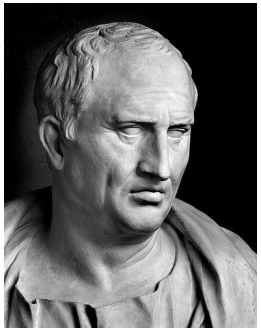
Quotation Construction: A Generative Approach

Cornell CIS

- **Purpose**
  - Use Artificial Intelligence and Machine Learning methods to model human language
  - To build a system that generates quotations which are indistinguishable from human quotations
- **Solutions**
  - N-gram probabilistic Language Models
  - Recurrent Neural Net based Language Model
  - Long Short Term Memory and Gated Recurrent Unit based Language Models

Data Collection

- **Cicero Corpus**
  - First Iteration: 186 unique quotations pulled exclusively from BrainyQuote
  - Second Iteration: 4,354 unique sentences from Republic of Cicero, Treatises on Friendship and Old Age, Letters of Marcus Tullius Cicero, and a variety of transcribed speeches to augment initial dataset
- **Trump Corpus**
  - First Iteration: 577 quotations pulled exclusively from BrainyQuote
  - Second Iteration: 2,379 tweets from Donald Trump's official twitter account to replace initial dataset



N-gram Language Models

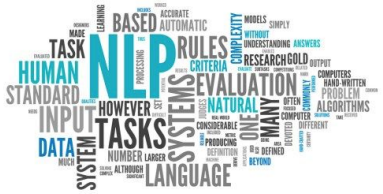
Cornell CIS

- **Bigram Language Model**
  - P(wi | wi-1) = count(wi,wi-1)/count(wi-1)
  - Compute the word frequency of the current word given the immediately preceding word
- **Trigram Language Model**
  - P(wi | wi-1,wi-2) = count(wi, wi-1, wi-2)/count(wi-1, wi-2)
  - Compute the word frequency of the current word given the past two words
- **Handling Unseen N-grams**
  - Smoothing Techniques
    - Add one
    - K-smoothing
    - Kneser Ney
- **Handling Unknown Words**
  - Introduce a ‘UNK’ token into vocabulary
  - Map all rare words (based on word frequency) to this token
  - Put uniform random distribution over all words mapped to ‘UNK’ token

Demo

- **Cicero Quiz Link:**
  - https://goo.gl/forms/eYgZmQsJMMdfATjw1

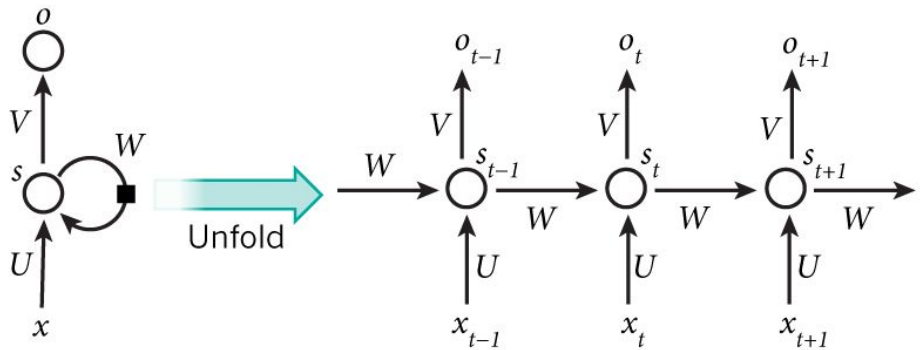
- **Donald Trump Quiz Link:**
  - https://goo.gl/forms/yTv29gNEOguivH0B3



Recurrent Neural Net Language Model

Cornell CIS

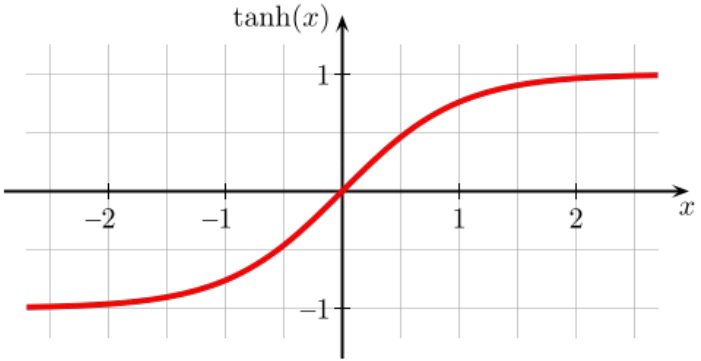
- **Intuition**
  - Look at the entire past sentence when predicting the next word
    - *I grew up in France, so I am fluent in _____*
    - Earlier location (i.e. France) should influence a later language (i.e. French)
- **Numpy Implementation**
  - Following a tutorial to help with the math, built a Numpy RNN from the ground up
    - Poor results on small dataset
    - Incredibly slow on larger dataset
- **Theano Implementation**
  - Uses GPU for matrix operations
  - Poor results on large dataset



Recurrent Neural Net Language Model

Cornell CIS

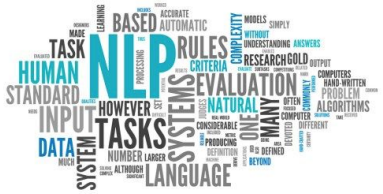
- **Vanishing Gradient**
  - Gradient of tanh activation function at tails is 0
  - Saturated neuron in unrolled chain drives gradient down for past time steps
  - Longer the unrolled chain, the more effect
  - Particularly relevant for language models: length of the chain (i.e. depth) dependent on sentence length
- **Effect**
  - RNN unable to learn long or even mid-range inter-sentence word dependencies



Recurrent Neural Net Language Model

Cornell CIS

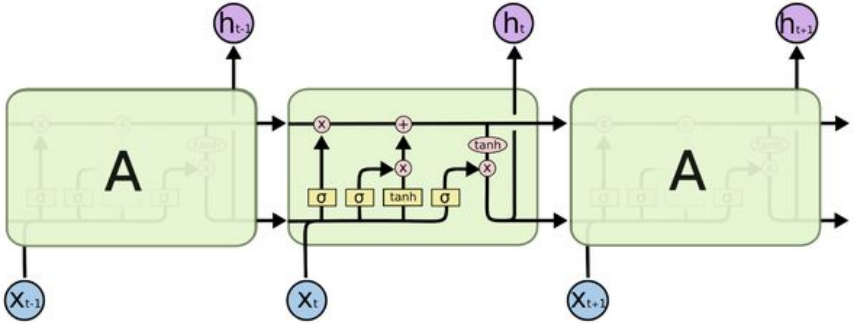
- **Cicero Generated Sentences**
  - Most RNN based Language Model sentences are poorly formed. Best results occur with short sentences.
    - “Messengers from men is appreciated seeing.”
    - “Describe the roman difficult fire that receiver demanding.”
    - “Lifetime my republic he beware the surprised method.”
- **Trump Generated Tweets**
  - Inherently chaotic structure of tweet => leniency for generated sentence
    - twitter good!
    - laughingstock president food.
    - shots very fired continue liar and party disaster @ turbines change.
- **Analysis**
  - Many poorly formed sentences
  - Unable to capture natural sentence structure (i.e. subject - verb - object)

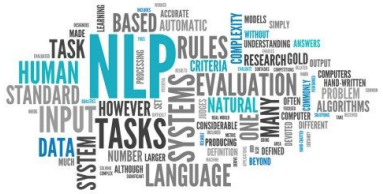


Long Short Term Memory Language Model

Cornell CIS

- **LSTM Architecture**
  - Developed in 1997 to solve the vanishing gradient issue
  - Internal Memory in addition to hidden state at each timestep
- **Gating Mechanisms**
  - Input gate: creates a candidate state with input
  - Forget gate: what information from past time step should propagate to current time step
  - Output gate: determines final hidden state

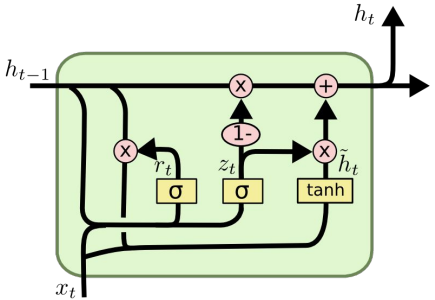




Gated Recurrent Unit Language Model

Cornell CIS

- **GRU Architecture**
  - Developed in 2014 to be a lightweight variant of the LSTM Network
  - Similar to RNN, there is a hidden state but not internal memory at each time step
- **Gating Mechanisms**
  - Update gate: what information from the previous hidden state and current input should be retained in the current hidden state
  - Reset gate: what information from the past hidden state should be forgotten
- **Embedding Layer**
  - Add pre-computed word embeddings
  - Allows the model to generalize better
  - Intuition on how words relate



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

Gated Recurrent Unit Language Model

Cornell CIS

- **Cicero Generated Sentences**
  - For themselves I will say against to the are beginning of a will I either discharge this in the concerning up in virtue.
  - What honour roman be senate to which titles ?
  - I every do not consul.
- **Trump Generated Tweets**
  - I want $ in calls .
  - president obama recognition - sad!
  - sadly they are not smart?
- **Analysis**
  - Many poorly formed sentences
  - The longer the sentence, the more likely something will go wrong
  - Learns the correct inter-sentence word pattern: subject-verb, verb-object, etc.

Intrinsic Evaluation

Cornell CIS

- **Perplexity**
  - Measure of how well our language model is able to generalize and capture the inter-sentence word dependencies of a held-out test set.

Trump Corpus

Language Model	Perplexity
Bigram	236.32
Trigram	431.52
Recurrent Neural Net (Theano)	463.66
GRU Neural Net (Theano)	315.38

Cicero Corpus

Language Model	Perplexity
Bigram	358.65
Trigram	139.64
Recurrent Neural Net (Theano)	773.08
GRU Neural Net (Theano)	116.85

Extrinsic Evaluation

Cornell CIS

- **Cicero Quiz Results:**
  - First iteration BrainyQuote dataset
  - 21 respondents
  - Bigram model: 66.3% average correctness score
  - Trigram model: 53.9% average correctness score
  - Average total scores higher than expected score ($p < 0.001$)
  - Quiz takers had more trouble with trigram model vs. bigram model ($p = 0.0001186$)
- **Trump Quiz Results:**
  - First Iteration BrainyQuote dataset
  - 24 respondents
  - Bigram model: 62.8% average correctness score
  - Trigram model: 60.7% average correctness score
  - Average total scores higher than expected score ($p < 0.001$)

Final Remarks

Cornell CIS

- **Overfitting Can Be Good!**
  - N-gram models trained on small corpus of ‘quality’ data produced amazingly good results
  - Amalgamate partial sentences from training data to produce quotation
  - Overfitting to training data is good. We want to produce quotations which are likely to have been in the training data
- **GRU does great on tweets**
  - Added complexity of GRU based language model
  - Inherent chaotic nature and malleable inter-sentence word ordering
  - Suprasses n-gram language model in terms of tweets produced
  - Doesn’t piece sentences together (as in the n-gram models) but learns inherent tweet structure (i.e. following a verb, a noun will have a high probability of being selected).