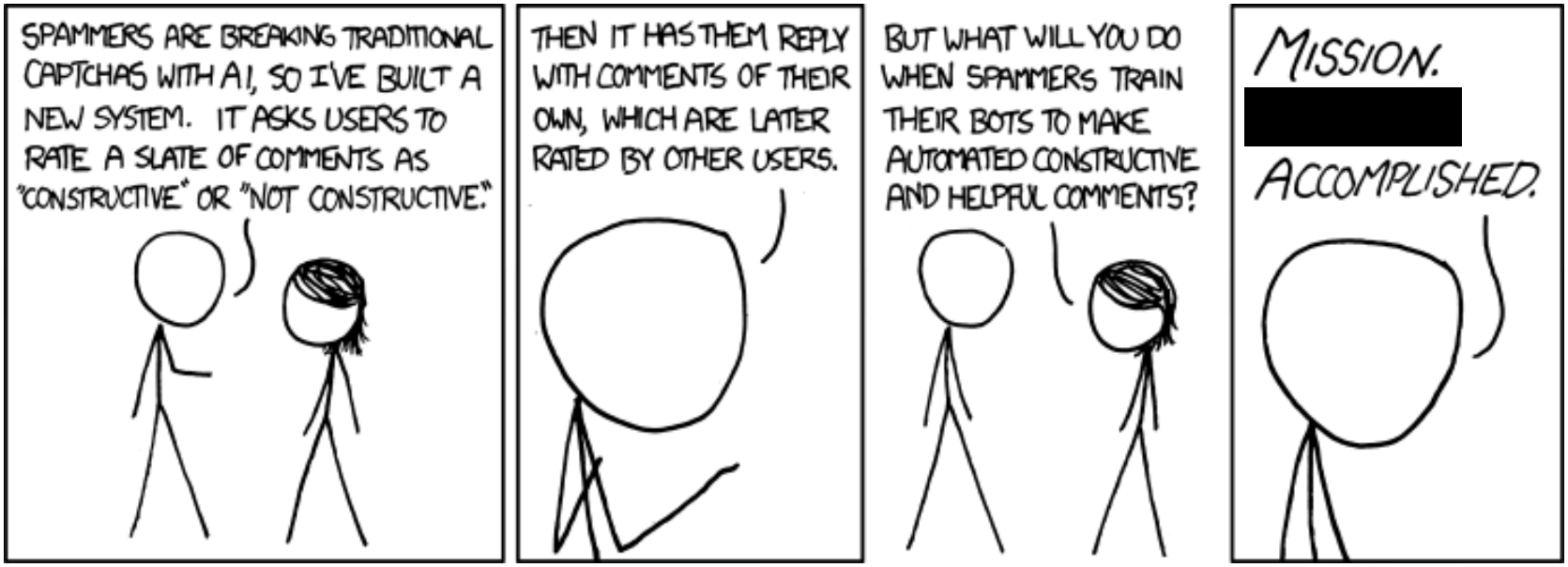
UNIVERSITY of WASHINGTON

Data Science UW Methods for Data Analysis

Intro to Natural Language Processing
Lecture 9
Nick McClure


SPAMMERS ARE BREAKING TRADITIONAL CAPTCHAS WITH AI, SO I'VE BUILT A NEW SYSTEM. IT ASKS USERS TO RATE A SLATE OF COMMENTS AS "CONSTRUCTIVE" OR "NOT CONSTRUCTIVE."
THEN IT HAS THEM REPLY WITH COMMENTS OF THEIR OWN, WHICH ARE LATER RATED BY OTHER USERS.
BUT WHAT WILL YOU DO WHEN SPAMMERS TRAIN THEIR BOTS TO MAKE AUTOMATED CONSTRUCTIVE AND HELPFUL COMMENTS?
MISSION. [redacted] ACCOMPLISHED.

Topics

- Review
- Natural Language Processing
  - Text Normalization
  - Text Distances
  - Corpus/Dictionaries
  - Naïve Bayes
  - Word Frequencies
  - Latent Dirichlet Allocation

Review

- Time Series
  - MA: Moving Average
  - AR: Auto Regressive
  - ARIMA: Auto Regressive Integrated Moving Average
  - Time Series Regression
- Spatial Statistics:
  - Global and point estimation
  - Variograms
  - Kriging
  - Clustering with Ripley's K and Ripley's L

Why Text?

- How much data?
  - Twitter has more text data recorded than all that has been written in print in the history of mankind. (http://www.internetlivestats.com/twitter-statistics/)
  - Most of the world's data is unstructured:
    - 2009 HP Survey: 70%
    - Gartner: 80%
    - Teradata: 85%
- Why use text?
  - Structured (numerical/categorical) data very often misses elements critical to modeling.
    - Un-transcribed notes, comments, logs
    - Surveys, medical charts

Measuring Text Distance

- Hamming Distance
  - Line up strings, count number of positions that are the different.
  - Assumes strings are of the same length.

$Hamming(beer, bear) = 1$ $Hamming(101101, 100011) = 3$

- Levenshtein distance
  - Measures edit distance between two strings (insertion, deletion, substitution only)

$Lev(beer, bear) = 1$ $Lev(banana, ban) = 3$

Measuring Text Differences

- Jaccard index
  - Size of intersection of characters divided by size of union of characters.

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

$$J(beer, bear) = 1 - \frac{3}{4} \qquad J(banana, ban) = 1 - \frac{3}{3}$$

- Weighted Jaccard Index
  - For each letter, calculate the minimum times it appears, m_i and the max, M_i

$$J'(A,B) = 1 - \frac{\sum m_i}{\sum M_i}$$

$$J(beer, bear) = 1 - \frac{m_a + m_b + m_e + m_r}{M_a + M_b + M_e + M_r} \qquad J(banana, ban) = 1 - \frac{m_a + m_b + m_n}{M_a + M_b + M_n}$$

$$J(beer, bear) = 1 - \frac{0 + 1 + 1 + 1}{1 + 1 + 2 + 1} = 1 - \frac{3}{5} \qquad J(banana, ban) = 1 - \frac{1 + 1 + 1}{3 + 1 + 2}$$

- R demo

Text Normalization (Pre processing)

- Strip extra white space:

I <3 statistics, it's my \u1072 fAvoRitE!! 11!!! → I <3 statistics, it's my \u1072 fAvoRitE!! 11!!!

- Remove Unicode text

I <3 statistics, it's my \u1072 fAvoRitE!! 11!!! → I <3 statistics, it's my fAvoRitE!! 11!!!

- Lower case

I <3 statistics, it's my fAvoRitE!! 11!!! → i <3 statistics, it's my favorite!! 11!!!

- Remove punctuation

i <3 statistics, it's my favorite!! 11!!! → i 3 statistics its my favorite 11

- Remove numbers

i 3 statistics its my favorite 11 → i statistics its my favorite

- Remove stop words

i statistics its my favorite → statistics favorite

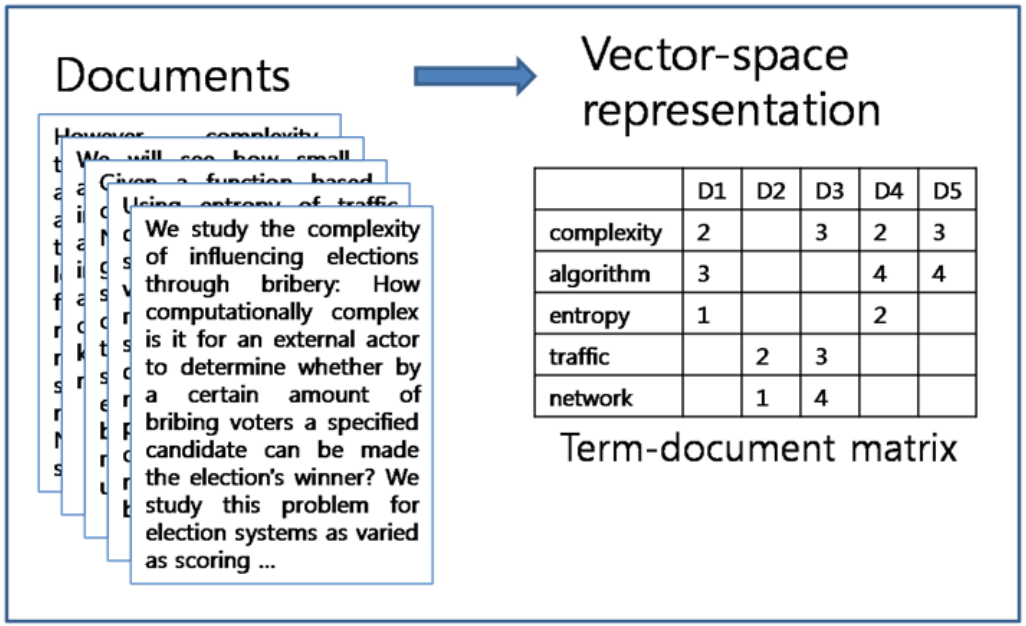
- Stem words (optional)

statistics favorite → statisti favori

- R-demo

Dictionary Creation

- Usually we have a whole bunch of entries, each entry is a sequence of variable length text.
- A document is just one entry.
- A collection of documents is called a corpus.
- We can represent that corpus many ways.
- Term document matrix (shown below) or a Document term matrix (transpose).



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

Naïve Bayes!

- Now we have a numeric representation of key words in our documents.
- We can use called Naïve Bayes algorithm to classify documents.

- Remember:

$$P(A|B) = P(B|A)\frac{P(A)}{P(B)}$$

Or

$$P(h_0|Data) = P(Data|h_0)\frac{P(h_0)}{P(Data)}$$

Another Application of Bayes Theorem (Naïve Bayes Classification)

- Naïve Bayes Classification.
- Let's say we want to test a hypothesis, h_0:

$$P(h_0|Data) = P(Data|h_0)\frac{P(h_0)}{P(Data)}$$

- We pick either the null or alternative, based on which has the highest probability given the data.

Another Application of Bayes Theorem

$$P(h_0|Data) = P(Data|h_0)\frac{P(h_0)}{P(Data)}$$

- We want to predict if a consumer will buy an ad:
- Null hypothesis is yes (or could be no)
- We have observed 10 outcomes in the past:

Age	Income	Student	Credit	Buys Ad
35	Medium	Yes	Fair	Yes
30	High	No	Average	No
40	Low	Yes	Good	No
35	Medium	No	Fair	Yes
45	Low	No	Fair	Yes
35	High	No	Excellent	Yes
35	Medium	No	Good	No
25	Low	No	Good	No
28	High	No	Average	No
35	Medium	Yes	Average	Yes

Another Application of Bayes Theorem

Age	Income	Student	Credit	Buys Ad
35	Medium	Yes	Fair	Yes
30	High	No	Average	No
40	Low	Yes	Good	No
35	Medium	No	Fair	Yes
45	Low	No	Fair	Yes
35	High	No	Excellent	Yes
35	Medium	No	Good	No
25	Low	No	Good	No
28	High	No	Average	No
35	Medium	Yes	Average	Yes

$$P(h_0|Data) = P(Data|h_0)\frac{P(h_0)}{P(Data)}$$

New Data: Consumer is 35 years old
And has a medium income

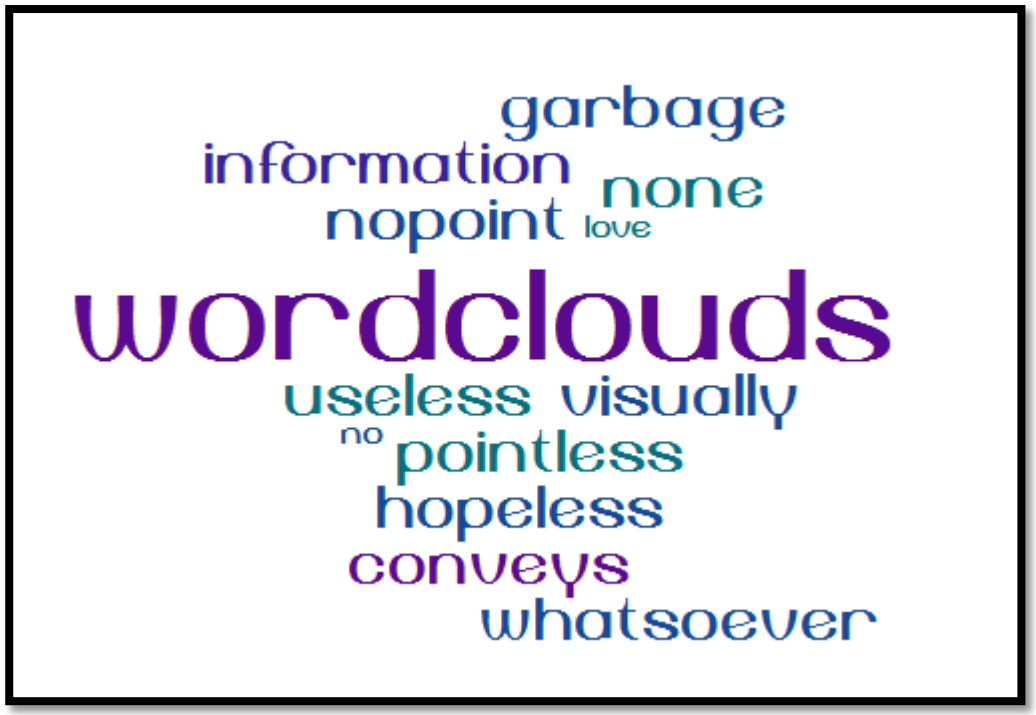
- P(buys Ad)=0.5, P(not buy Ad) = 0.5
- P(35yrs & med income) = 4/10 = 0.4
- P(35yrs & med income | buys ad) = 3/5 = 0.6
- P(35yrs & med income | not buy ad) = 1/5 = 0.2

- P(buy | demographics) = $P(demographics|buy)\frac{P(buy)}{P(demographics)}$
- =0.6 * (0.5 / 0.4) = 0.75
- Similarly, P(not buy | demographics) = 0.2 * (0.5 / 0.4)=0.25

Wordclouds

> Completely useless display of information that people love to see.

> R demo



Creating Features

- Text Frequency: Word frequency or TF
  - Count how many times a word appears in a single document.
  - Terms that occur in fewer documents are more descriptive and may contain more information (Rarity matters).
- Inverse Document Frequency (IDF)
  - Inverse of the proportion of documents containing term in the whole collection.
  - #documents / #documents with word might be too severe.
    - A word appearing twice instead of once shouldn't have twice the impact

$$IDF = \log\left(\frac{\#Documents}{\#Documents\ with\ Word}\right)$$

- Maybe we should tie these together:
- TF-IDF: Rare terms in whole collection that appear frequently in some documents maybe very important!
  - Multiply these two

How to Use TF–IDF

$$TF - IDF = \log\left(\frac{\#Documents}{\#Documents\ with\ Word}\right) \times f(Word)$$

- Finding important words to describe the document collections or subgroups of collections.
- Using the count of important words as a feature in a model.
- Using the distribution of a document's TF-IDF values.
  - Characterize writing styles
  - Comparing authors
  - Determining original authors
  - Finding plagiarism

- R-demo

Latent Dirichlet Allocation

- Creates k-subtopics and determines which topics apply to which document (multiple allowed) with a probability.
- In other words, each document is considered to be generated by drawing words from a mixture of topics.
- Steps:
  - Initialize the number of topics. Use informed estimate or use trial-and error, choosing the number that results in the most interpretable topics.
  - Assign every word to a random topic in each document.
  - Iterate and update topic assignments via:
    - How common is that word across topics (for all documents)?
    - How common is that topic in that document.
- We are trying to estimate the distribution of topics…
  - Via MCMC!!!! (prior = Dirichlet distribution)

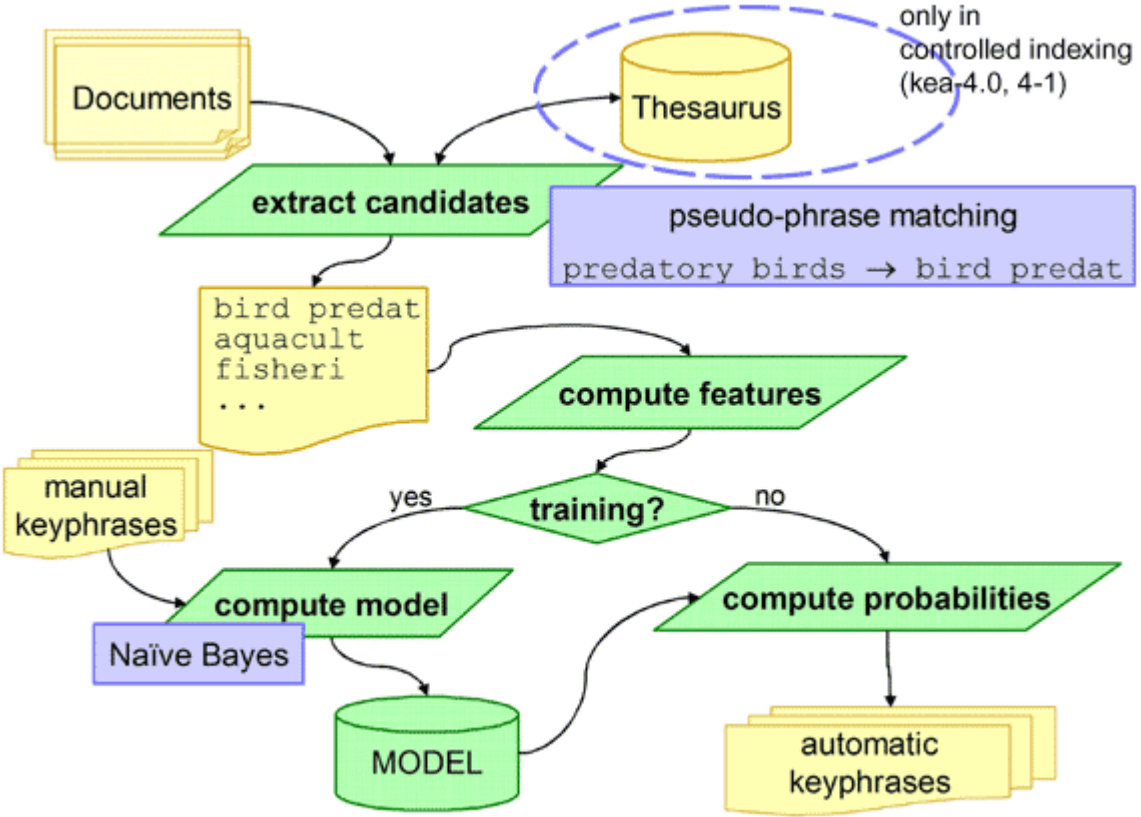
- R-demo

RAKE: Rapid Automatic Keyword Extraction

- LDA is computationally hard and to apply this to many documents is quite time and memory consuming.
- Tweets come in at up to 500k per minute.
- If we can give our algorithm an idea of what to look for:
  - TF-IDF: Keywords will have higher TF-IDF values.
  - First occurrence: Keywords will tend to appear in the beginning of the document, so we calculate the % of document before the first occurrence.
  - Length: Long keywords are not preferred, we know that in English, 3-8 letter words are preferred.
  - Node Degree: Number of phrases (synonyms, rearrangements, stems) that are related to other candidate phrases.

RAKE: Rapid Automatic Keyword Extraction

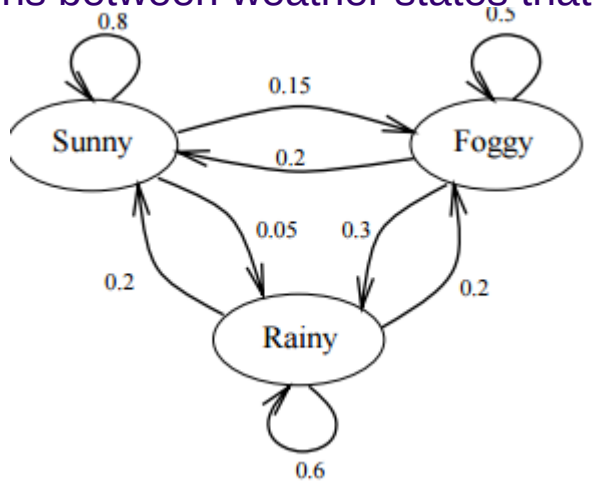
> After candidates are computed, we use a Naïve Bayes model to predict other keywords in the model.



> R-demo

Part of Speech Tagging

- Tag each word in a sentence what kind of word it is.
- There are many many types:
  - https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
- Idea is to train a classifier that takes into account the nearby words to predict current word.
  - There are large sources of already pre-tagged texts out there for us to use.
  - The most common type of model used here is a "Hidden Markov Model".
  - For example, if we observe the weather everyday, we want to devise a day-to-day model that predicts the next day based on the prior day. We want to choose probabilities of transitions between weather states that maximizes the P(data|parameters):



R demo

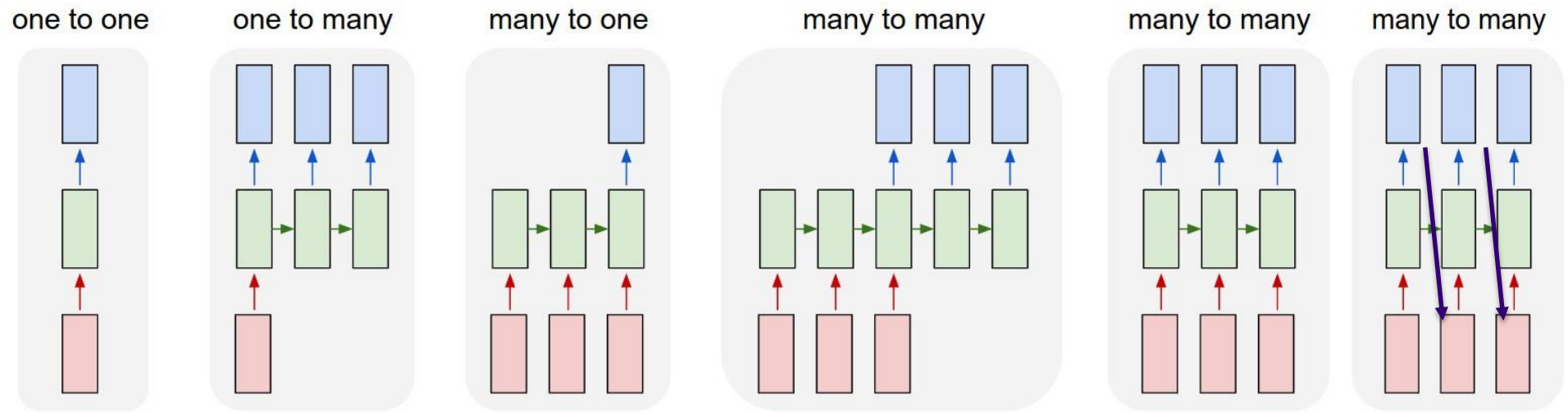
Taking NLP Further

- Know that there are machine learning models that use Neural Networks to represent language.
- Word2Vec is a popular way to represent language.
  - Skipgram: Map each word into a vector of probabilities of appearing in different positions.
  - CBOW: Predict the next word based on all prior word probabilities of appearing.
- If we consider the probability vector of the words, we end up with interesting equations such as:

"King" – "Man" + "Woman" = "Queen"

Taking NLP Further

- Recurrent Neural Networks are neural networks that operate in chains of inputs.
- The input to an RNN is a sequence of (usually) characters in a string.



Standard NN | Image Captioning | Sentiment Analysis | Language Translation | Sequence Labeling or Sequence Prediction

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Further interesting links

- Microsoft Research 2008: Bayesian Inference in Traffic Patterns. http://research.microsoft.com/pubs/101948/TAinfer.pdf
- Google Analytics (Multi armed bandit experiments for AB testing): https://support.google.com/analytics/answer/2844870?hl=en
- Google Research: Voice recognition: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37567.pdf
- CNA: Forecasting insurance loss: http://www.casact.org/education/annual/2010/handouts/C4-Zhang.pdf
- Swype texting: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/39190.pdf

Last Homework

- Use the ‘Beer Review’ data set to:
- (1) Normalize and create text features. (Term Matrix exactly like in examples)
- (2) Reduce matrix size.
- And then EITHER/OR:
  - (A1) Normalize “rating” to be between 0 and 1
  - (A2) Create a logistic model that predicts a normalized review score between zero and one.
  - (B) Pick a style (w/more than 500 reviews) and create a logistic classifier that predicts if a review is pertaining to that style.
- Use all the features you want, state, date, rating, date factors, etc.
- **EXCEPT DON’T USE ‘brewery’ or ‘link’.**