Deep Learning for NLP

Jay Urbain, PhD

Professor, Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

urbain@msoe.edu

TOPICS: Classification

- What is Natural Language Processing
- Why is NLP hard?
- NLP Vocabulary
- Text Classification
  - Word embeddings
  - Network architectures: MLP, CNN, RNN
  - TensorFlow, Keras, Pandas

WHAT IS NATURAL LANGUAGE PROCESSING (NLP)?

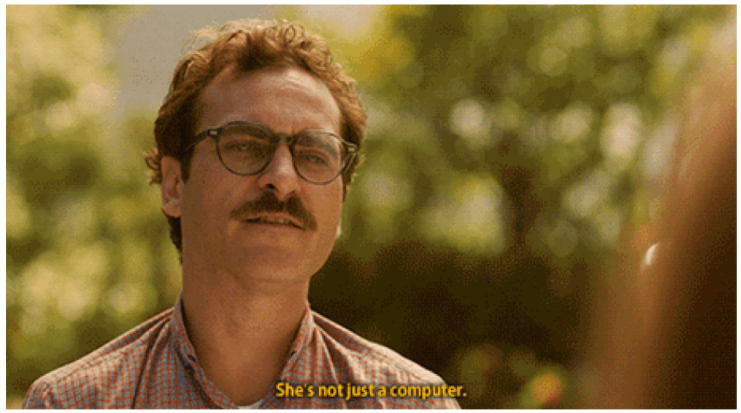
WHAT IS A NATURAL LANGUAGE PROCESSING (NLP)?

"Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. "

Wikipedia

GOAL: for computers to process or "*understand*" natural language in order to perform tasks that are useful such as question answering

- Add structure to unstructured text
- Full understanding for now is unsolved



What is Natural Language Processing?

"Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. "

Wikipedia

NLP APPLICATIONS

- Keyword search
- Text classification
- Named entity recognition
- Machine translation
- Question answering, chatbot dialog systems
- Much more...

https://www.google.com/calendar/render?tab...

CLINICAL & TRANSLATIONAL RESEARCH INFORMATICS

Patient Deidentification

https://cis.ctsi.mcw.edu/deid/

2019-04-04 13:04:03.931

Date offset:

10

Patient name:

Input text:

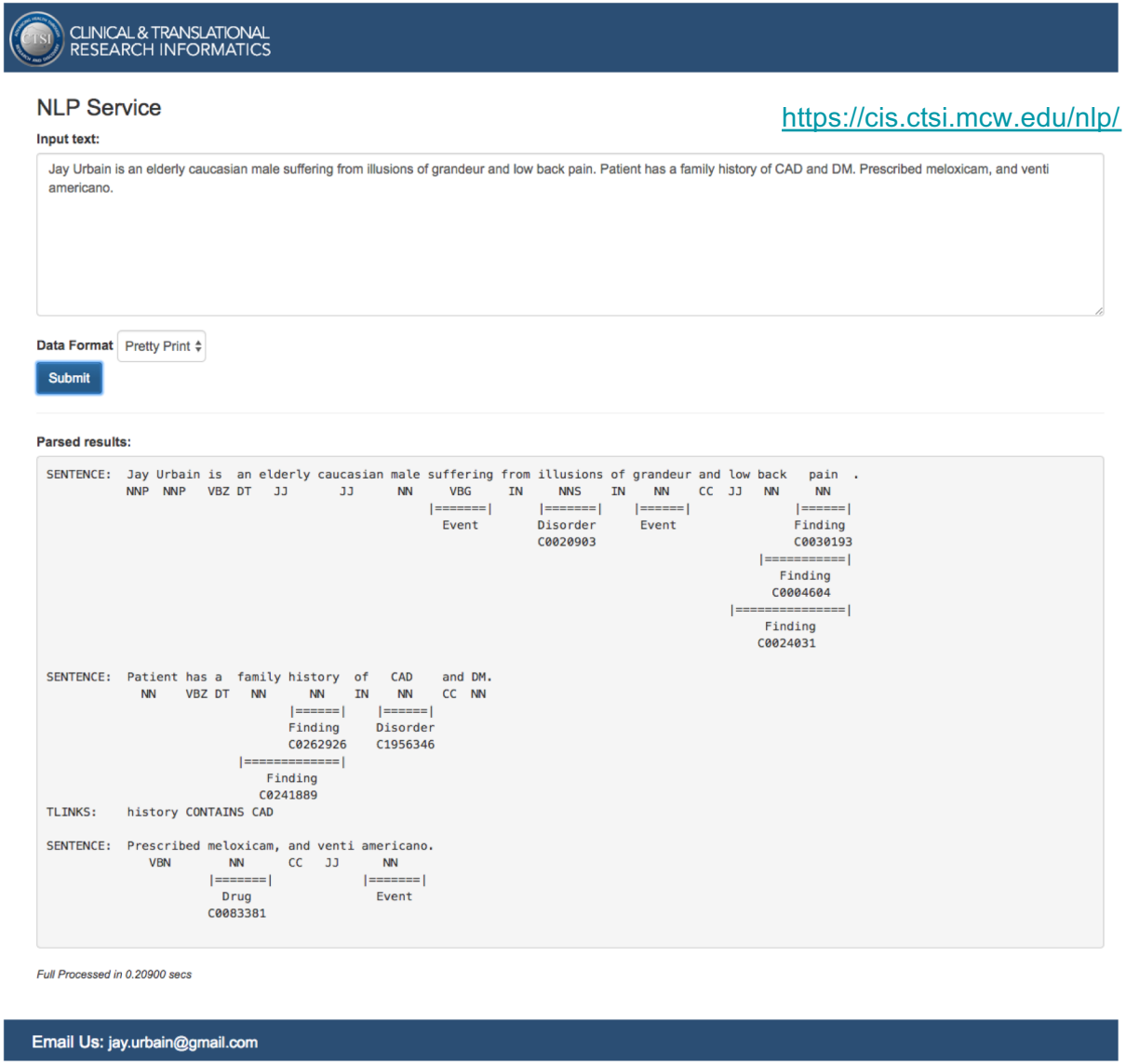
Jay Urbain, jay.urbain@gmail.com, born December 6, 2156 is an elderly caucasian male suffering from illusions of grandeur and LBP. He is married to Kimberly Urbain, who is much better looking. Patient father, Francis Urbain has a history of CAD and DM. Jay has been prescribed meloxicam, and venti americano. He lives at 9050 N. Tennyson Dr., Disturbia, WI with his wife and golden retriever Mel. You can reach him at 414-745-5102.

Data Format Pretty Print

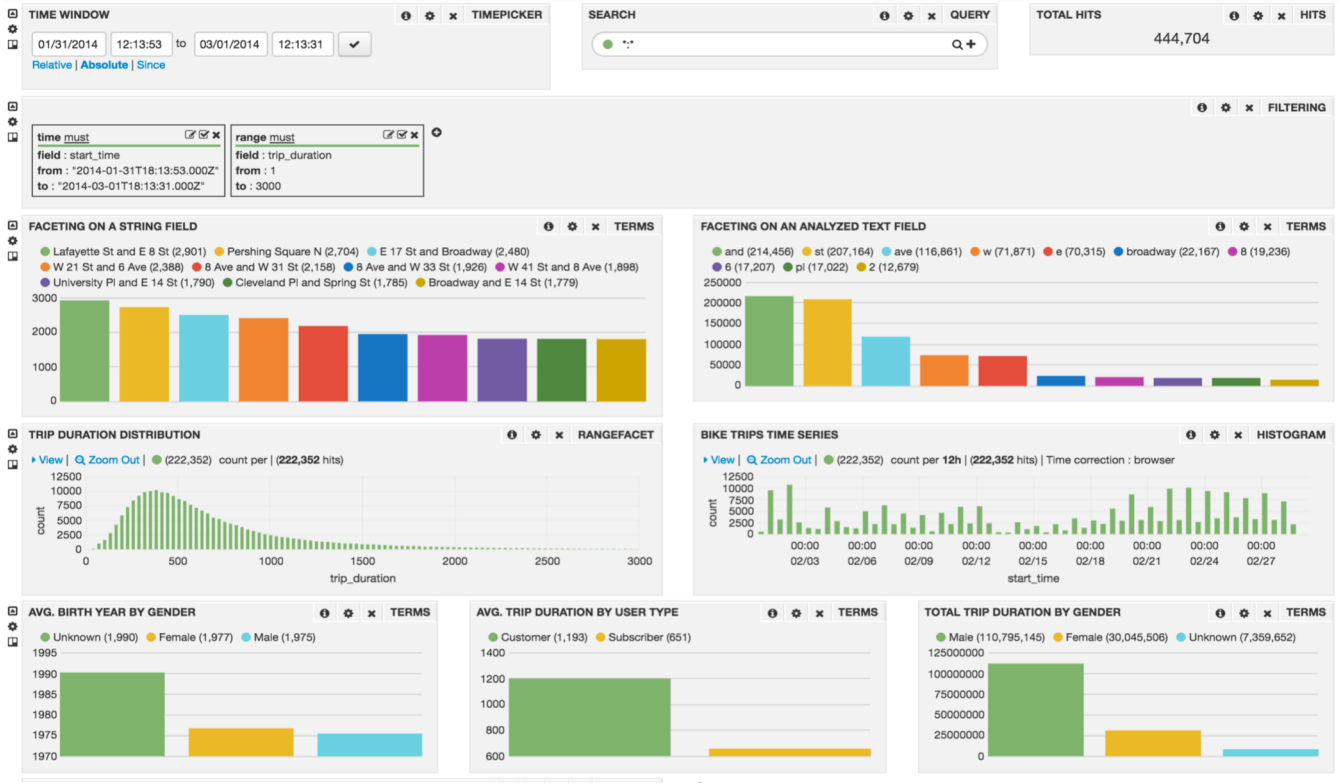
Submit

Parsed results:

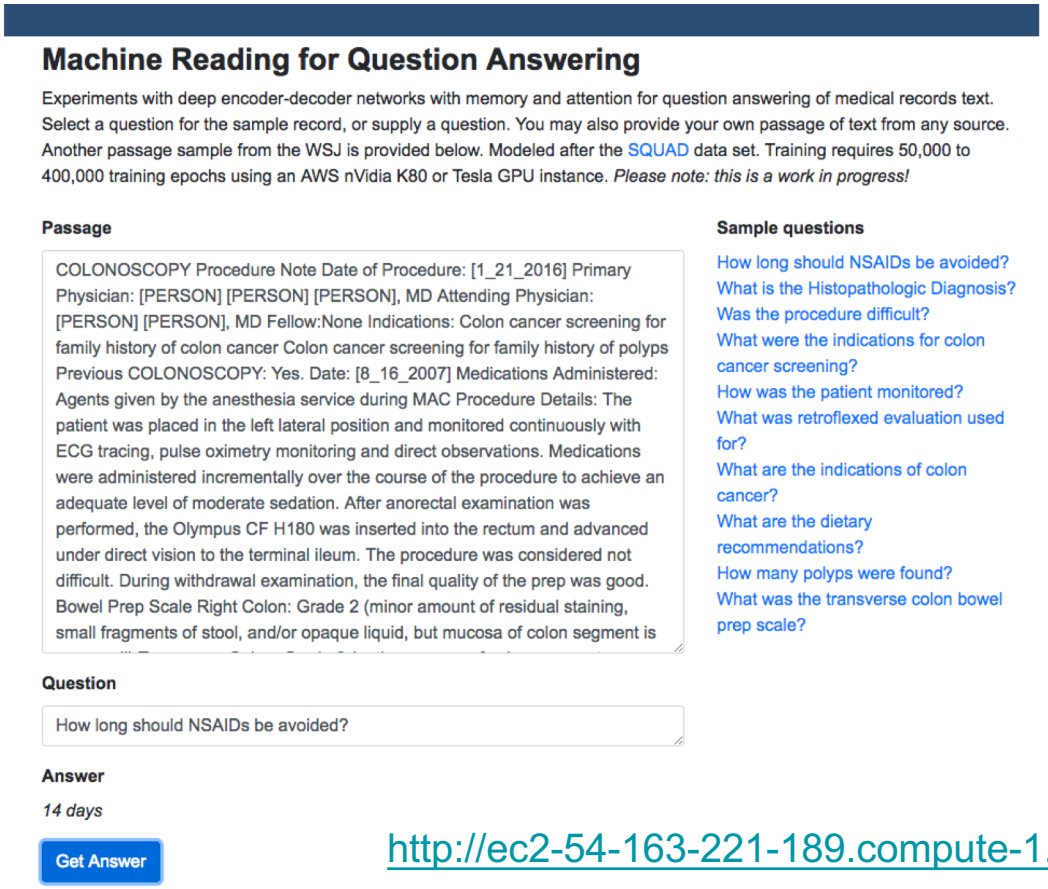
Clinical Named Entity Identification



Search driven discovery: aggregating statistics from named entities



Question Answering



http://ec2-54-163-221-189.compute-1.amazonaws.com:8080/

WHY IS NLP HARD?
… and why Deep Learning may help

Language: the ultimate UI

Where is A Bug's Life playing in Milwaukee?

A Bug's Life is playing at the Downer Theater.

When is it playing there?

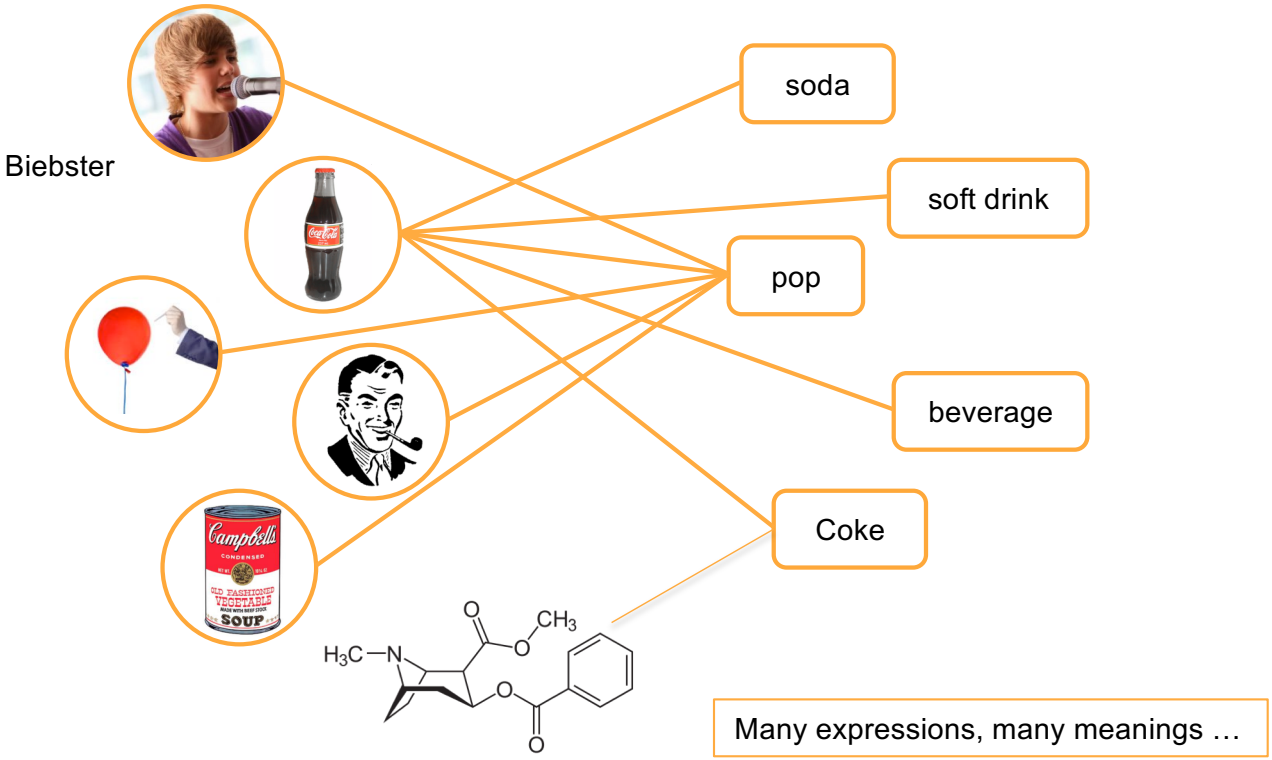
It's playing at 2pm, 5pm, and 8pm.

OK. I'd like 1 adult and 2 children for the first show.
How much would that cost?

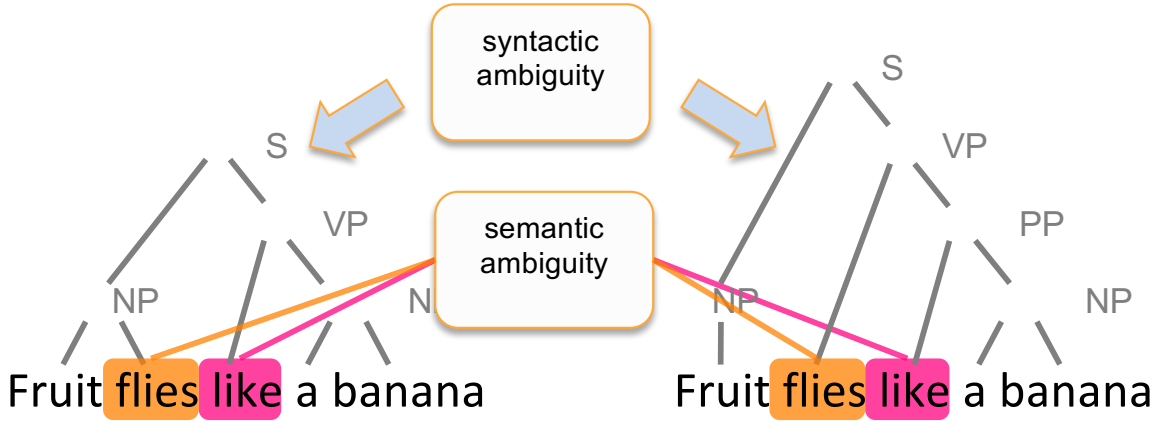
But we need domain knowledge, discourse knowledge, world knowledge

Meanings and expressions

Big obstacle: relation between meanings and expressions is not one-to-one



Syntactic & semantic ambiguity

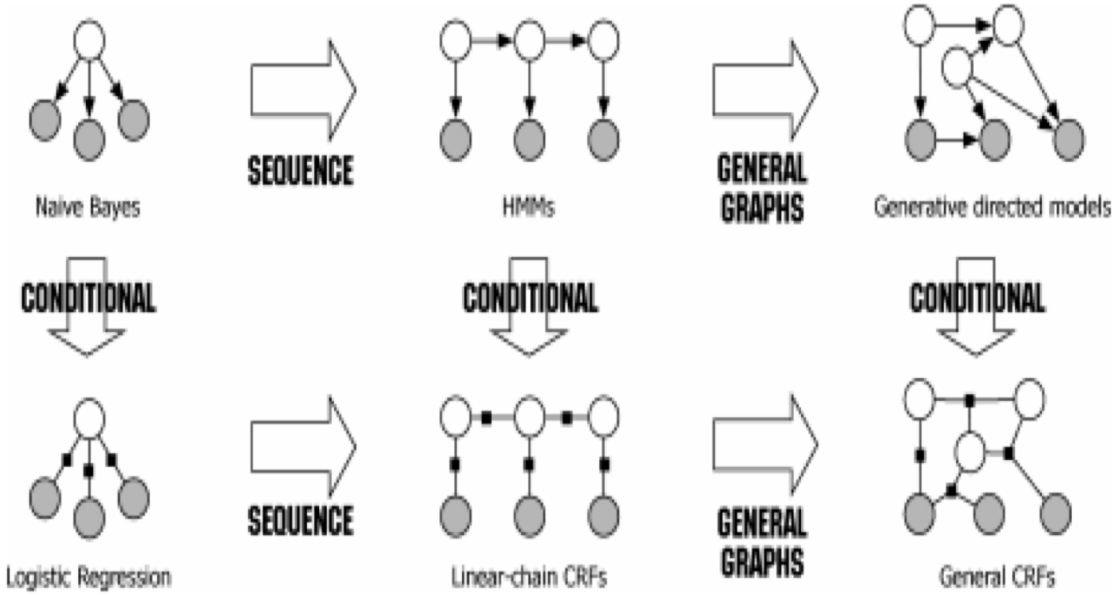


["like" is a remarkable word: it can be used as a noun, verb, adverb, adjective, preposition, particle, conjunction, or interjection.]

photos from worth1000.com

Traditional NLP use “shallow” statistical machine learning models: HMM, MEMM, CRF

- http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf



Traditional Models: HMM, MEMM, CRF

- http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf
- Linear chain CRF for sequence class, Named entity CRF features

Definition 2.2. Let Y, X be random vectors, $\theta = \{\theta_k\} \in \Re^K$ be a parameter vector, and $\mathcal{F} = \{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$ be a set of real-valued feature functions. Then a *linear-chain conditional random field* is a distribution $p(\mathbf{y}|\mathbf{x})$ that takes the form:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}, \qquad (2.18)$$

where $Z(\mathbf{x})$ is an input-dependent normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}. \qquad (2.19)$$

W=v	$w_t = v$	$\forall v \in \mathcal{V}$
T=j	part-of-speech tag for w_t is j (as determined by an automatic tagger)	$\forall$POS tags j
P=I–j	w_t is part of a phrase with syntactic type j (as determined by an automatic chunker)	
Capitalized	w_t matches `[A-Z][a-z]+`	
Allcaps	w_t matches `[A-Z][A-Z]+`	
EndsInDot	w_t matches `[^\.]+.*\.`	
	w_t contains a dash	
	w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`	
Acro	w_t matches `[A-Z][A-Z\\.]*\\.[A-Z\\.]*`	
Stopword	w_t appears in a hand-built list of stop words	
CountryCapital	w_t appears in list of capitals of countries	
⋮	many other lexicons and regular expressions	
$q_k(\mathbf{x}, t+\delta)$ for all k and $\delta \in [-1, 1]$		

Custom solutions for each domain, and for each application.
Difficulty capturing long-range dependencies, context, and semantics

NLP: With deep Learning, reality is getting closer to vision

NLP VOCABULARY

VOCABULARY

- *token* - A unit of text, typically a word. Could also be:
  - Group of words like "New York"
  - Sub-word like "mega" in "megabyte"
  - Letter like "m"
- *tokenizing* - Creating tokens, representing tokens as a distinct number
- *document* - Sequence of *tokens*
  - Could be whole book or a tweet
  - Classify each *document* as having a positive or a negative sentiment
- *corpus* - A set of *documents*
  - We'll use a *corpus* containing movie reviews since each *document* is paired with a rating indicating *sentiment*

VOCABULARY

- *n-grams* - adjoining sequence of *n* tokens (words)
  - Sometimes referred to as shingles
- *sentiment analysis* - evaluate mood of n-grams
  - Determine the attitude / emotion of text
- *tf-idf* = term frequency * inverse document frequency
  - Numerical statistic intended to reflect significance of a word to a document in a corpus
  - Popular term-weighting schemes
  - TF increases each time a word appears in a document
  - IDF decreases each time a word appears in a corpus

TEXT CLASSIFICATION

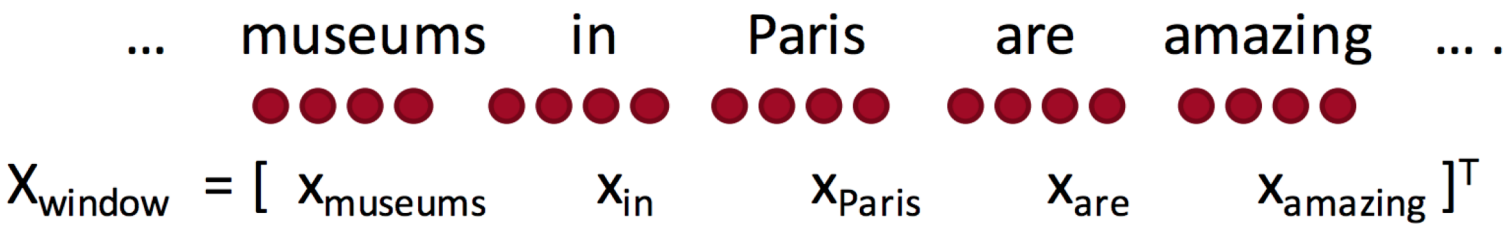
TEXT CLASSIFICATION

- We will be training several neural network models for text classification.
- In text classification, documents (or segments of text) are assigned to different *categories*.
- For example, categories could be positive or negative sentiment, different authors, different topics, or different writing styles.

- Interesting problems like ambiguity arise:
  - "To sanction" can mean "to permit" or "to punish."
  - Berkshire Hathaway vs. Anne Hathaway
- Capturing context at different hierarchical levels is key. Deep Learning can help us!

WINDOW TEXT CLASSIFICATION

Classify **Paris** in the context of this sentence with window of length 2:



EMBEDDINGS

WORKING WITH TEXT

- Neural networks don't take raw text as input. Only numeric tensors.
- *Vectorizing* text is the process of transforming text into numeric tensors. This can be done in multiple ways:
  - **Segment text into words, and transform each word into a vector.**
  - Segment text into characters, and transform each character into a vector.
  - Extract n-grams of words or characters, and transform each n-gram into a vector.
- Apply categorical transformation:
  - One-hot encoding
  - Word embedding

ONE-HOT ENCODING

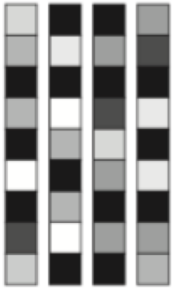
- Words are categorical, they have no *ordinal* relationship. So they can't just be numerically encoded.
- For each word:
  - Numerically encode each word: {red:0, green:1, blue:2}
  - Generate vector of zeros the length of all possible words/categories
  - Set the index value for the word in the vector

red,	green,	blue
1,	0,	0
0,	1,	0
0,	0,	1

WORD EMBEDDINGS

- One-hot results in *sparse vector representation* (mostly zeros, hi dimensions).
- Word embeddings provide *dense vector representations*.
- Idea is to “embed” words into a lower-dimensionality space.
- The dimensions of this space are typically defined by word context, i.e., semantically similar words are embedded near each other.
- Popular algorithms:
  - Point-wise mutual information (PMI),Word2Vec: Skip-gram or CBOW, GLoVE, FastTect

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

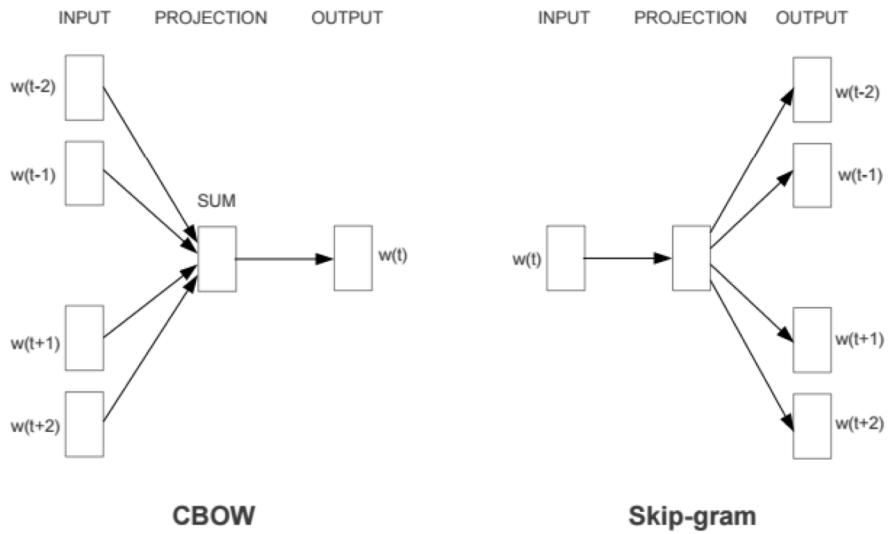
The Distributional Hypothesis (Firth, 1957)

- ‘You can tell a word by the company it keeps’

  - *The cat sat on the mat*
  - *The dog sat on the mat*
  - *The elephant sat on the mat*
  - *The quickly sat on the mat*

Embeddings: Word2Vec

- **CBOW** model predict missing word (focus word) using context (surrounding words).
- **Skip gram** model predicts context based on the word in focus.
- Context is a fixed number of words to the left and right of the word in focus.
- Maximize average log probability of context words co-occurring with focus words.



Properties of embeddings

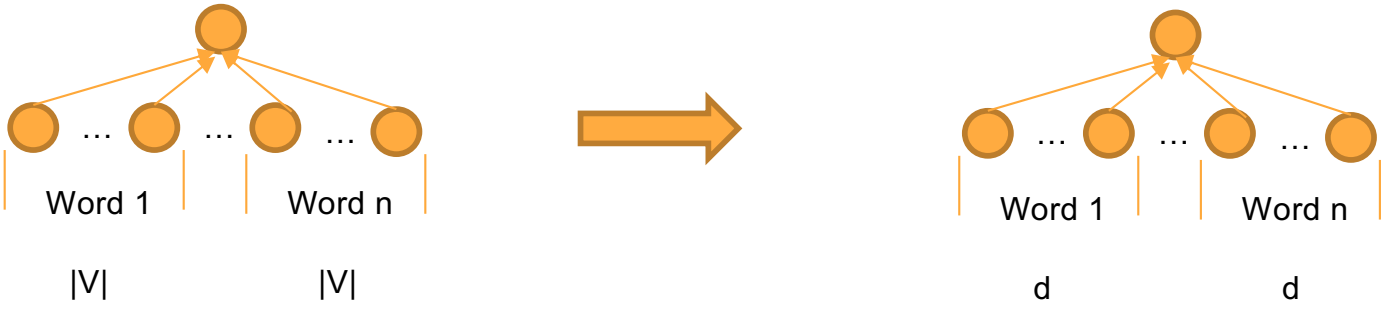
- They are dense: 0.53, 0.2, -1.2, ….
- They are low dimensional: (typically 50 <= d <= 300)
- Embed domain semantics:

  `king' - `man` + `woman` ≈ `queen`

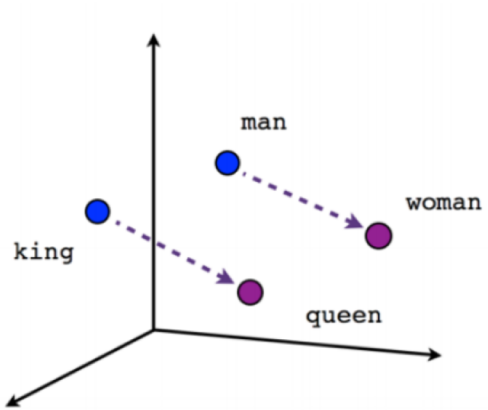
  `paris` - `france` + `spain` ≈ `madrid`
- Generalize easily!

Properties of embeddings (cont.)

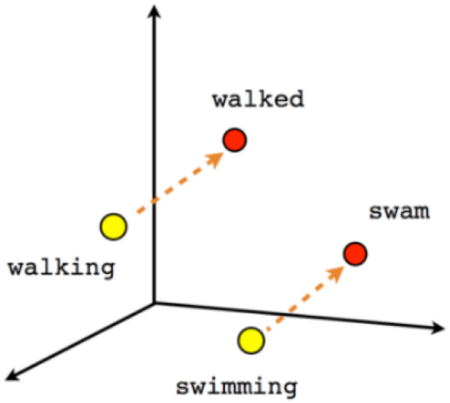
- They are used as input across a variety NLP tasks (transfer learning!):



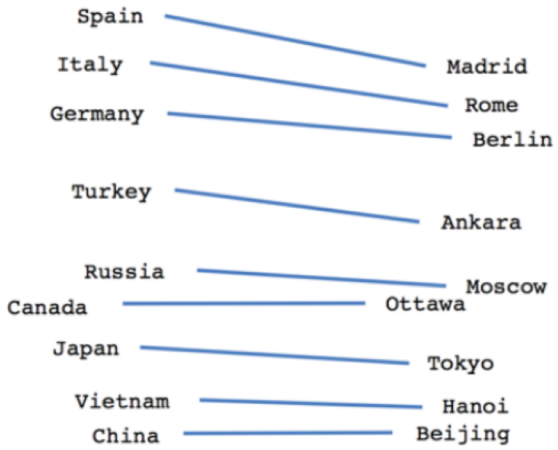
Total # of input parameters:	n * |V|	>>	n * d



Male-Female

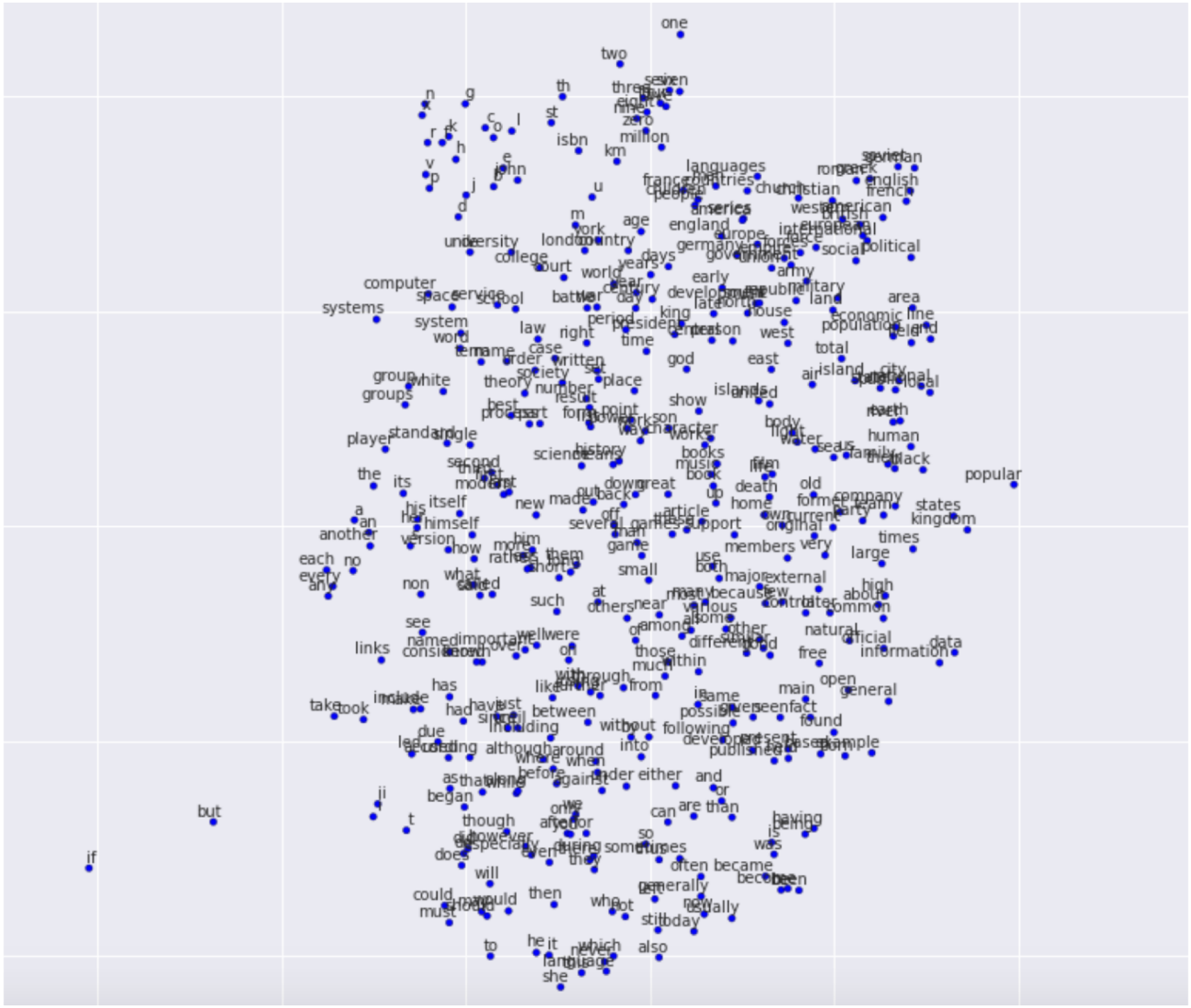


Verb tense



Country-Capital

https://www.tensorflow.org/tutorials/representation/word2vec


one
two
isbn
million
computer
systems
but
if
popular

USING WORD EMBEDDINGS

1. Learn an Embedding
   - **Use KERAS embedding layer**

2. Reuse an Embedding
   - Download pre-trained word vectors

TENSORFLOW, KERAS, PANDAS

WHAT IS TENSORFLOW?

Created by Google, tensorflow.org

- "Open source software library for machine intelligence"
  - Available on GitHub
- Flexibility—express your computation as a data flow graph
  - If you can express it in TF syntax you can run it
- Portability—CPUs and GPUs, workstation, server, mobile
- Language options—Python, C++, R
- Performance—Tuned for performance on CPUs and GPUs

KERAS

- Developed by Francois Chollet, Google Research Engineer
- Modular neural network written in Python
- Runs on TensorFlow, CNTK, and Theano (no longer supported)
- Keras library allows for easy and fast prototyping
- Runs on GPUs and CPUs
- Now packaged with latest TensorFlow

Keras

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()

# Stacking layers by using .add():
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

More here:

https://keras.io/

PANDAS

- Open-source, BSD-licensed project
- Fast and efficient DataFrame object for data manipulation with integrated indexing
  - DataFrames used in this lab
  - Ideal for neural networks
- Contains tools for reading and writing data between in-memory data structures and different formats such as:
  - CSV and text files – used in this lab
  - Microsoft Excel
  - SQL databases
  - HDF5

NETWORK ARCHITECTURE - MLP

MULTI-LAYER PERCEPTRON

- MLP = Multi-layer Perceptron
  - Traditional *fully connected feed-forward network*
  - Also called *dense network*
  - Fixed size input

LAB DISCUSSION / OVERVIEW

LAB OVERVIEW

- Perform text classification for sentiment analysis
  - Classify each document into one of two different sentiments:
    - "positive“ or "negative“
- Components of lab include:
  - Pandas
  - TensorFlow and Keras
  - MLP, CNN, RNN/LSTM

LAB PROCESS

Data Preparation:

1. Load libraries, data and visualize / evaluate data
2. Divide dataset between Training, Validation and Test sets
3. Implement TF-IDF using Keras
4. Create matrix for labels

LAB PROCESS

Build and evaluate 3 network architectures:

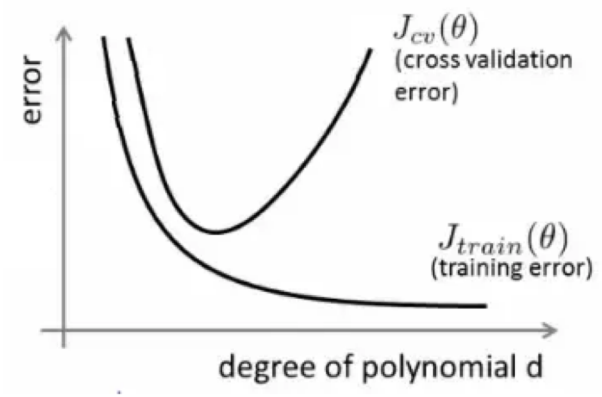
1. Design simple neural network
2. Visualize network architecture
3. Train model
4. Compare test, validation and training accuracy
5. Attempt to improve the model

- **Complete the Simple Multi-Layer Perceptron Model portion of the lab.**
- **Stop when you reach One-Dimensional Convolutional Neural Network Model, we'll regroup.**

TRAINING TERMINOLOGY

Overfitting and Underfitting

- Overfitting refers to a model that fits the training data too well.
- Your model begins to “memorize” the training data and becomes less likely to generalize to unseen test data.
- Validation error >> training error
- Correct by either simplifying model, generalizing model, or getting additional data.
- Underfitting refers to large training error. Increase complexity of model.

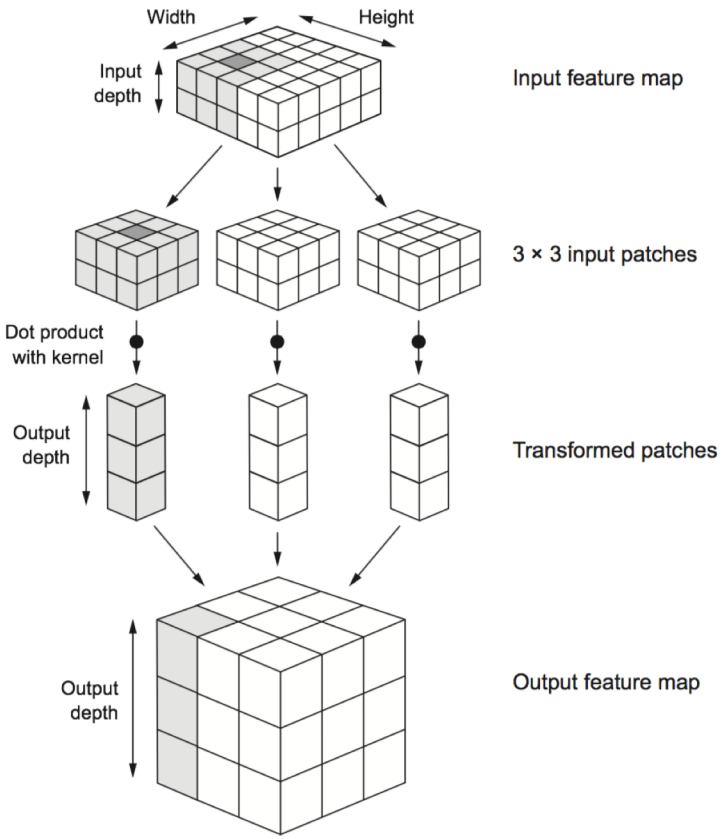


Learning terminology

- Learning rate -- weighting of dE/dW
- Decay rate – rate to reduce learning rate
- Decay steps: number of steps to execute before changing learning rate
- Number of epochs – number of times to cycle through the input data
- Batch size: number of samples that going to be propagated through the network

NETWORK ARCHITECTURE – CNNs

Convolution Neural Network (CNN)



FRANÇOIS CHOLLET, 2018

- A CNN is made up of Layers: convolution, pooling, fully connected.
- Each layer transforms an input 3D volume to an output 3D volume.
- Learns local parameters in input space using a stack of (e.g., 3x3) convolution filters
- The patterns the filters learn are translation invariant.
- Learn spatial hierarchies of patterns by adding conv. and pooling layers. E.g., edges, shapes, motifs.

LAB PROCESS

Build and evaluate the One-Dimensional Convolutional Neural network architectures

1. Design simple neural network
2. Visual network architecture
3. Train model
4. Compare test, validation and training accuracy
5. Attempt to improve the model

- **If you finish early, complete the RNN model.**

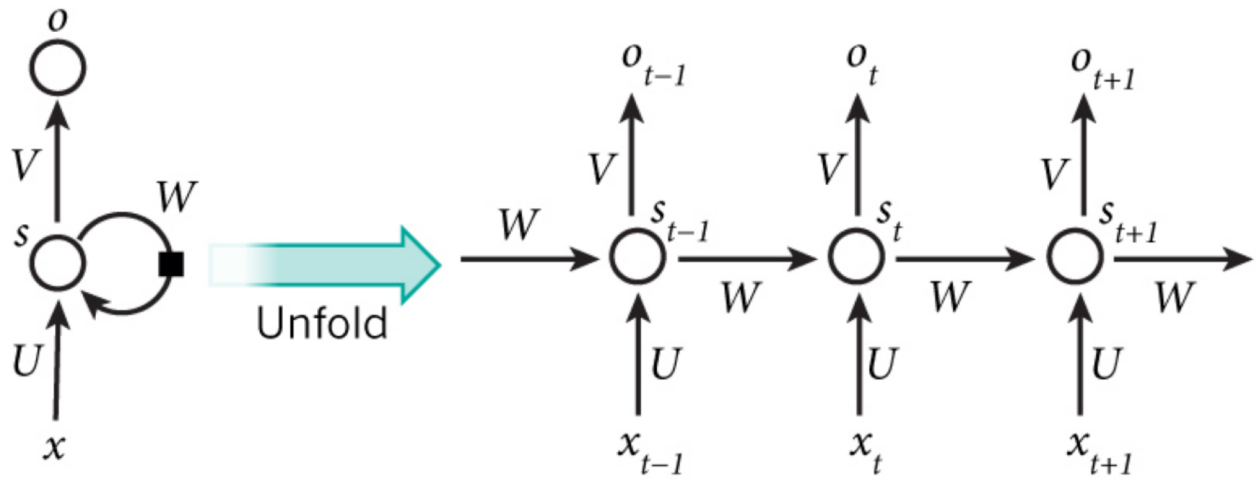
NETWORK ARCHITECTURE – RNNS / LSTMS

RECURRENT NEURAL NETWORK

- RNN = Recurrent Neural Network
  - Similar to traditional feed-forward network
  - RNNs include previous output state
  - Limited to looking back only a few steps due to vanishing gradient
    - Errors are backpropagated through time
    - Inputs from previous time steps get exponentially down weighted and are eventually driven to zero

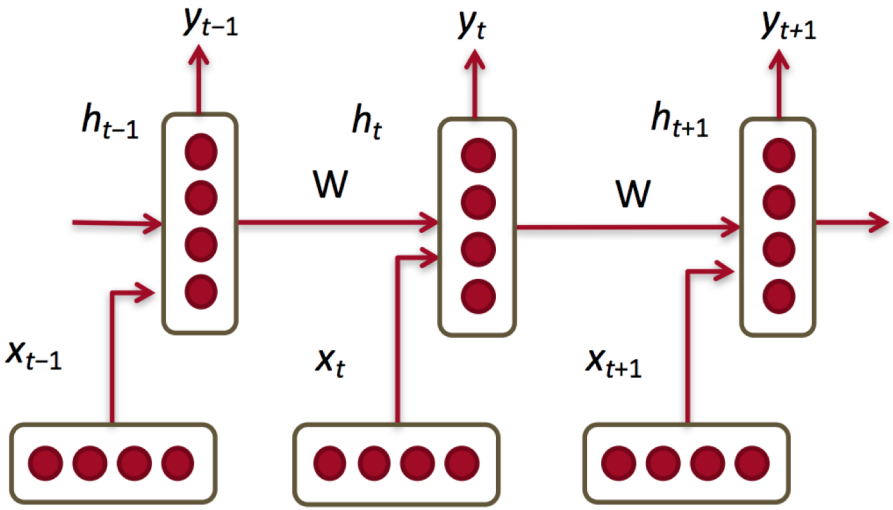
Recurrent Neural Network (RNN)

- RNNs have connections between units along a sequence.
- RNN's use the hidden state from the last time step and the input at the current step to make predictions.
- Allow RNN's can capture dynamic temporal behavior for a time sequence.



RECURRENT LAYERS

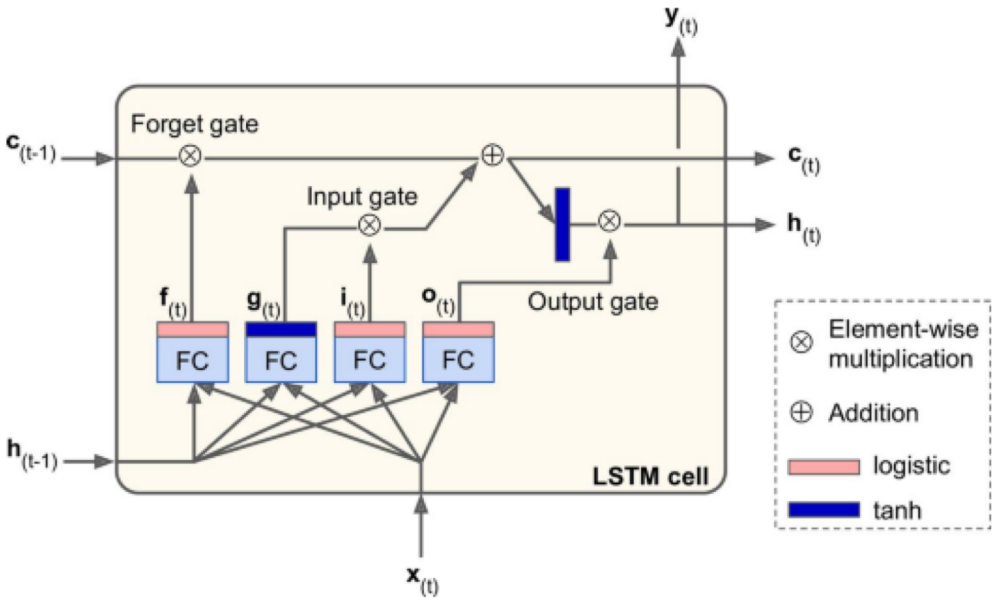
- RNNs share the weights between the time steps
- Condition the neural network on all previous words
  - (and all document in the end)
- RAM requirement only scales with number of timesteps



LONG SHORT TERM MEMORY

- LSTM = Long Short Term Memory
  - Variant of RNN
  - No (less) of a vanishing gradient problem
  - LSTMs can learn “very deep” tasks that require memories of words N words ago

Long Short Term (LSTM) Cell



$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
\mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
\end{aligned}
$$

LAB DISCUSSION / OVERVIEW

LAB PROCESS

Build and evaluate the LSTM network architecture

1. Design simple neural network
2. Visual network architecture
3. Train model
4. Compare test, validation and training accuracy
5. Attempt to improve the model

This can be a take-home exercise.

References

Neural Responding Machine for Short-Text Conversation (2015-03)

A Neural Conversational Model (2015-06)

A Neural Network Approach to Context-Sensitive Generation of Conversational Responses (2015-06)

The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems (2015-06)

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models (2015-07)

A Diversity-Promoting Objective Function for Neural Conversation Models (2015-10)

Attention with Intention for a Neural Network Conversation Model (2015-10)

Improved Deep Learning Baselines for Ubuntu Corpus Dialogs (2015-10)

A Survey of Available Corpora for Building Data-Driven Dialogue Systems (2015-12)

Incorporating Copying Mechanism in Sequence-to-Sequence Learning (2016-03)

A Persona-Based Neural Conversation Model (2016-03)

How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation (2016-03)

Learning resources

Stanford class on deep learning for NLP.http://cs224d.stanford.edu/syllabus.html

Hinton's Coursera Course. Get it right from the horse's mouth. He explains things well.
https://www.coursera.org/course/neuralnets

Online textbook in preparation for deep learning from Yoshua Bengio and friends. Clear and understandable.
http://www.iro.umontreal.ca/~bengioy/dlbook/

TensorFlow tutorials.
https://www.tensorflow.org/versions/r0.8/tutorials/index.html

Additional Sources

https://cis.ctsi.mcw.edu/deid/

https://cis.ctsi.mcw.edu/nlp/

https://github.com/jayurbain/machine-learning

https://github.com/jayurbain/ctsi-mcw-deid-service

https://github.com/jayurbain/ctsi-mcw-deid-service