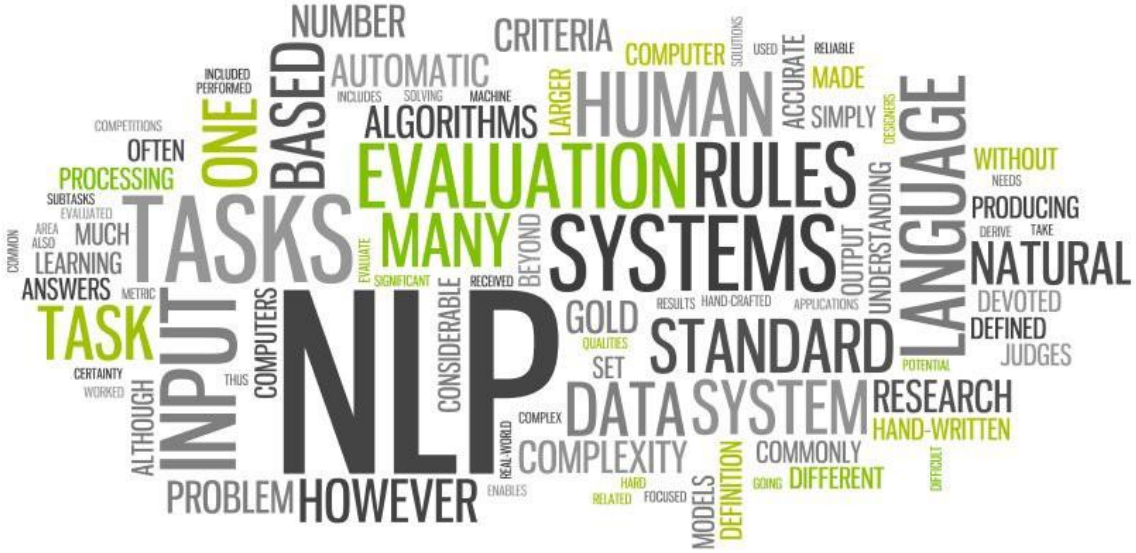
- Lexical IE:
  - Named entity recognition
  - Event extraction
  - Entity (or event) typing
  - Linking/grounding
- Relational IE:
  - Relation extraction
    - Sentence-level relation extraction
    - Document-level relation extraction
  - Coreference resolution

Outline

- Background and Motivation
- Lexical IE
  - Named Entity Recognition
  - Entity (Event) Typing
  - Linking
  - Event Extraction
- Relational IE
  - Relation Extraction
  - Coreference Resolution
- Research Directions

Lexical IE

- Lexical IE: extracting key tokens or token spans from text
  - E.g., recognizing entities, events, terms, keywords, etc.
- Prerequisite of relational IE



Named Entity Recognition (NER)

- Definition: to identify all mentions of named entities (people, organizations, locations, dates)
  - One of the well-recognized lexical IE tasks

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```

↓

```
[PERS Pierre Vinken] , 61 years old , will join
[ORG IBM] 's board as a nonexecutive director
[DATE Nov. 2] .
```

Named Entity Recognition (NER)

- NER can be learned as sequence tagging
- Define entity-specific sequence tags
  - B-PERS, B-DATE,…: beginning of a mention of a person/date...
  - I-PERS, I-DATE,…: inside of a mention of a person/date...
  - O: outside of any mention of a named entity

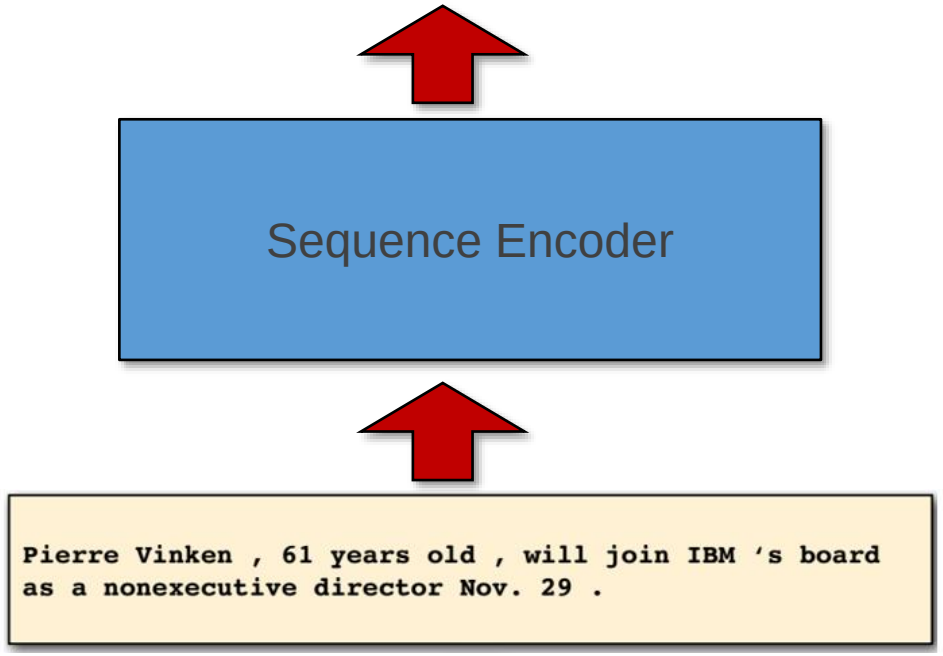
```
[PERS Pierre Vinken] , 61 years old , will join
[ORG IBM] 's board as a nonexecutive director
[DATE Nov. 2] .
```

```
Pierre_B-PERS Vinken_I-PERS ,_O 61_O years_O old_O ,_O
will_O join_O IBM_B-ORG 's_O board_O as_O a_O
nonexecutive_O director_O Nov._B-DATE 29_I-DATE ._O
```

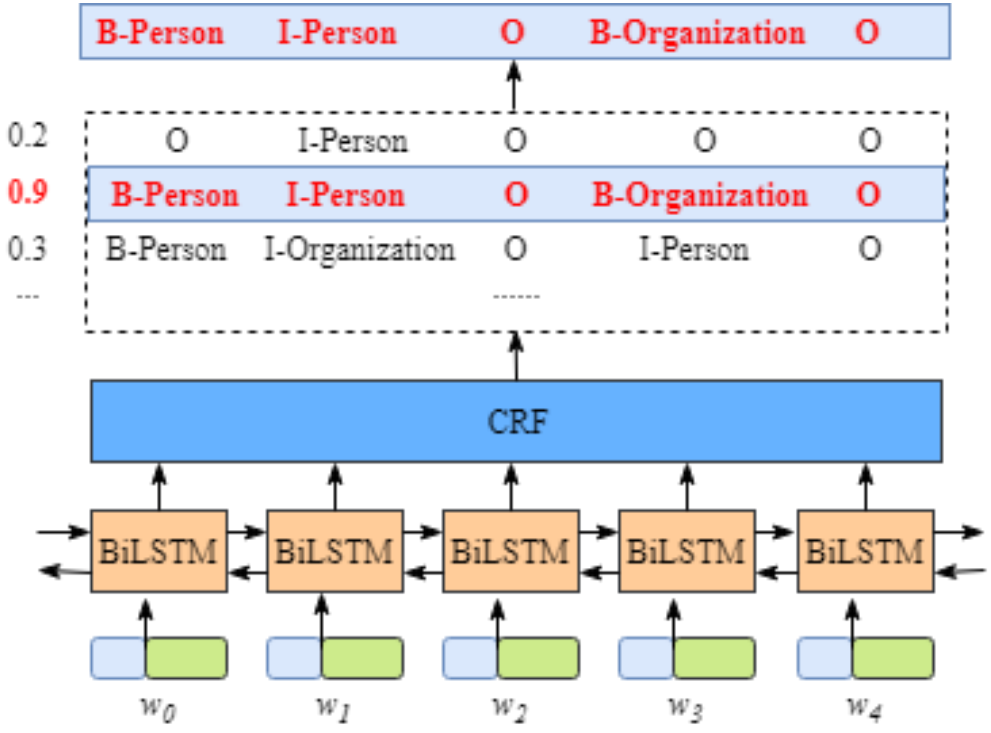
Named Entity Recognition (NER)

- NER can be learned as sequence tagging

Token-level classification
- B-PERS? B-DATE? O?

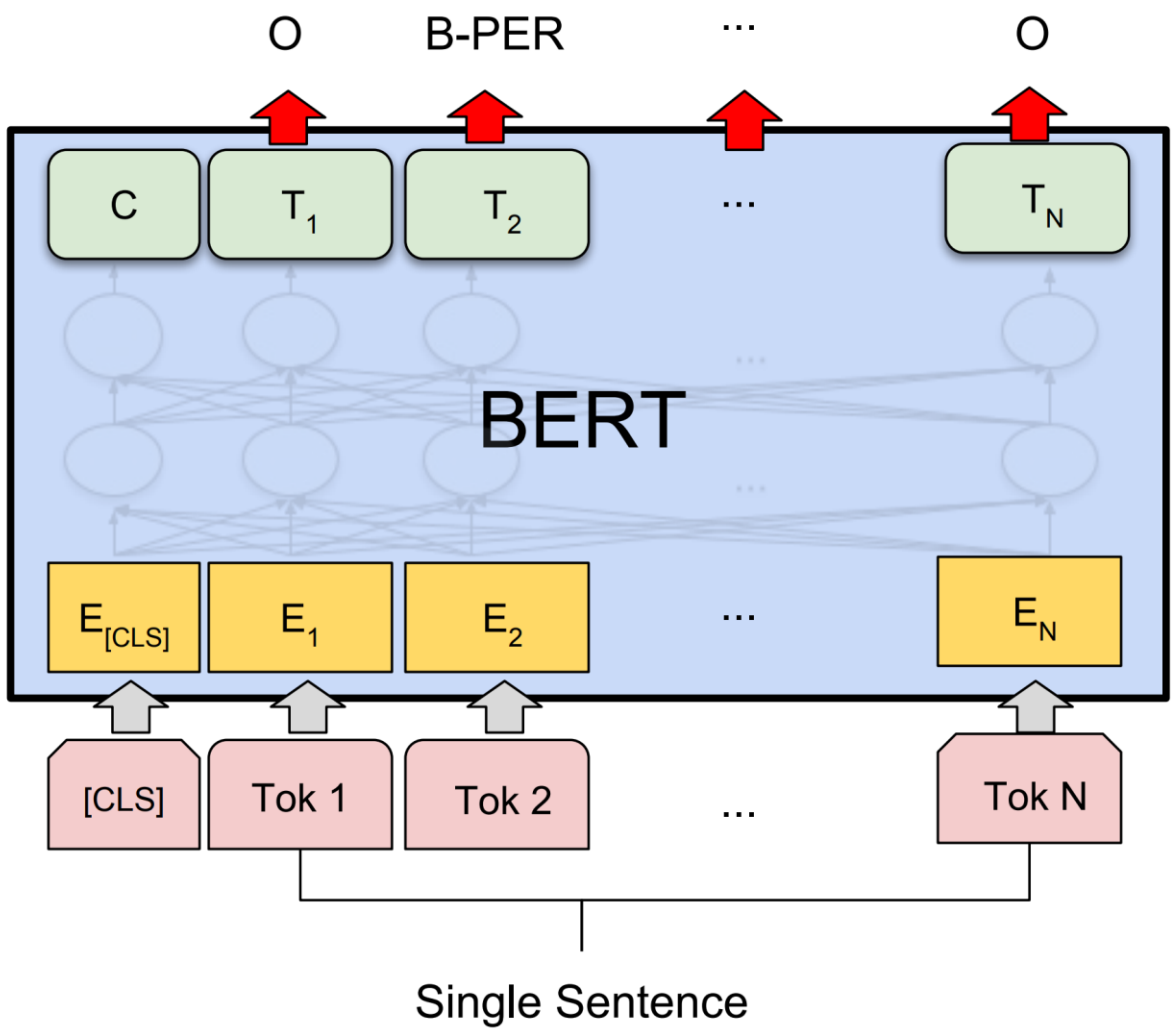


Bi-LSTM-CRF for NER



BERT-CRF for NER

- Pre-trained Transformer (e.g., BERT) as sequence encoder



That's NER

Dark Souls III is a 2016 action role-playing video game developed by FromSoftware and published by Bandai Namco Entertainment for PlayStation 4, Xbox One, and Microsoft Windows. It is the fourth instalment of the Souls series and the final instalment of the Dark Souls trilogy.

Sequence Tagging

Dark Souls III (MISC) is a 2016 action role - playing video game developed by FromSoftware (ORG) and published by Bandai Namco Entertainment (ORG) for PlayStation 4 (MISC), Xbox One (MISC), and Microsoft Windows (MISC). It is the fourth instalment of the Souls (MISC) series and the final instalment of the Dark Souls (MISC) trilogy .

Entity Typing

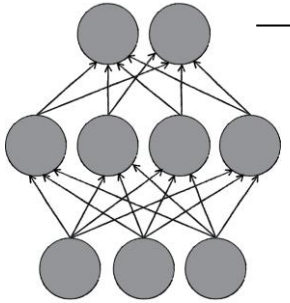
Australia beat New Zealand by 2-0.

- The NER model recognizes Australia as a named entity.
  - Is Australia most likely to be a *country*, a soccer team, or an *athlete*?
- Entity typing: contextual multi-class classification that decides the (fine-grained) type of the entity.

Entity Typing

Multi-class classifier

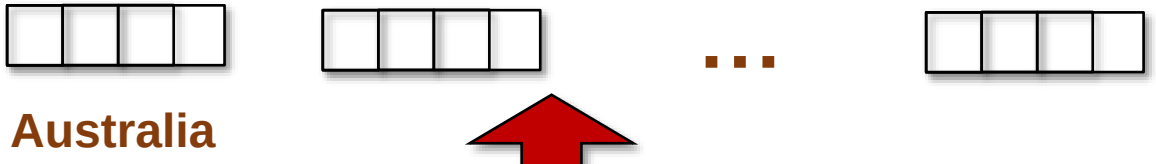
Feed-forward NN



$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)}$$

Cross-entropy loss

Token Embedding



Sequence Encoder

BERT

Text

Australia beat New Zealand by 2-0.

The basic solution

Entity Typing as NLI

USC Viterbi School of Engineering

What if there are thousands of types? What if there are unseen types?

- A multi-class classifier can handle neither of them

New solution: Entity Typing as Natural Language Inference [Li et al. TACL 2022]

Premise

GT-R can accelerate to 60 mph in less than three seconds, while GR Supra cannot.

Hypothesis

GT-R is faster than GR Supra.

Run Model

Entailment	91.2%
Contradiction	1%
Neutral	7.7%

Premise

GT-R can accelerate to 60 mph in less than three seconds, while GR Supra cannot.

Hypothesis

GT-R is slower than GR Supra.

Run Model

Entailment	0.4%
Contradiction	99.4%
Neutral	0.3%

Natural Language Inference (AllenNLP demo)

Entity Typing as NLI

USC Viterbi School of Engineering

What if there are thousands of types? What if there are unseen types?

- A multi-class classifier can handle neither of them

New solution: Entity Typing as Natural Language Inference [LYC, TACL 2022]

Premise

Australia beat New Zewland by 2:0.

Hypothesis

Australia is a soccer team.

Run Model

Entailment	68.5%
Contradiction	0.1%
Neutral	31.5%

Premise

Australia beat New Zewland by 2:0.

Hypothesis

Australia is a pianist.

Entailment	0%
Contradiction	99.5%
Neutral	0.5%

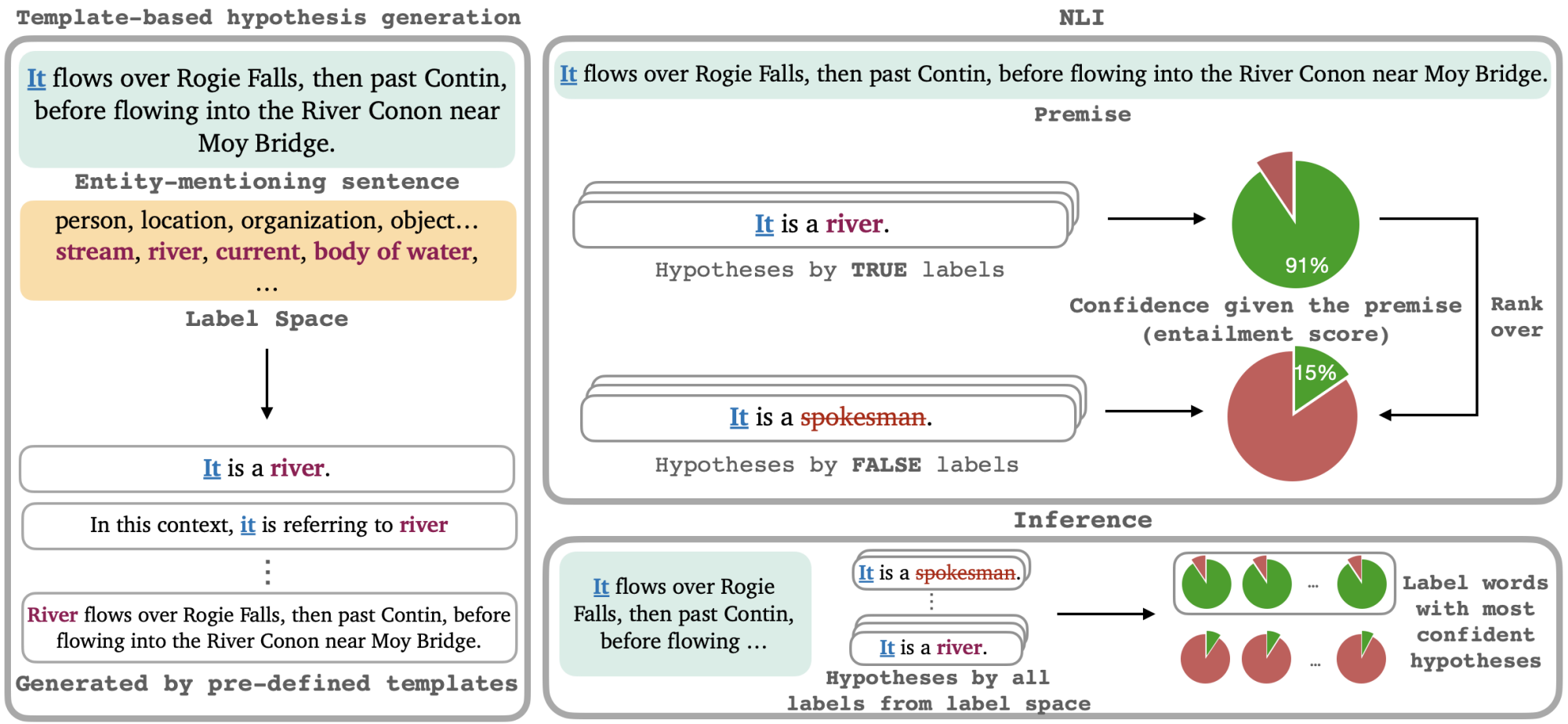
Type inference can be transformed to NLI

Entity Typing as NLI

What if there are thousands of types? What if there are unseen types?

- A multi-class classifier can handle neither of them

New solution: Entity Typing as Natural Language Inference [LYC, TACL 2022]



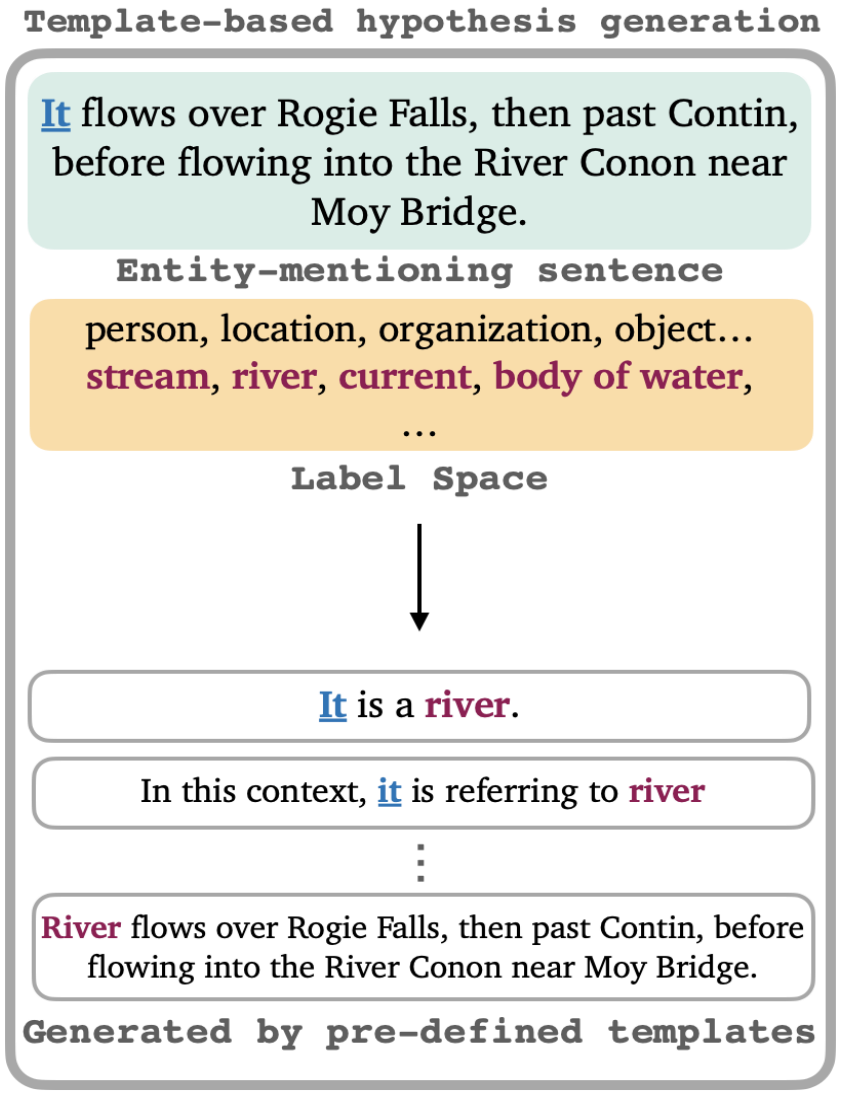
Fine-tuning NLI + Learning-to-rank

Entity Typing as NLI

USC Viterbi School of Engineering

Template-based Type Description Generation

- Taxonomic Statement: " *[ENTITY]* is a *[LABEL]* "
- Contextual Explanation*:* " In this context*, [ENTITY]* is referring to *[LABEL]* "
- *Label Substitution:* Replace the *[ENTITY]* in the sentence with *[LABEL]*



Li et al. *Ultra-fine Entity Typing with Indirect Supervision from Natural Language Inference*. **TACL 2022**

Entity Typing as NLI

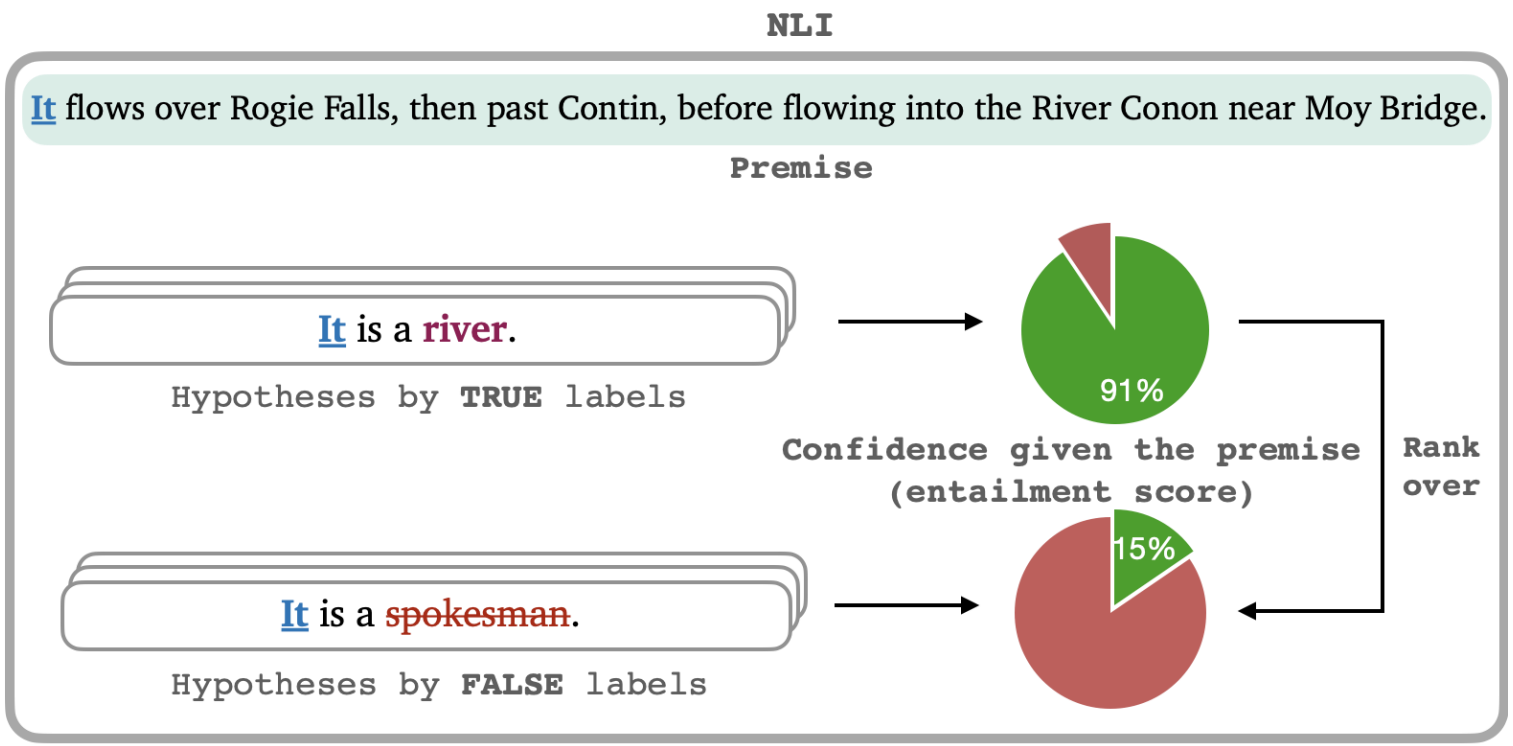
USC Viterbi School of Engineering

Learning-to-rank

- Positive: correct type description
- Negative: randomly sampled incorrect type descriptions

Entailment score should be higher on any positive descriptions

(Also captures label dependencies: Alice is a pianist => Alice is an artist**)**



$$\mathcal{L}_t = [\varepsilon(s, T(a')) - \varepsilon(s, T(a)) + \gamma]_+$$

Li et al. *Ultra-fine Entity Typing with Indirect Supervision from Natural Language Inference*. **TACL 2022**

Entity Typing as NLI

- Typing as NLI is more precise than typing as classification

Model		OntoNotes		FIGER	
		macro-F1	micro-F1	macro-F1	micro-F1
Hierarchy-Typing (Chen et al., 2020b)		73.0	68.1	83.0	79.8
Box4Types (Onoe and Durrett, 2020)		77.3	70.9	79.4	75.0
DSAM (Hu et al., 2021)		83.1	78.2	83.3	81.5
SEPREM (Xu et al., 2021)		–	–	86.1	82.1
MLMET (Dai et al., 2021)		85.4	80.4	–	–
LITE	pre-trained on NLI+UFET	**86.6**	**81.4**	80.1	74.7
	NLI+task-specific training	86.4	80.9	**86.7**	**83.3**

Zero-shot transferring the *NLI Model* trained on UFET even outperforms *supervised SOTA methods* on OntoNotes and FIGER

However, inference cost is the downside. (consider *M* test cases and *N* labels)

- *MXN* times of encoding for NLI
- *M* times of encoding for a classifier

Li et al. *Ultra-fine Entity Typing with Indirect Supervision from Natural Language Inference*. **TACL 2022**

Entity Linking

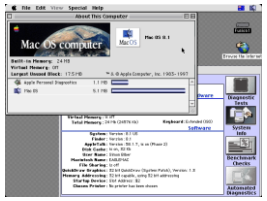
It's a version of ***Chicago*** – the standard classic Macintosh menu font, with that distinctive thick diagonal in the "N".	***Chicago*** was used by default for Mac menus through MacOS 7.6, and OS 8 was released mid-1997..	***Chicago VIII*** was one of the early 70s-era ***Chicago*** albums to catch my ear, along with ***Chicago II***.

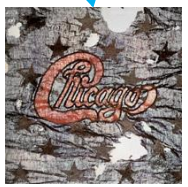
EL: grounding (incl. disambiguating) entity mentions to real-world objects

Slides are adapted from Dan Roth

Entity Linking

It's a version of ***Chicago*** – the standard classic ***Macintosh*** menu font, with that distinctive thick diagonal in the "N".	***Chicago*** was used by default for ***Mac*** menus through ***MacOS 7.6***, and ***OS 8*** was released mid-1997..	***Chicago VIII*** was one of the early 70s-era ***Chicago*** albums to catch my ear, along with ***Chicago II***.





EL: grounding (incl. disambiguating) entity mentions to real-world objects

Wikification: Linking Entities to Wikipedia

USC Viterbi School of Engineering

Blumenthal (D) is a candidate for the U.S. Senate seat now held by Christopher Dodd (D), and he has held a commanding lead in the race since he entered it. But the Times report has the potential to fundamentally reshape the contest in the Nutmeg State.

Richard Blumenthal
From Wikipedia, the free encyclopedia

Democratic Party (United States)
From Wikipedia, the free encyclopedia

United States Senate
From Wikipedia, the free encyclopedia

Blumenthal (D) is a candidate for the U.S. Senate seat now held by Christopher Dodd (D), and he has held a commanding lead in the race since he entered it. But the Times report has the potential to fundamentally reshape the contest in the Nutmeg State.

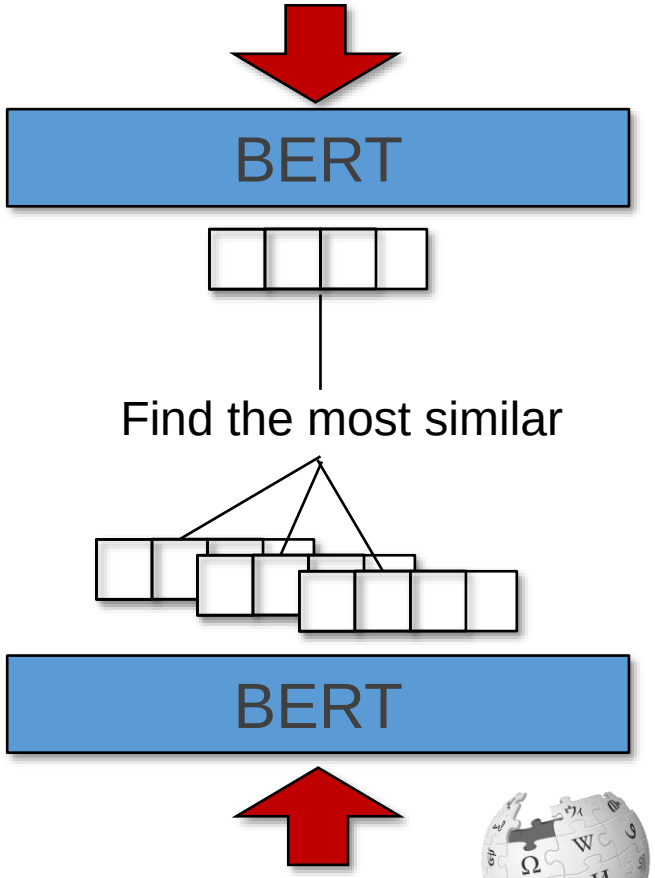
Chris Dodd
From Wikipedia, the free encyclopedia

The New York Times
From Wikipedia, the free encyclopedia

Connecticut
From Wikipedia, the free encyclopedia

A common solution

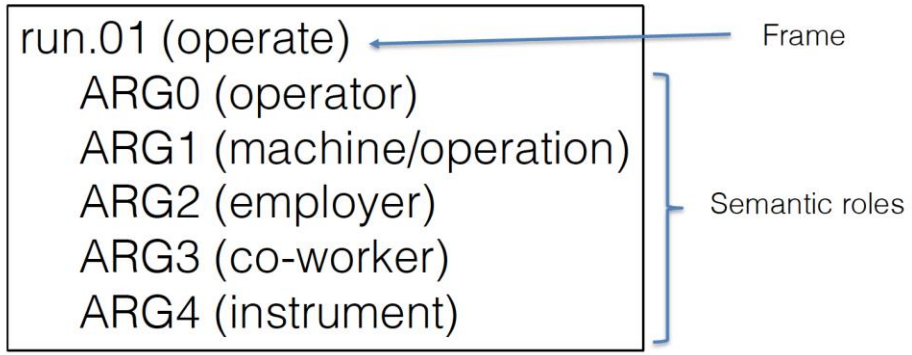
Blumenthal (D) is a candidate for the …

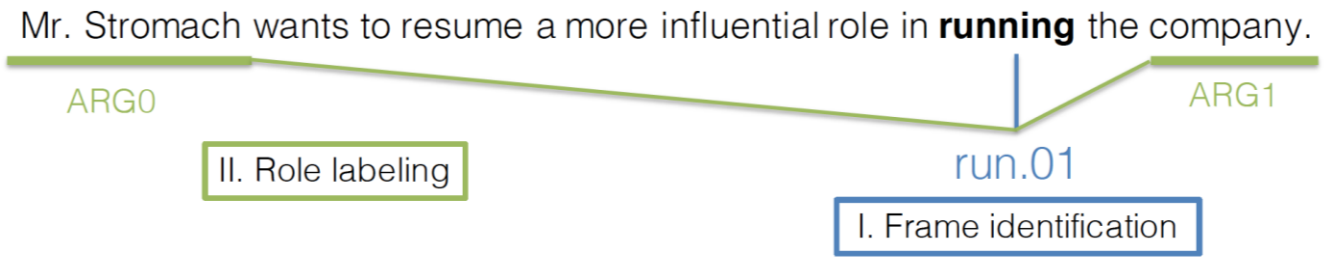


{ All *Blumenthal's* with hyperlinks in WIKIPEDIA The Free Encyclopedia }

Event Extraction / Semantic Role Labelling

- Predicates: some words represent events
- Arguments: specific roles that involves in the event
- PropBank: frames of event components





Leave this to the lecture on 11/30

Outline

- Background and Motivation
- Lexical IE
  - Named Entity Recognition
  - Typing
  - Linking
  - Event Extraction
- Relational IE
  - Relation Extraction
  - Coreference Resolution
- Research Directions

After Identifying the Concepts

Entities, events, etc…

- Concepts are not standalone

Bill Gates founded Microsoft in 1975.

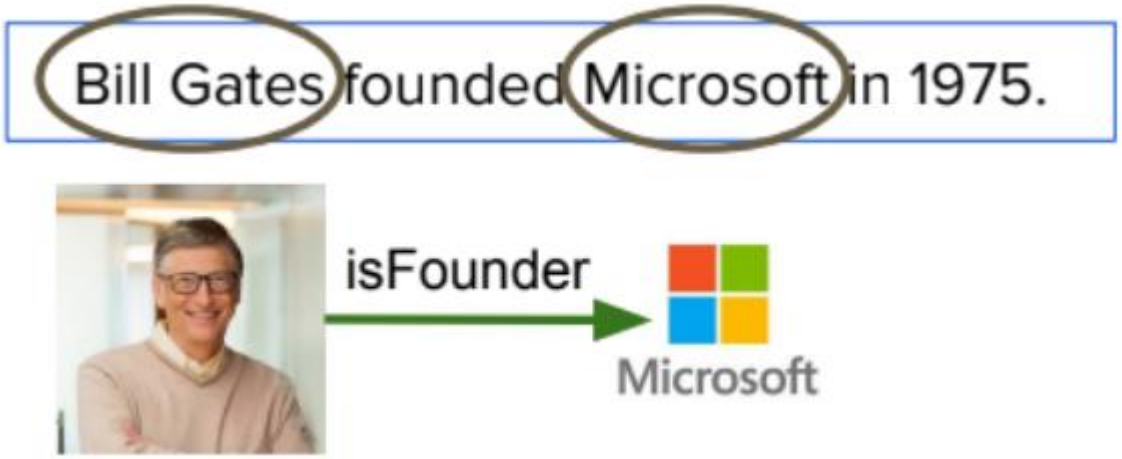
Microsoft

He has been the CEO until the year of 2000.

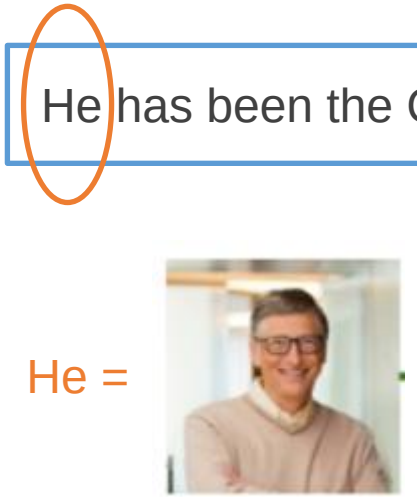
He?

Relational IE

- Induce the structure of concepts from the output of lexical IE



Relation



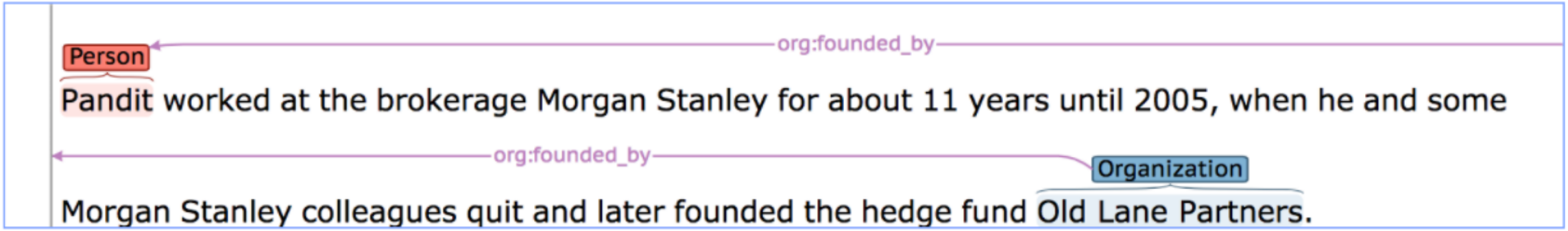
Co-reference

Representative tasks

- **Relation extraction**
  - Sentence-level relation extraction
  - Document-level relation extraction
- **Coreference resolution**

Sentence-level Relation Extraction

- Given a sentence mentioning two entities/events, identify the relation between them.



Sentence-level Relation Extraction

USC Viterbi School of Engineering

- Can we simply model this problem as lexical IE?
  - ✗ The relation may not be directly mentioned in a text span.

In the 1992 24 Hours of Le Mans, the Mazda's 787b outran the first runner-up (Jaguar XJR-12) by 2 laps.

Extraction: Mazda's 787b *is the winner of* 1992 24 Hours of Le Mans

"Winner" has not been mentioned in the sentence.

Sentence-level Relation Extraction

USC Viterbi School of Engineering

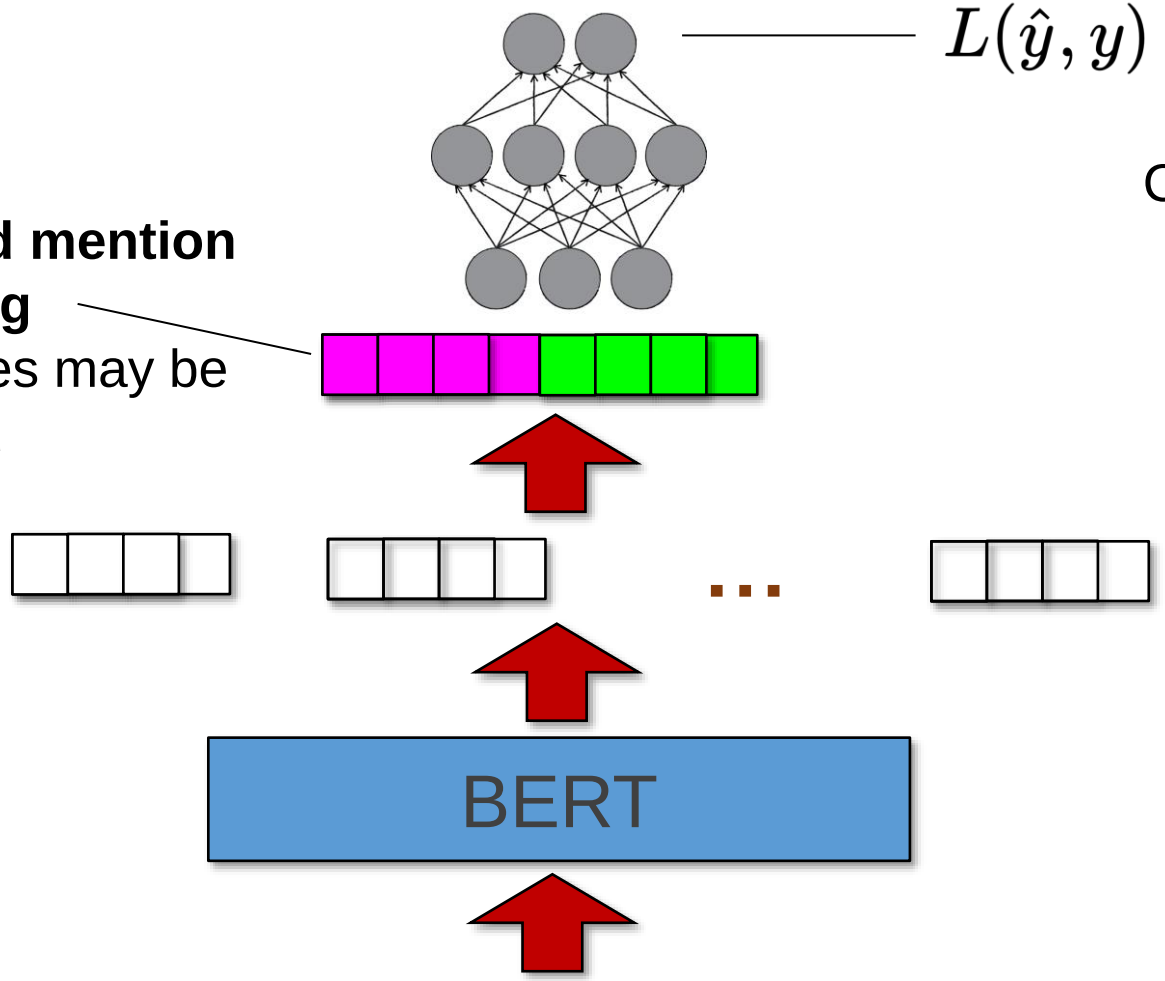
Multi-class classifier

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)}$$

Cross-entropy loss

Contextualized mention pair embedding

- Mention types may be incorporated



The basic solution

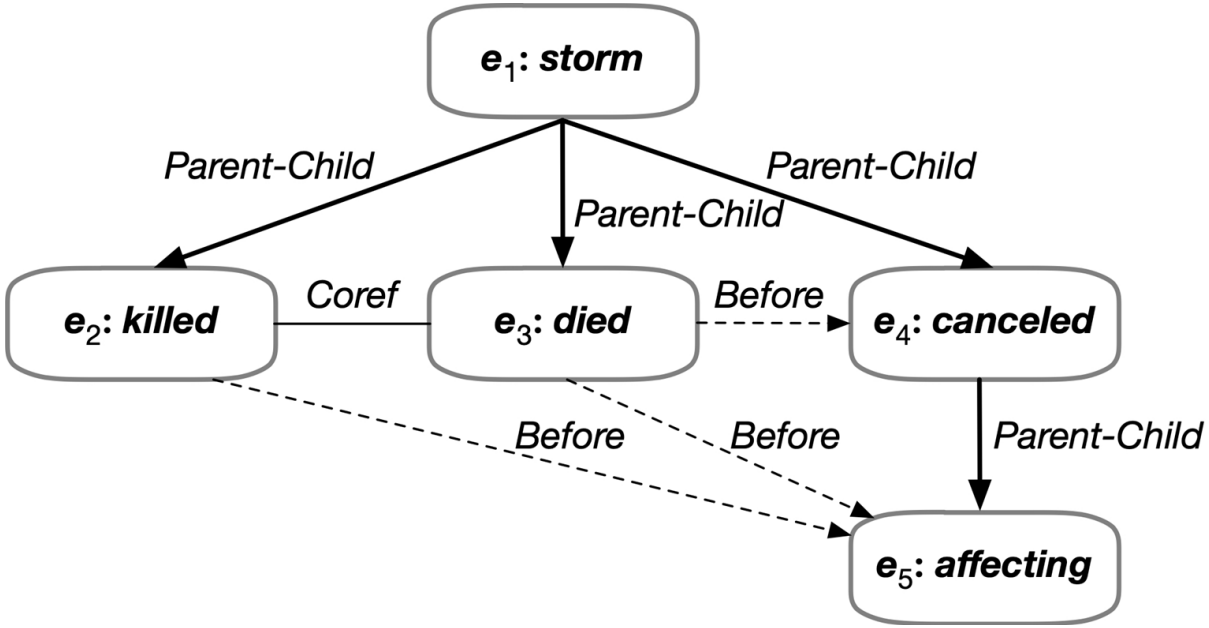
In the 1992 24 Hours of Le Mans, the Mazda's 787b outran the first runner-up (Jaguar XJR-12) by 2 laps.

Document-level Relation Extraction

Infer the relation for **each pair of entities/events** in the document.

On Tuesday, there was a typhoon-strength (***e***$_1$***:storm***) in Japan. One man got (***e***$_2$***:killed***) and thousands of people were left stranded. Police said an 81-year-old man (***e***$_3$***:died***) in central Toyama when the wind blew over a shed, trapping him underneath. Later this afternoon, with the agency warning of possible tornadoes, Japan Airlines (***e***$_4$***:canceled***) 230 domestic flights, (***e***$_5$***:affecting***) 31,600 passengers.

A paragraph from HiEve



Temporal Relations
Subevent Relations (Memberships)
Coreference

We conduct multi-class classification on every mention pair

Document-level Relation Extraction

USC Viterbi School of Engineering

Relation themselves are interrelated

Symmetry

e3:died is BEFORE *e4:canceled* => *e4:canceled* is AFTER *e3:died*

Transitivity

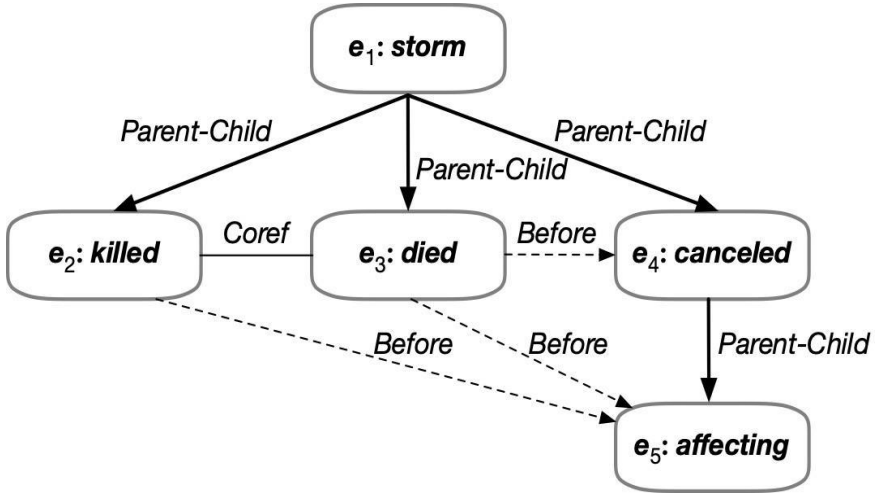
e1:storm is PARENT of *e4:canceled* ∧ *e4:canceled* is a PARENT of *e5:affecting*
=> *e1:storm* is a PARENT of *e5:affecting*

Conjunction

e3:died is BEFORE *e4:canceled* ∧ *e4:canceled* is a PARENT of *e5:affecting*
=> *e3:died* BEFORE *e5:affecting*

Implication & *Negation*

e1:storm is PARENT of *e4:canceled* => *e1:storm* BEFORE *e4:canceled*
e3:died is BEFORE *e4:canceled* => ¬ *e3:died* is COREF of *e4:canceled*



The model needs to learn to give **globally consistent** prediction

Learning Logical Constraints

USC Viterbi School of Engineering

Extending the cross-entropy to capture logical constraints

- L_A Annotation: $\top \rightarrow r(e_1, e_2)$ $\boxed{\rightarrow}$ $-w_r \log r_{(e_1,e_2)}$
- L_S Implication Loss: $\alpha(e_1, e_2) \leftrightarrow \bar{\alpha}(e_2, e_1)$ $\boxed{\rightarrow}$ $|\log \alpha_{(e_1,e_2)} - \log \bar{\alpha}_{(e_2,e_1)}|$
- L_C Conjunction Loss: $\alpha(e_1, e_2) \wedge \beta(e_2, e_3) \rightarrow \gamma(e_1, e_3)$ $\boxed{\rightarrow}$ $\log \alpha_{(e_1,e_2)} + \log \beta_{(e_2,e_3)} - \log \gamma_{(e_1,e_3)}$

  $\alpha(e_1, e_2) \wedge \beta(e_2, e_3) \rightarrow \neg\delta(e_1, e_3)$ $\boxed{\rightarrow}$ $\log \alpha_{(e_1,e_2)} + \log \beta_{(e_2,e_3)} - \log(1 - \delta_{(e_1,e_3)})$
- Training Objective: $L = L_A + \lambda_S L_S + \lambda_C L_C$

α \ β	PC	CP	CR	NR	BF	AF	EQ	VG
PC	PC, ¬AF	–	PC, ¬AF	¬CP, ¬CR	BF , ¬CP, ¬CR	–	BF , ¬CP, ¬CR	–
CP	–	CP, ¬BF	CP, ¬BF	¬PC, ¬CR	–	AF, ¬PC, ¬CR	AF, ¬PC, ¬CR	–
CR	PC, ¬AF	CP, ¬BF	CR, EQ	NR	BF , ¬CP, ¬CR	AF, ¬PC, ¬CR	EQ	VG
NR	¬CP, ¬CR	¬PC, ¬CR	NR	–	–	–	–	–
BF	BF , ¬CP, ¬CR	–	BF , ¬CP, ¬CR	–	BF , ¬CP, ¬CR	–	BF , ¬CP, ¬CR	¬AF, ¬EQ
AF	–	AF, ¬PC, ¬CR	AF, ¬PC, ¬CR	–	–	AF, ¬PC, ¬CR	AF, ¬PC, ¬CR	¬BF , ¬EQ
EQ	¬AF	¬BF	EQ	–	BF , ¬CP, ¬CR	AF, ¬PC, ¬CR	EQ	VG, ¬CR
VG	–	–	VG, ¬CR	–	¬AF, ¬EQ	¬BF , ¬EQ	VG	–

Table 1: The induction table for conjunctive constraints on temporal and subevent relations. Given the relations $\alpha(e_1, e_2)$ in the left-most column and $\beta(e_2, e_3)$ in the top row, each entry in the table includes all the relations and negations that can be deduced from their conjunction for e_1 and e_3, i.e. $\mathrm{De}(\alpha, \beta)$. The abbreviations PC, CP, CR, NR, BF, AF, EQ and VG denote PARENT-CHILD, CHILD-PARENT, COREF, NOREL, BEFORE, AFTER, EQUAL and VAGUE, respectively. Vertical relations are in black, and TempRel are in blue. "–" denotes no constraints.

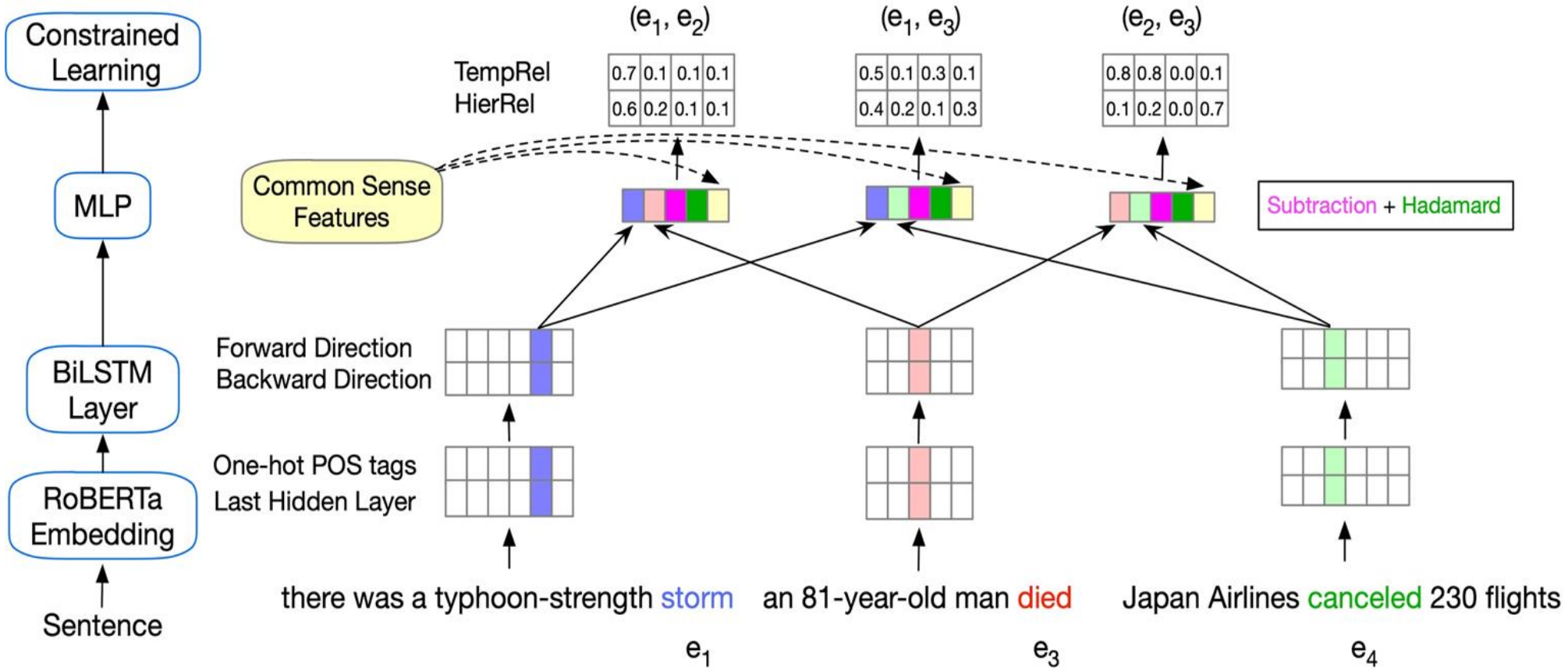
Learning Logical Constraints

- Joint learning on every three mentions

Relation annotations

Logical constraints
- Symmetry, transitivity, conjunction, implication. negation.

Loss Function: $L = L_A + \lambda_S L_S + \lambda_C L_C$



Wang, et al. "Joint Constrained Learning for Event-Event Relation Extraction." EMNLP, 2020

How Good is Constrained Learning

USC Viterbi School of Engineering

Constrained learning surpasses SOTA TempRel extraction on MATRES [Ning+, ACL-18] by relatively 3.27% in F_1.

Model	P	R	F_1
CogCompTime (Ning et al., 2018c)	0.616	0.725	0.666
Perceptron (Ning et al., 2018b)	0.660	0.723	0.690
BiLSTM+MAP (Han et al., 2019b)	-	-	0.755
LSTM+CSE+ILP (Ning et al., 2019)	0.713	0.821	0.763
Joint Constrained Learning (ours)	**0.734**	**0.850**	**0.788**

On HiEve [Glavaš+, LREC-14] for subevent extraction, it relatively surpasses previous methods by at least 3.12% in F_1.

Model	F_1 score		
	PC	CP	Avg.
StructLR (Glavaš et al., 2014)	0.522	**0.634**	0.577
TACOLM (Zhou et al., 2020a)	0.485	0.494	0.489
Joint Constrained Learning (ours)	**0.625**	0.564	**0.595**

Key Observations

- Constraints are a natural bridge for learning resources with different sets of relations
- Adding constraints in learning is sufficient to enforce logical consistency of outputs, surpassing ILP in inference (w/ constrained learning) by 2.6-12.3% in ACC

More Challenges of RE

- Encoding very long documents
  - Beltagy, Iz, Matthew E. Peters, and Arman Cohan. "Longformer: The long-document transformer." arXiv preprint arXiv:2004.05150 (2020).
- Automatically discovering constraints from data
  - Pan, Xingyuan, Maitrey Mehta, and Vivek Srikumar. "Learning Constraints for Structured Prediction Using Rectifier Networks." *ACL,*. 2020.
- Mitigating label noise
  - Zhou, Wenxuan, and Muhao Chen. "Learning from Noisy Labels for Entity-Centric Information Extraction." *EMNLP*, 2021

Coreference Resolution

- Multiple entities and pronouns in a document: Which corresponds to which

Christopher Robin is alive and well. **He** is the same person that you read about in the book, **Winnie the Pooh**. As a **boy**, **Chris** lived in a pretty home called **Cotchfield Farm**. When **Chris** was three years old, **his father** wrote a poem about **him**. The poem was printed in a magazine for others to read. **Mr. Robin** then wrote a book

Coreference Resolution

Learn a pairwise similarity measure (local predictor)

A 3-step process

- Identifying mention spans
- Learning the similarity function
- Clustering mentions

Kind of like **binary RE + clustering**

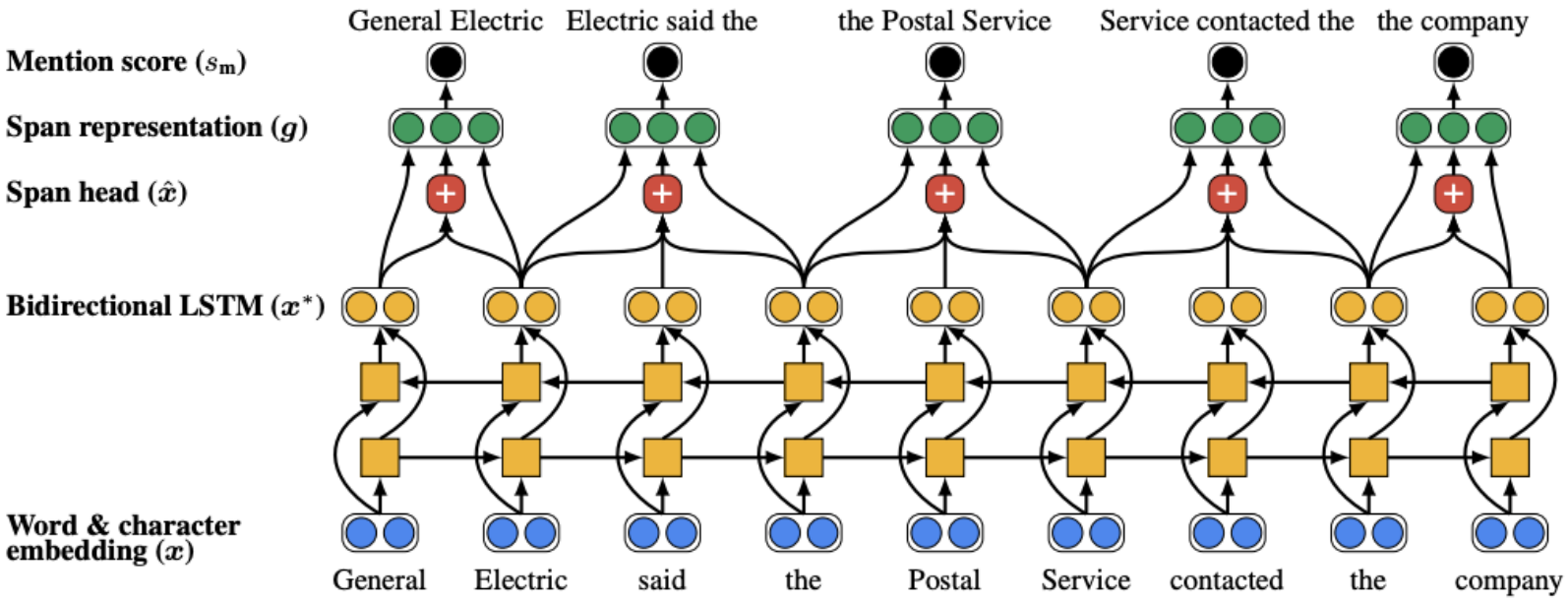
Christopher Robin is alive and well. **He** is the same person that you read about in the book, **Winnie the Pooh**. As a **boy**, **Chris** lived in a pretty home called **Cotchfield Farm**. When **Chris** was three years old, **his father** wrote a poem about **him**. The poem was printed in a magazine for others to read. **Mr. Robin** then wrote a book

Mention Detection

- Earlier pipeline approach – build a parsing-based mention detector
  - Include pronouns, name entities, common nouns (using information from parse tree)
- End-to-End Approach
  - Learning jointly with coreference models

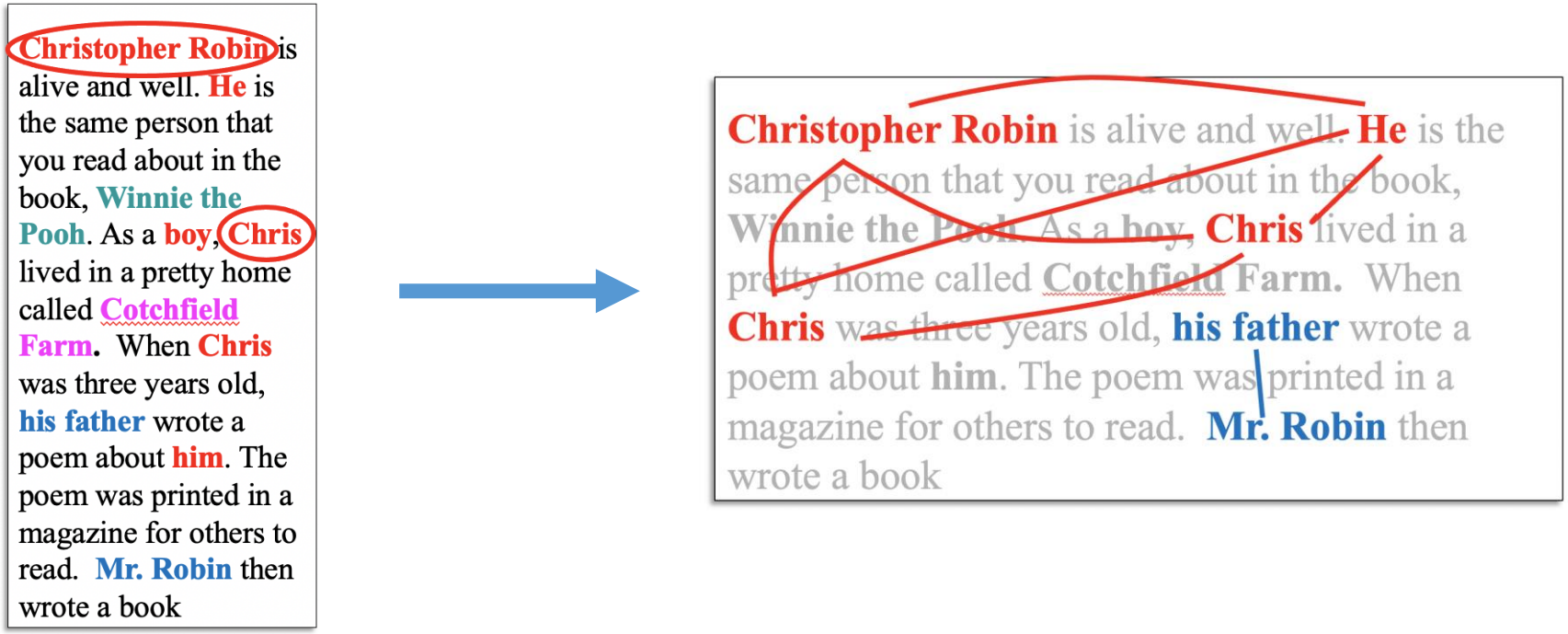
Mention Detection

- Bi-LSTM mention detector



Pairwise Coreference Resolution

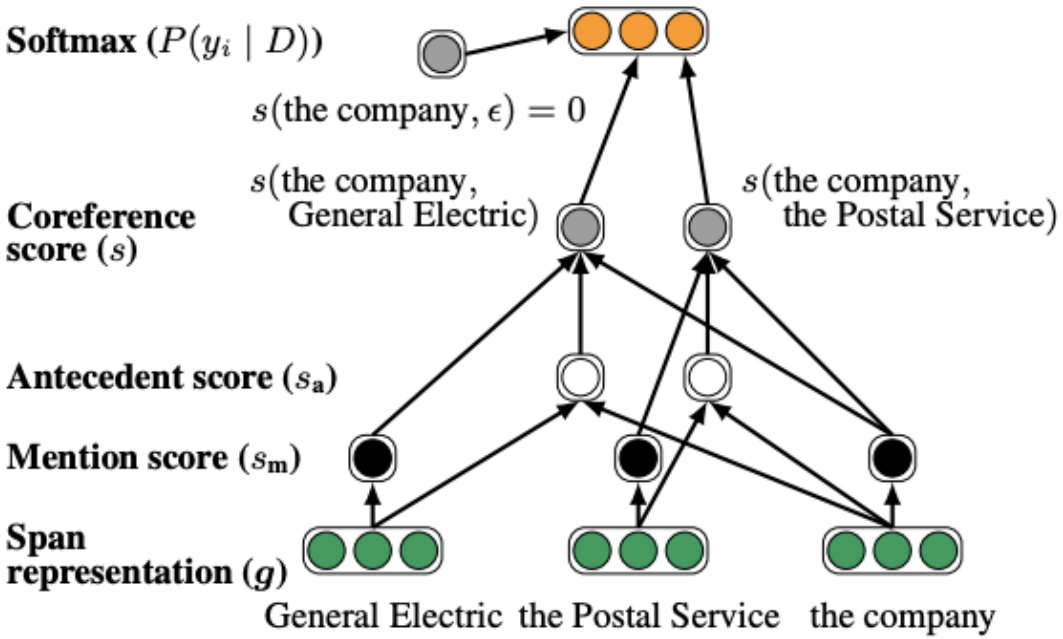
1. Compute similarity between token spans
2. Generate coreference clusters



Pairwise Coreference Resolution

How to learn the similarity function?

- Feature-based approach, learn a statistical classifier with features
  - same sub-string? positions in the paragraph?
- End-to-End approach (as binary relation extraction)



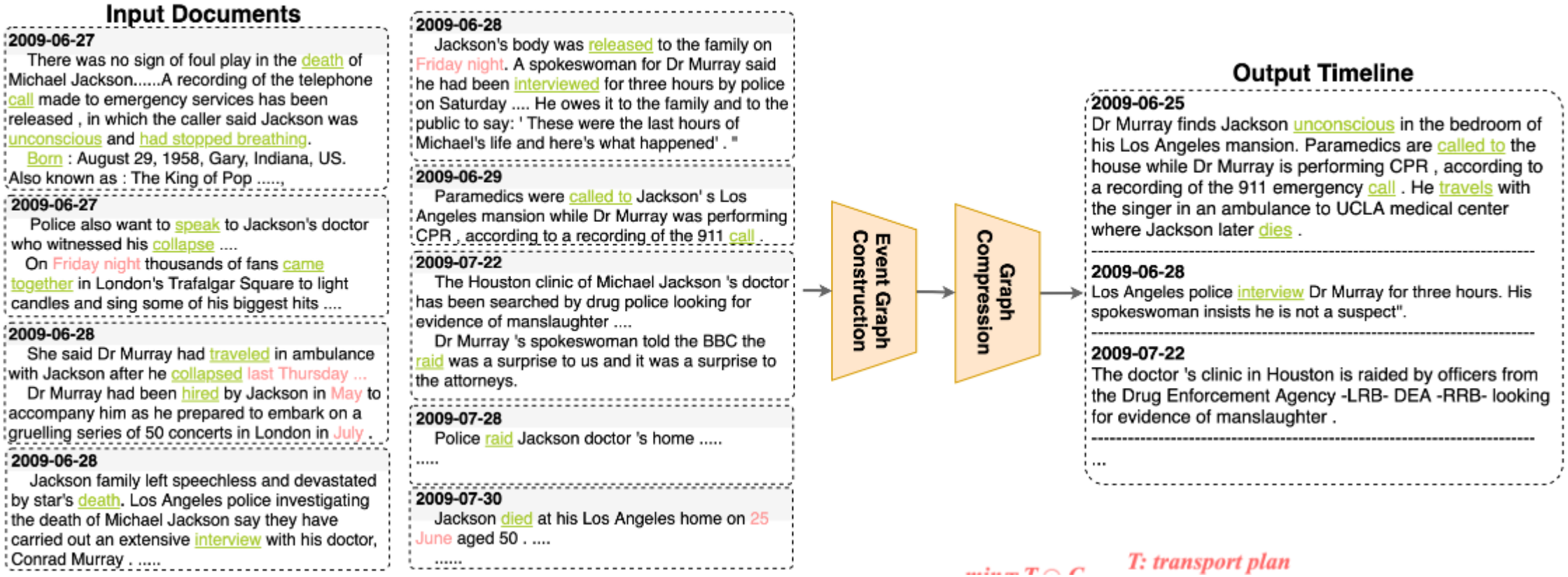
Outline

- Background and Motivation
- Lexical IE
  - Named Entity Recognition
  - Typing
  - Linking
  - Event Extraction
- Relational IE
  - Relation Extraction
  - Coreference Resolution
- Research Directions

Cross-document IE

Relevant concepts and facts may be extracted from multiple documents.

- Long-form processing
- Insufficient supervision



Indirectly Supervised IE

IE models can be lack of supervision signals due to expensive annotations.

- One solution: using indirect supervision from other tasks

Original Passage
In addition , *many people* were poisoned when **toxic gas** was released. *They* were poisoned and did not know how to protect *themselves* against **the poison**.
Our formulation
Q1: Who were poisoned when toxic gas was released?
A1: [*They*, *themselves*]
Q2: What was released when many people were poisoned?
A2: [**the poison**]
Q3: Who were poisoned and did not know how to protect themselves against the poison?
A3: [*many people*, *themselves*]
Q4: Whom did they not know how to protect against the poison?
A4: [*many people*, *They*]
Q5: They were poisoned and did not know how to protect themselves against what?
A5: [**toxic gas**]

QA for coreference resolution.

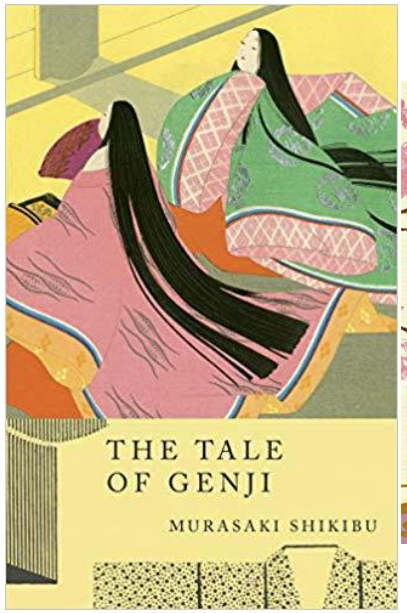
Wu, et al. "CorefQA: Coreference resolution as query-based span prediction." *ACL* 2020.
Li et al. "Ultra-fine Entity Typing with Indirect Supervision from natural Language Inference. *TACL*, 2022

We also talked about **NLI for Entity Typing** today

Other directions:
- QA for IE
- NLG (incl. Summarization) for IE
- IR for IE

Trustworthiness of IE

- What if extractions from different docs are inconsistent with each other?



(The Tale of Genji, *Genre*, **?e**)

DBpedia

Novel

ヂビペディア

Monogatari (story)
Love story
Royal family story
Realistic novel
Ancient literature

Consolidation: Are they complementary or contradictory to each other?
How to verify each extracted fact? (open-domain fact verification)

Out-of-Distribution Detection in IE

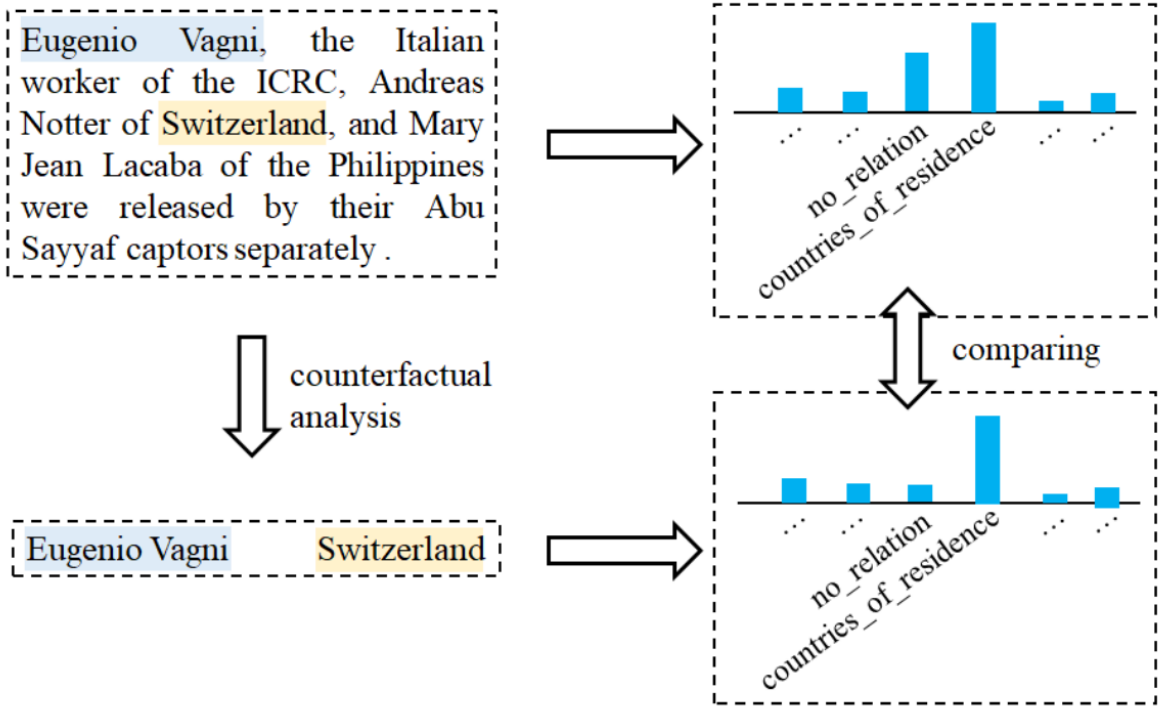
IE system should know to extract nothing

- If the input is out-of-distribution (i.e. no-relation or unknown relation)
- Also related to *learning/inference with abstention*

Zhou, et al. "Contrastive Out-of-Distribution Detection for Pretrained Transformers." *EMNLP* 2021

Spurious correlation in IE

- Relation extraction: Is the system really extracting the relation from the context, or just guessing the relation based on entity names/types?



Wang et al. Should We Rely on Entity Mentions for Relation Extraction? Debiasing Relation Extraction with Counterfactual Analysis. NAACL 2022

Takeaways

- Lexical IE:
  - Named entity recognition: a sequence tagging problem
  - Entity typing: multi-class classification on contextualized entity representation
    - SOTA solution: Can be reformulated to NLI like other discriminative tasks
  - Entity linking: addresses based on contextual similarity of entities
- Relational IE:
  - Relation extraction: classification based on contextual representation of entity pairs
  - Coreference resolution: mention detection + mention clustering
- Common challenges to current IE systems:
  - Lack of direct supervision (annotations are expensive)
  - Robustness (noisy training, unbiased training, spurious correlation).
  - Generalizability (domain-shift, new extraction types)

Thank You