



NATURAL LANGUAGE PROCESSING



# A LOOK INTO YELP: CAN WE PREDICT A RATING?

Susmitha Sayana and Saloni Tripathi

# Presentation Outline

1. **References**
2. Introduction to Yelp data + Problem
3. Previous Research Methods
4. *Nugget*: Our method using RNN
5. *Nugget*: RNN Implementation
6. Results
7. Future Applications

# References

**Asghar , Nabiha.** “Yelp Dataset Challenge: Review Rating Prediction.” *Computing Research Repository*, 17 May 2016, [arxiv.org/pdf/1605.05362.pdf](https://arxiv.org/pdf/1605.05362.pdf).

Doan, Tri, and Jugal Kalita. “Sentiment Analysis of Restaurant Reviews on Yelp with Incremental Learning.” *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, doi:10.1109/icmla.2016.0123.

**Guo, Yiwon, et al.** “Predicting Restaurants’ Rating And Popularity Based On Yelp Dataset.” *Stanford University*, 2017, [cs229.stanford.edu/proj2017/final-reports/5244334.pdf](https://cs229.stanford.edu/proj2017/final-reports/5244334.pdf).

Mubarok, Mohamad Syahrul, et al. “Aspect-Based Sentiment Analysis to Review Products Using Naïve Bayes.” 2017, doi:10.1063/1.4994463.

Tang, Duyu, and Meishan Zhang. “Deep Learning in Sentiment Analysis.” *Deep Learning in Natural Language Processing*, 2018, pp. 219–253., doi:10.1007/978-981-10-5209-5\_8.

**Nielsen, Michael.** “Neural Networks and Deep Learning.” *Neural Networks and Deep Learning*, Determination Press, 1 Jan. 1970, [neuralnetworksanddeeplearning.com/](http://neuralnetworksanddeeplearning.com/).

**Olah, Christopher.** “Understanding LSTM Networks.” *Understanding LSTM Networks -- Colah's Blog*, Github.com , 27 Aug. 2015, [colah.github.io/posts/2015-08-Understanding-LSTMs/](https://colah.github.io/posts/2015-08-Understanding-LSTMs/).

Goodfellow, Ian, et al. “Deep Learning.” *Deep Learning*, MIT Press, 2016, [www.deeplearningbook.org/](http://www.deeplearningbook.org/).

# Presentation Outline

1. References
- 2. Introduction to Yelp data + Problem**
3. Previous Research Methods
4. Our method using RNN
5. RNN Implementation
6. Results
7. Future Applications

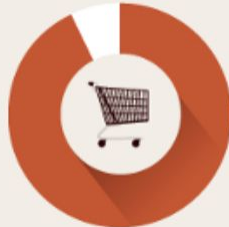


## Local-search service

- Average of 150+ million users accessing mobile page every month
  - Revenue of over \$850 million in 2017
- Not just restaurants reviews
  - Users submit reviews for Shopping more than anything else
- Yelp is all over the world
  - Largest age group from 19-34
- Helps small businesses through reviews
  - Usually spend 60% of time on social media, yelp to advertise

### YELP = PURCHASES

Yelp is a powerful deciding factor for many consumers. Reviews and ratings heavily inform where and what they choose to purchase.



**93%**

of people who conduct research on review sites typically make purchases at the businesses they look up.

# Reviews are the most convincing aspect of Yelp

## YELP REVIEWS ARE TRUSTED REVIEWS

Consumers consider Yelp a trusted source for reviews and opinions.



**72%**

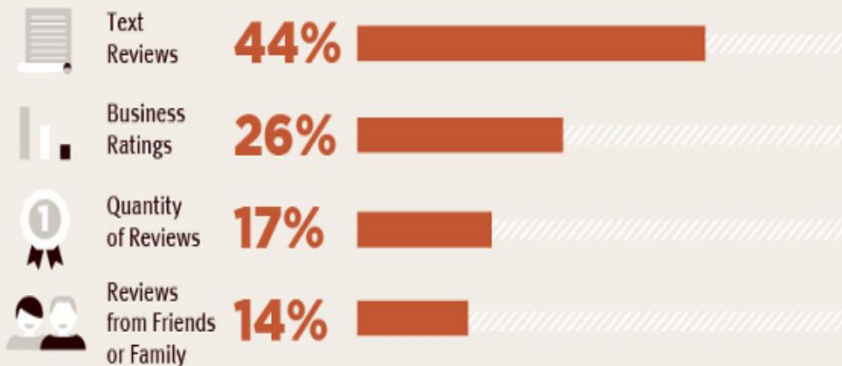
of consumers trust online reviews as much as personal recommendations.



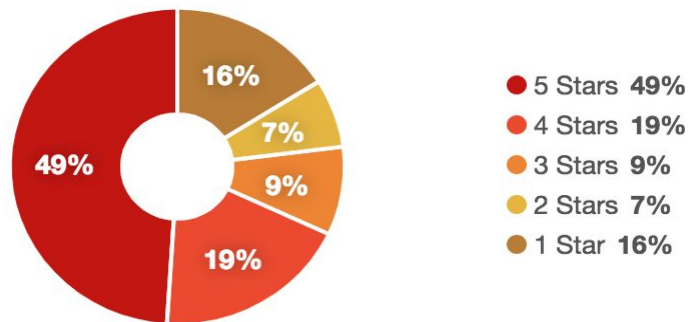
**90%**

of Yelp users say positive reviews impact their company buying choices.

When deciding what businesses to go to, customers based their choices on:



### Rating Distribution

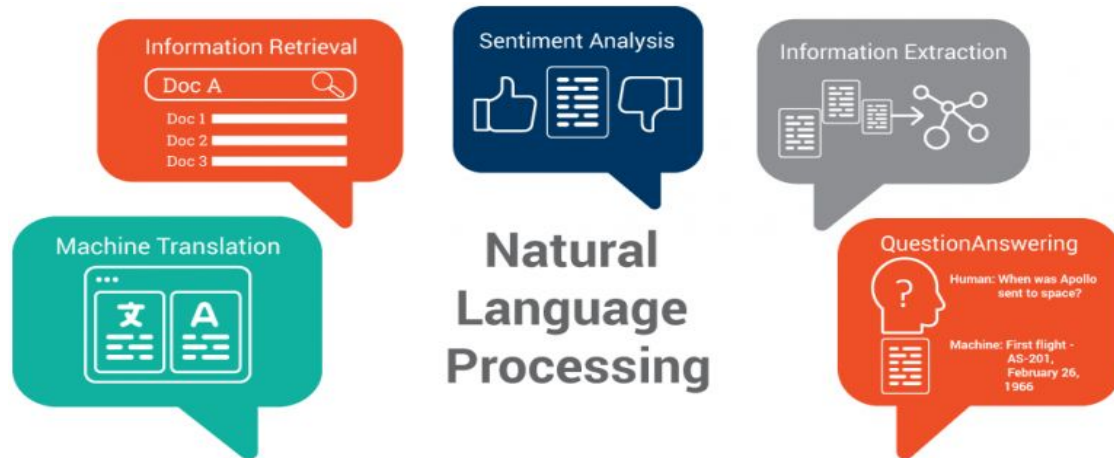


# Using the text reviews from Yelp, can we predict the star rating they give?

- Using Natural Language Processing (NLP), we can attempt to use computation techniques on natural language (in any form) to teach the computer to derive a meaning from that
- Meanings can vary by the sentiments in the text to identify the primary subject of a sentence

Definitions paraphrased: <https://towardsdatascience.com/predict-product-success-using-nlp-models-b3e87295d97>

Image taken from: <https://medium.com/greyatom/introduction-to-natural-language-processing-78baac3c602b>



# How we can use sentiment analysis?

Sentiment analysis is the automated process of understanding an opinion about a given subject from written or spoken language

- Most common way to use sentiment analysis is to classify text to class
  - Many studies use sentiment analysis for this same purpose with reviews
- First step towards sentiment analysis is to collect data
  - Application Program interfaces (APIs) can help to collect data
  - Yelp Dataset from Kaggle
    - Collected for Kaggle challenge
    - Dataset includes reviews, in text and star rating, images, business type, etc.
- Second step in sentiment analysis is pre-processing
  - Technique to reduce noise of text, dimensionality
    - Remove stopwords, remove numbers, stemming, lowercase, remove punctuation

<https://medium.com/@tomyuz/a-sentiment-analysis-approach-to-predicting-stock-returns-d5ca8b75a42>



# Directions within Sentiment Analysis

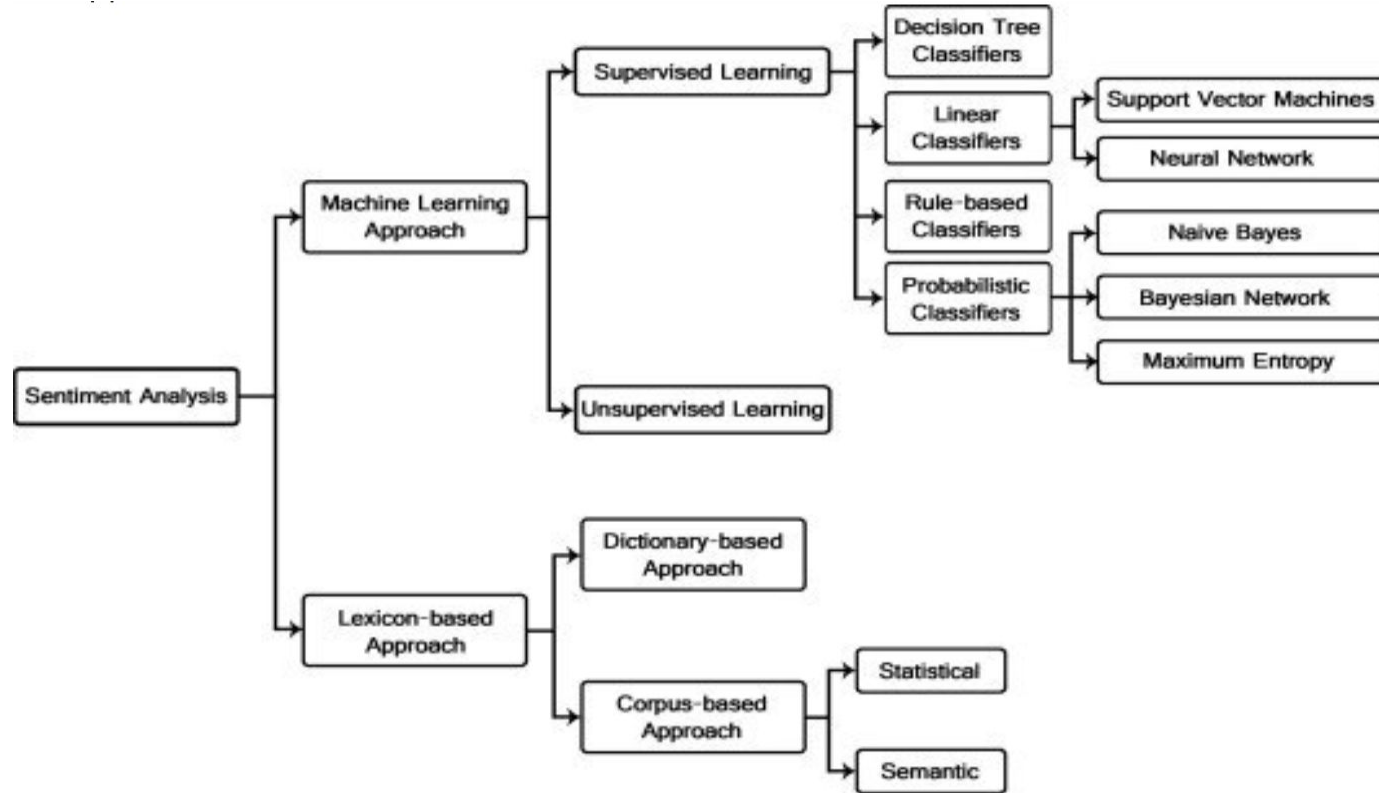


Image taken from: <https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html>

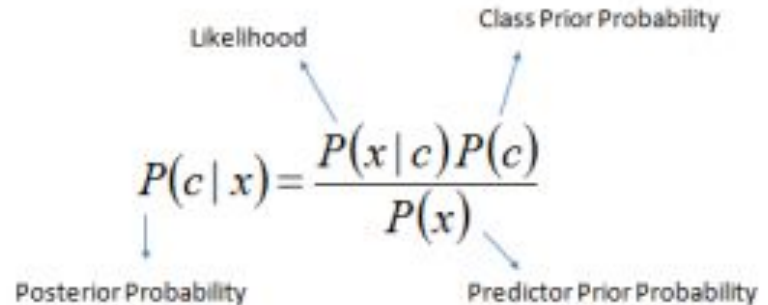
# Presentation Outline

1. References
2. Introduction to Yelp data + Problem
3. **Previous Research Methods**
4. *Nugget*: Our method using RNN
5. *Nugget*: RNN Implementation
6. Results
7. Future Applications

# Previous Research: Naïve Bayes

- Multiple research studies on this same or similar Yelp dataset has been done using Naïve Bayes model
  - Naïve Bayes is a classification technique based on Bayes Theorem, and relies on the assumption of independent predictors
  - Usually starts by separating dataset into frequency table and using this frequency to predict likelihood probabilities
- Easy to use Naïve Bayes Classifier to predict class of dataset
  - Is best to use when independence of predictors holds
  - Performs best when using categorical variables rather than numerical variables

- $P(c|x)$  is the posterior probability of *class* ( $c$ , *target*) given *predictor* ( $x$ , *attributes*).
- $P(c)$  is the prior probability of *class*.
- $P(x|c)$  is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$  is the prior probability of *predictor*.



The diagram shows the formula for posterior probability:  $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ . Arrows point from the terms to their definitions:  $P(c|x)$  to Posterior Probability,  $P(x|c)$  to Likelihood,  $P(c)$  to Class Prior Probability, and  $P(x)$  to Predictor Prior Probability.

Taken from: <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

# Previous Research: Logistic Regression

- First run a simple linear regression with the selected features:
  - One problem with linear regression is that the dependent variable should be continuous, instead of discrete
- Multinomial Logistic Regression predicts the probabilities of the dependent variable falling into a given category and classifies the prediction into the class with the highest probability
- Logistic Regression is an algorithm that can be trained as a classifier and uses a linear decision boundary as a means of classifying a set of features
- Being that Logistic Regression is conditional opposed to a generative supervised learning algorithm, we decided it was a great choice to contrast against Naive Bayes
- Logistic Regression is a great first conditional classification algorithm to learn and can be used to classify multivariate data as well as fit polynomial terms to find decision boundaries

With the parameter  $\theta$ , it applies softmax function to compute the probability for each category:

$$P(y = i|x; \theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^k e^{\theta_j^T x}}.$$

To estimate the parameter  $\theta$ , maximum a posteriori estimation (MAP) is often used, which is an extension of maximum likelihood estimator with regularization.

Taken from: "Predicting Restaurants' Rating And Popularity Based On Yelp Dataset" by Yiwen Guo, et al.

# Previous Research: Support Vector Machine (SVM)

- Known as Linear SVM using scikit-learn, we need to import LinearSVC
- Has been shown to perform well on several text classifications tasks
- SVM will try to find a way to separate the different classes of our data
  1. Vectorized our text, so each review is now represented as a set of coordinates in a high-dimensional space
  2. During training, the SVM will try to find some hyperplanes that separate our training examples
  3. When we feed it the test data (minus the matching labels), it will use the boundaries it learned during training to predict the rating of each test review

```
1 from sklearn.svm import LinearSVC
2
3 # initialise the SVM classifier
4 classifier = LinearSVC()
5
6 # train the classifier
7 t1 = datetime.now()
8 classifier.fit(X_train, y_train)
9 print(datetime.now() - t1)
```

```
1 preds = classifier.predict(X_test)
2 print(list(preds[:10]))
3 print(y_test[:10])
4
5 >>>[4, 1, 1, 2, 5, 4, 1, 5, 5, 1]
6 >>>[5, 1, 1, 3, 5, 4, 1, 5, 5, 2]
```

# Data & text review distribution

- 80% of the dataset for training, 20% for testing
- total of 1000 reviews that contained a Yelp review and the binary labeled sentiment score
- Used EDA (Exploratory Data Analysis) to tease trends and clues in data
- Data set looks like after each review has been
- Distribution of lengths of positive and negative reviews as this has the potential to skew our model
  - identical number of positive and negative scores with similar distributions of the length of review

Taken from:

<https://medium.com/@aariff.deen/creating-a-real-time-star-prediction-application-for-yelp-reviews-using-sentiment-analysis-9c7e94978bf6>

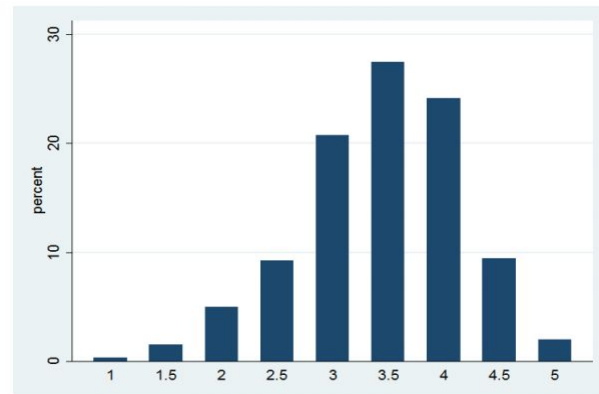
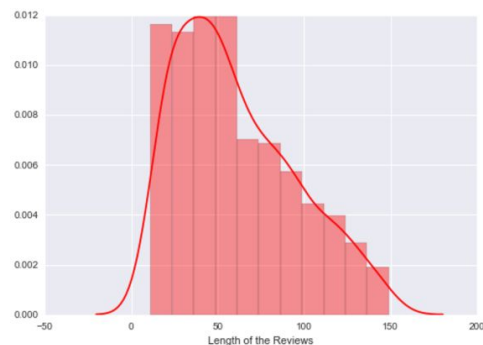
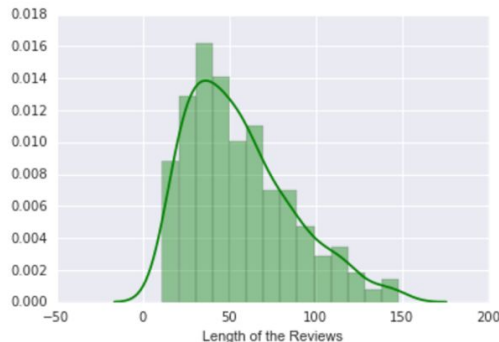


Fig. 1. Rating distribution

	reviews	sentiment
0	[crust, is, not, good]	0
1	[not, tasty, and, the, texture, was, just, nasty]	0
2	[stopped, by, during, the, late, may, bank, ho...	1
3	[the, selection, on, the, menu, was, great, an...	1
4	[now, i, am, getting, angry, and, i, want, my,...	0



# Comparing Logistic Regression

- Compare best predictor-logistic regression with a random-number predictor, and a constant-number predictor
  - Random-number predictor makes prediction based on a random number generator
  - Constant-number predictor always predicts the restaurant to be 3.5 stars, since this category has the largest share of restaurants
- Logistic regression performs significantly better than the random number predictor, but only slightly better than the constant-number predictor
- Best performed method is logistic regression possibly because it is more robust
  - result can be utilized to provide restaurant improvement suggestions for business owners

Taken from: "Predicting Restaurants' Rating And Popularity Based On Yelp Dataset" by Yiwen Guo, et al.

$\alpha$	Training Accuracy	Test Accuracy
$10^4$	0.3394	0.3392
$10^2$	0.3883	0.3867
$10^0$	0.3894	0.3884
$10^{-2}$	0.3895	0.3883
$10^{-4}$	0.3883	0.3883

TABLE III  
PREDICTION RESULTS WITH REGULARIZATION FOR LINEAR REGRESSION.

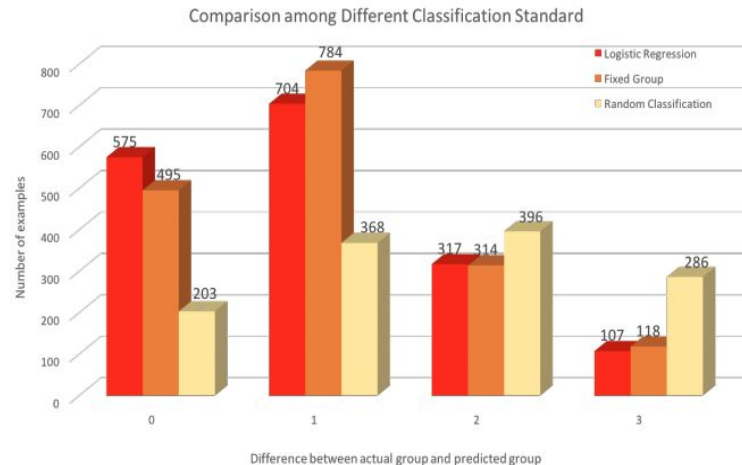
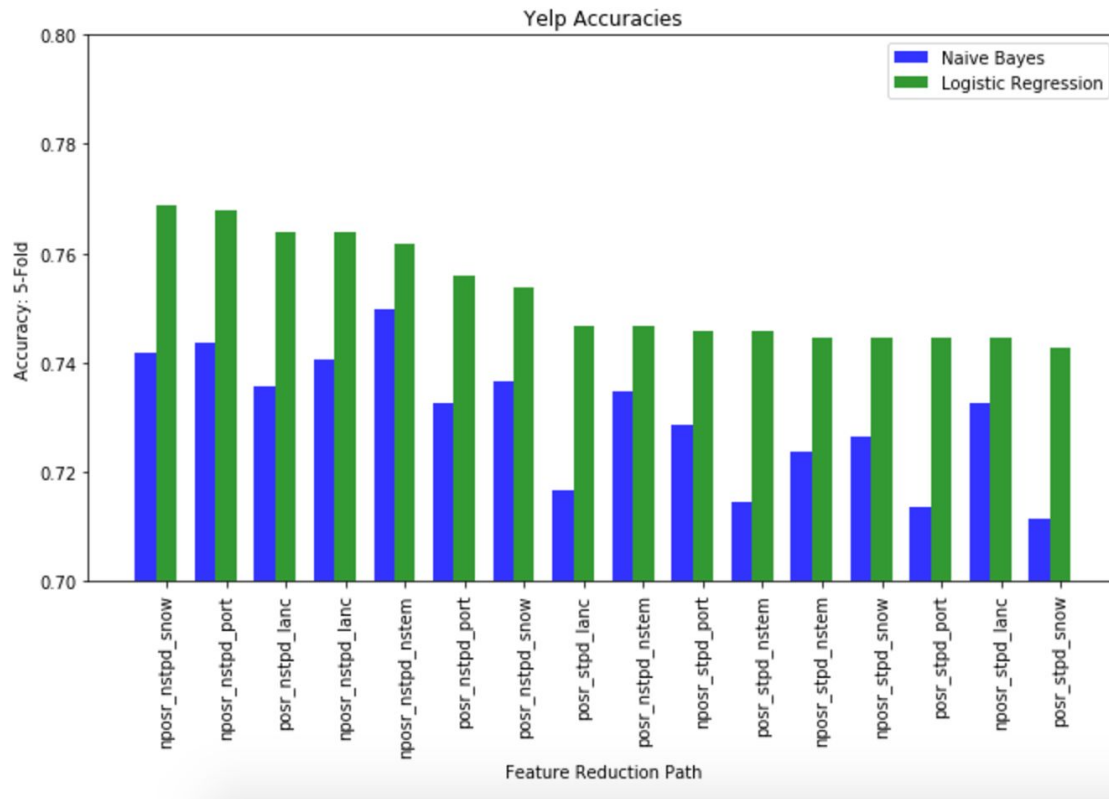


Fig. 9. Comparison between logistic regression predictor, random-number predictor, and constant-number predictor

# Comparing Naive Bayes and Logistic Regression

- In this evaluation technique, they first break our dataset into 5 equal portions
- Tested the algorithm five times on each portion, while training it first on the remaining portions each time
- This experiment used 5-fold cross validation and found that Logistic Regression consistently outperformed Naive Bayes in accuracy





# Comparing different prediction models: Results

- Each of the sixteen prediction systems, we perform 3-fold cross validation on the training set and compute two metrics, Root Mean Squared Error (RMSE) and accuracy
- overall trends are quite the same for the training and validation (test) datasets
- Perceptrons are still the worst, followed by the Naive Bayes classifier
- Logistic Regression achieved the highest accuracy of 64% using the top 10,000 Unigrams & Bigrams as features
- Adding trigrams to the previous model does not help, because trigrams repeat rarely.
  - unlikely that two different user would use the exact same 3-tuple to describe a restaurant

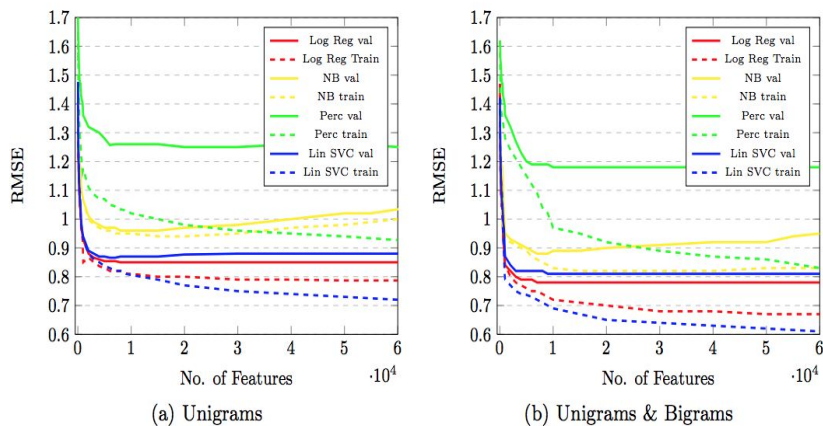


Figure 3. RMSE plots for (a) Unigrams, and (b) Unigrams & Bigrams

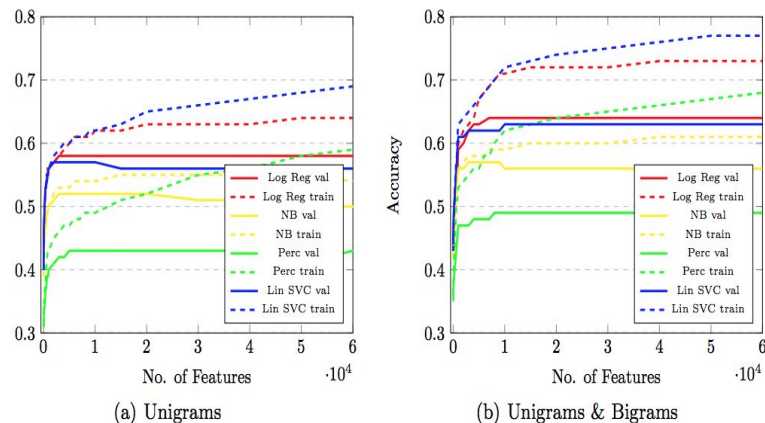


Figure 4. Accuracy plots for (a) Unigrams, and (b) Unigrams & Bigrams

# Presentation Outline

1. References
2. Introduction to Yelp data + Problem
3. Previous Research Methods
4. ***Nugget*: Our method using RNN**
5. *Nugget*: RNN Implementation
6. Results
7. Future Applications
- 8.

# History of NNs

**1943-** The first artificial neuron was the Threshold Logic Unit (TLU) , Warren McCulloch and Walter Pitts. A computational representation of the structure of neurons in a human brain

**1950-** “Computing Machinery and Intelligence,” Alan Turing imagined conversing with such a computer via a teleprinter

Alan Turing proposes a 'learning machine' that could learn and become artificially intelligent.

- very year that Turing's paper went to print, *I, Robot*, a collection of Isaac Asimov's short stories about intelligent humanoids

**1951-** *First Neural Network Machine:* Marvin Minsky and Dean Edmonds build the first neural network machine

**1957-** *Discovery of Perceptron:* Frank Rosenblatt invents the perceptron while working at the Cornell Aeronautical Laboratory.

**1981-** *Discovery of Explanation Based Learning:* Gerald Dejong introduces Explanation Based Learning

**1982-** *First Recurrent Neural Network:* John Hopfield popularizes Hopfield networks, a type of recurrent neural network

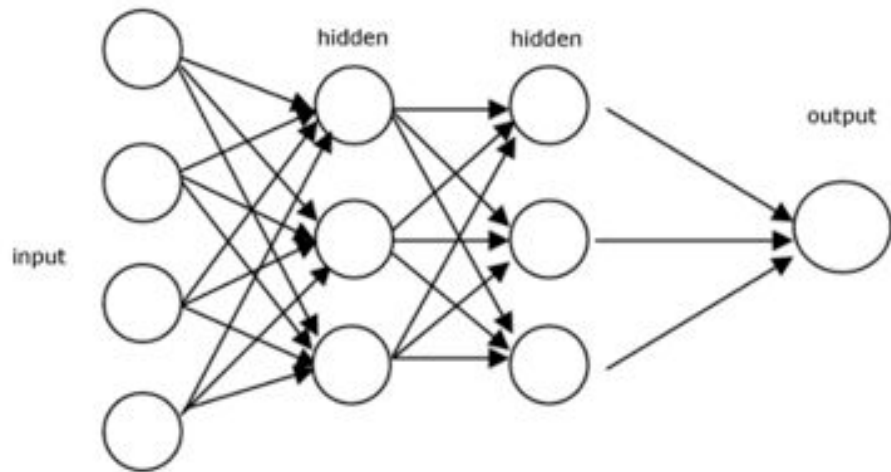
**1986-** *Application of Backpropagation* by Seppo Linnainmaa's reverse mode of automatic differentiation

# Understand RNNs via:

- 1) Feedforward Networks
- 2) Recurrent Networks
- 3) Recurrent Neurons
- 4) Backpropagation Through Time (BPTT)
- 5) RNN Implementation

# Feedforward Networks

- Channelling information through a series of operations that take place in the neurons of a network
- FFNs pass the information directly through each layer ONLY once
- FFNs take an input and produce an output from it, the output is usually a classification
- FFNs are trained on a set of pre-labelled data
- objective of the training phase: to reduce the error while FFN tries to guess the class
- error in FFN is reduced by updating weights in the backward pass
- once training is complete, use the weights to classify new sets of data
- At test time, FFNs cannot recall previous input data → point in time decision making



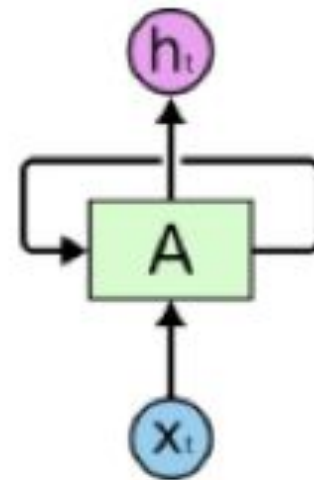
A typical feed-forward network architecture

# Algorithm

- 1.) **Forward propagation:** forward propagate the activations through all layers from input to output, reaching prediction
- 2.) **Compute loss function:** compute error between prediction and ground truth
- 3.) **Backpropagation:** use the chain rule for differentiation for backpropagating the gradients through the layers in the opposite direction from the output to the input
- 4.) **Update:** weights

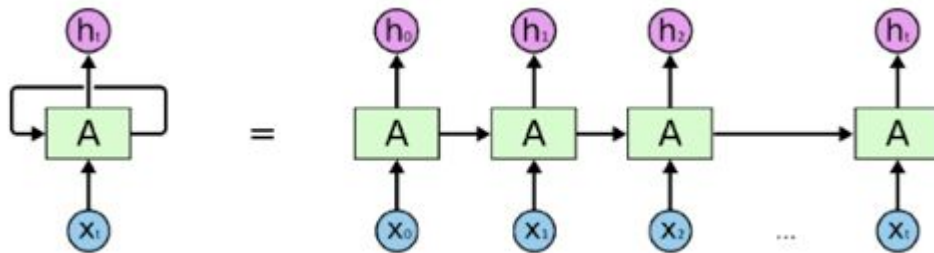
# What is a RNN?

- 1986
- Recurrent Neural Network are networks with loops in them, which allows information to be carried over through time
- RNNs do the same operation over and over on sets of sequential input (definition of operation tbd later)
- RNNs have loops:
- A chunk of neural network called 'A', takes an input  $x_t$  and outputs a value  $h_t$
- A loop allows information to be passed from one step of the network to the next



# What is a RNN?

- Loops make RNNs seem complicated, but they're just copies of the same network
- Each network passes information to the next network and overtime, the model starts to learn
- If we unwind the loop, we're left with a sequence of networks:



- shows that RNNs are closely related to sequences and lists
- RNNs are actually the architecture of NNs used for sequences and lists
- So in order to understand RNNs, we should first understand sequences

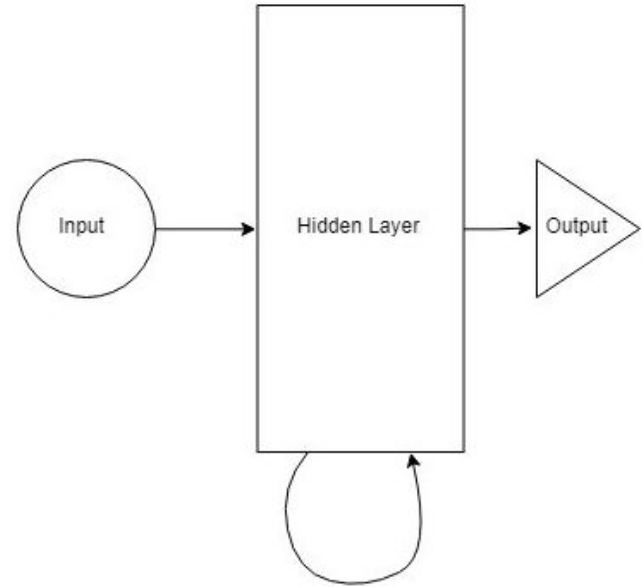


# Why do we need RNNs?

- Sequence: is a stream of data (finite or infinite), which are interdependent
- Sequential Models: Time Series
- time series data: are informative pieces of strings
- Examples:
  - Sentence of words; sequence of characters
  - Image of handwritten words or characters
  - Sequence of DNA
  - Video of image frames
- Scenarios, such as sentence generation or text translation, where words generated are dependent on words generated before
- We need to have some bias based on the previous output and this is when we use a RNN
- RNNs have some memory about what happened earlier in the sequence of data
- helps the system to gain context and to eventually 'learn'

# Recurrent Neurons

- In the diagram, the hidden layer applies the forward propagation to the current input as well as to the previous state
- The output is compared to the actual output and then an error value is computed
- The network learns by back propagating the error via the network to update the weights
- The standard backpropagation used in FFNs won't work here, because RNNs are cyclic
- FFNs can use the above layer to calculate the error derivatives, but RNNs have a different structure
- The new algorithm is called Backpropagation Through Time (BPTT)

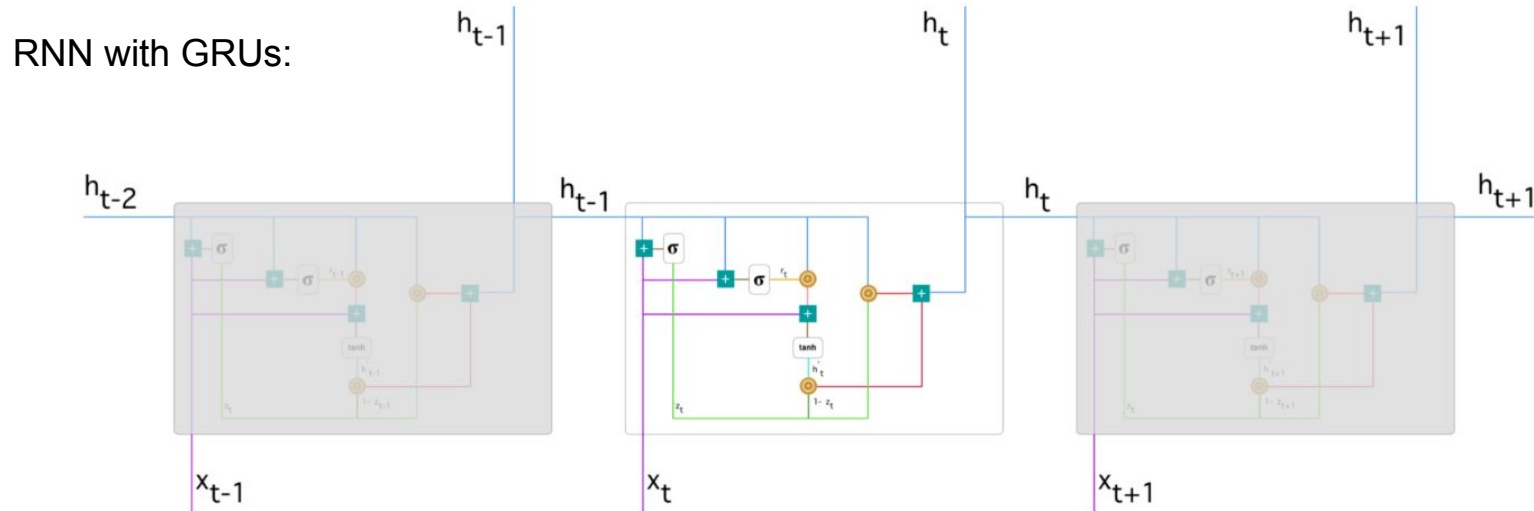


# The Problem of Long-Term Dependencies

- RNNs are popular because they can connect previous information to the present task
  - Ex.) Using previous video frames to understand the present frame
- But can RNNs actually do this?
- Yes if: implementing the present task only requires recent information
  - The gap between the relevant info. and the place it's needed is SMALL
- No if: we need more context
- As the gap increases in size, RNNs become unable to learn to connect previous information to the present task
- Long Short Term Memory cells (LSTMs) solve this problem

# Gated Recurrent Units

- One version of RNNs is the GRU, which still uses a recurrent network, but now has two specific gates: an update gate, and a reset gate
- Each unit of a GRU passes information in the form of a memory state using the previous memory state and current input
- In this way, it behaves in a similar fashion to the recurrent neural networks we have seen



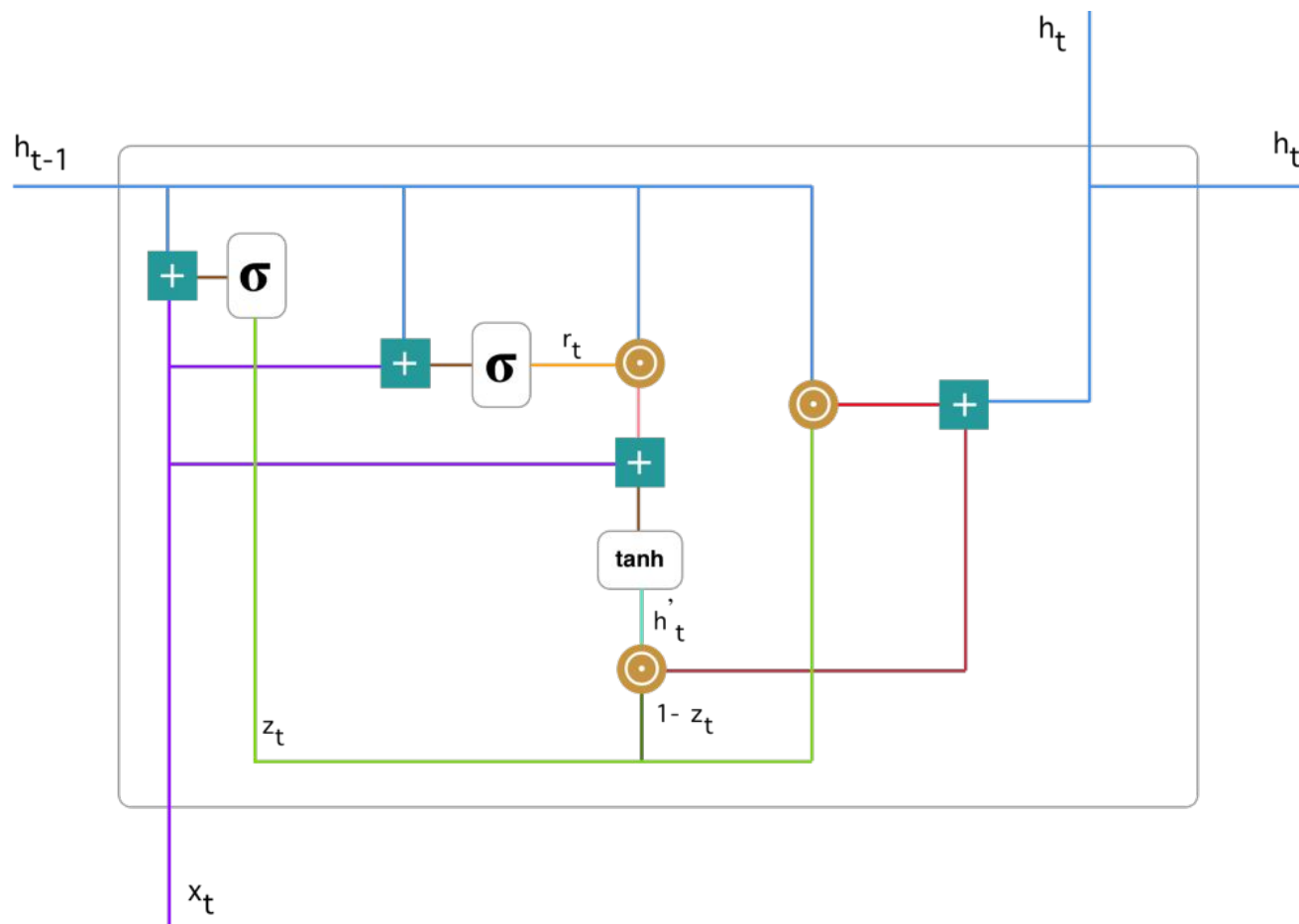


Image: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>

# Introducing labels for gradient calculation

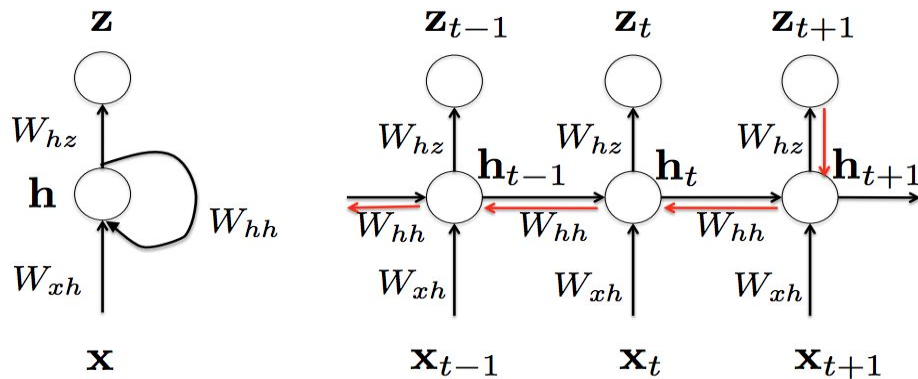


Figure 1: It is a RNN example: the left recursive description for RNNs, and the right is the corresponding extended RNN model in a time sequential manner.

- RNN model above has one hidden layer as depicted in Fig. 1. Notice that it is very easy to extend the one hidden case into multiple layers, which has been discussed in deep neural network before.
- Considering the varying length for each sequential data, we also assume the parameters in each time step are the same across the whole sequential analysis. Otherwise it will be hard to compute the gradients.

# Calculations of gradient for RNN

RNN models a dynamic system, where the hidden state  $\mathbf{h}_t$  is not only dependent on the current observation  $\mathbf{x}_t$ , but also relies on the previous hidden state  $\mathbf{h}_{t-1}$ . More specifically, we can represent  $\mathbf{h}_t$  where  $f$  is a nonlinear mapping:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

In other words, RNN can use the hidden variables as a memory to capture long term information from a sequence.

Suppose that we have the following RNN model, such that

$$\begin{aligned}\mathbf{h}_t &= \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t + \mathbf{b}_h) \\ z_t &= \text{softmax}(W_{hz}\mathbf{h}_t + \mathbf{b}_z)\end{aligned}$$

where  $z_t$  is the prediction at the time step  $t$ , and  $\tanh(x)$  is defined as

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Derivation from: "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation" by Gang Chen

# Calculations of gradient for RNN

In addition, sharing the weights for any sequential length can generalize the model well. As for sequential labeling, we can use the maximum likelihood to estimate model parameters.

We can minimize the negative log likelihood the objective function:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = - \sum_t y_t \log z_t$$

In the following, we will use notation  $L$  as the objective function for simplicity.

And further we will use  $L(t + 1)$  to indicate the output at the time step  $t + 1$

Let's set  $\alpha_t = W h_{t-1} + b_z$ , and then we have  $z_t = \text{softmax}(\alpha_t)$  (according to  $z_t = \text{softmax}(W h_{t-1} + b_z)$ )

By taking the derivative with respect to  $\alpha_t$ , we get the following:

$$\frac{\partial \mathcal{L}}{\partial \alpha_t} = -(y_t - z_t)$$

Derivation from: "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation" by Gang Chen



# Calculations of gradient for RNN

$$\frac{\partial \mathcal{L}}{\partial \alpha_t} = -(y_t - z_t)$$

$$\frac{\partial \mathcal{L}}{\partial W_{hz}} = \sum_t \frac{\partial \mathcal{L}}{\partial z_t} \frac{\partial z_t}{\partial W_{hz}}$$

$$\frac{\partial \mathcal{L}}{\partial b_z} = \sum_t \frac{\partial \mathcal{L}}{\partial z_t} \frac{\partial z_t}{\partial b_z}$$

$$\frac{\partial \mathcal{L}(t+1)}{\partial W_{hh}} = \frac{\partial \mathcal{L}(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial W_{hh}}$$

$$\frac{\partial \mathcal{L}(t+1)}{\partial W_{hh}} = \frac{\partial \mathcal{L}(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial W_{hh}}$$

$$\frac{\partial \mathcal{L}(t+1)}{\partial W_{hh}} = \sum_{k=1}^t \frac{\partial \mathcal{L}(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial W_{hh}}$$

# Calculations of gradient for RNN

$$\frac{\partial \mathcal{L}(t+1)}{\partial W_{hh}} = \sum_{k=1}^t \frac{\partial \mathcal{L}(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_t}{\partial W_{hh}}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(t+1)}{\partial W_{xh}} &= \frac{\partial \mathcal{L}(t+1)}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial W_{xh}} + \frac{\partial \mathcal{L}(t+1)}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial W_{xh}} \\ &= \frac{\partial \mathcal{L}(t+1)}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial W_{xh}} + \frac{\partial \mathcal{L}(t+1)}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial W_{xh}} \end{aligned}$$

$$\frac{\partial \mathcal{L}(t+1)}{\partial W_{xh}} = \sum_{k=1}^{t+1} \frac{\partial \mathcal{L}(t+1)}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial W_{xh}}$$

$$\frac{\partial \mathcal{L}}{\partial W_{xh}} = \sum_t \sum_{k=1}^{t+1} \frac{\partial \mathcal{L}(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial W_{xh}}$$

However, there are gradient vanishing or exploding problems to RNNs. Notice that  $\partial \mathbf{h}_{t+1} / \partial \mathbf{h}_k$  in above eq. indicates matrix multiplication over the sequence. Because RNNs need to backpropagate gradients over a long sequence (with small values in the matrix multiplication), gradient value will shrink layer over layer, and eventually vanish after a few time steps.

Derivation from: "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation" by Gang Chen

# Presentation Outline

1. References
2. Introduction to Yelp data + Problem
3. Previous Research Methods
4. Our method using RNN
- 5. RNN Implementation**
6. Results
7. Future Applications

# RNN Implementation

Via Python

GOAL - predict a Yelp review's star rating using the text of the review via:

- i.) 1-hot vector
- ii.) 2 layers deep **GRU** recurrent neural network

# Our Architecture:

1.) Input layers - tokenize and create a sequence of one-hot vectors for each word

Embedding process - it's taking the 20,000 word corpus and reducing it to 100-word vectors (groups similar words together)

2.) GRU = sequential layer, number of units is the same as the max sequence length

3.) Dropout - each time you run the model, it will dropout a different 20% to regularize and ensure no bias

4. ) Output layers - our final prediction comes from here, use softmax to model output as probability distribution

Loss function = categorical cross-entropy: used to calculate errors for back propagation

Optimizer = rms prop

Embedding layer: 20,000 neurons

GRU layer 1: 1,000 neurons

GRU layer 2: 100 neurons

Dense layer: 6 neurons

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 1000, 100)	2000100
gru_1 (GRU)	(None, 1000, 100)	60300
dropout_1 (Dropout)	(None, 1000, 100)	0
gru_2 (GRU)	(None, 100)	60300
dropout_2 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 6)	606

Total params: 2,121,306  
 Trainable params: 2,121,306  
 Non-trainable params: 0

None

Train on 180000 samples, validate on 45000 samples

Epoch 1/10

144000/144000 [=====] - 5118s 36ms/step - loss: 0.8457 - acc: 0.6001 - val\_loss: 0.6908 - val\_acc: 0.6844

Epoch 2/10

144000/144000 [=====] - 5078s 35ms/step - loss: 0.6476 - acc: 0.7097 - val\_loss: 0.6611 - val\_acc: 0.7006

Epoch 3/10

144000/144000 [=====] - 5057s 35ms/step - loss: 0.5978 - acc: 0.7343 - val\_loss: 0.6552 - val\_acc: 0.7026

Epoch 4/10

144000/144000 [=====] - 5030s 35ms/step - loss: 0.5622 - acc: 0.7526 - val\_loss: 0.6519 - val\_acc: 0.7061

Epoch 5/10

144000/144000 [=====] - 4979s 35ms/step - loss: 0.5295 - acc: 0.7683 - val\_loss: 0.6478 - val\_acc: 0.7118

Epoch 6/10

144000/144000 [=====] - 5037s 35ms/step - loss: 0.4966 - acc: 0.7852 - val\_loss: 0.6721 - val\_acc: 0.7033

Epoch 7/10

144000/144000 [=====] - 5155s 36ms/step - loss: 0.4616 - acc: 0.8038 - val\_loss: 0.6968 - val\_acc: 0.7014

Epoch 8/10

78848/144000 [=====>.....] - ETA: 35:20 - loss: 0.4112 - acc: 0.8274

# Results

Goal: increase training accuracy and decrease training loss w/ more epochs

7 epochs gives us:

TRAINING on 180k samples	VALIDATING on 45k samples (the test set)
Accuracy 80.38%	Accuracy 70.14%
Loss .4616	Loss .6968

# Tester app

```
##insert any review

#review = "My recent visit with this location was HORRIBLE....If i could give this Starbucks-10 I WOULD.. I went to order my favorite hibiscus d
review = "This starbucks is just okay. I think the location is too small for the volume of people they get. Since its right across from Columbia
seq=tokenizer.texts_to_sequences([review])
seq_pad = pad_sequences(seq,maxlen=1000)

print("number of stars:").
model.predict_classes(seq_pad)[0]
```

```
number of stars:
3
```



# Future Applications

## Applications with RNN

- Try using LSTM and compare for a higher accuracy
- Expand to a larger dataset with more viewers reviews
- Get more exact or precise user data to better understand the accuracy

# Our Future Plans

Saloni- Work

Susmitha- Work