EEL6935 Course Project
Final Report

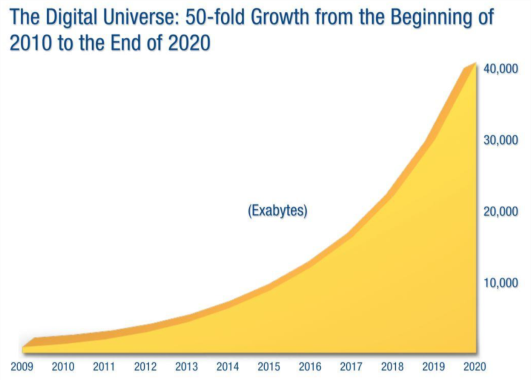
Caleb Bryant[1] Jixin Feng[2]

EEL6935 T21

Department of
[1]Computer & Information Science & Engineering
[2]Electrical & Computer Engineering
University of Florida, Gainesville, FL

Background

- Tremendous volume of unstructured text generated everyday
- 40ZB (40×10^{21}bytes) by year 2020, 50-fold from 2010[1]
- generated from news media, social networks, medical records, business transactions...
- effective processing method is needed



[1]Gantz, J., & Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future, 2007(2012), 1-16.

Sentiment Analysis

Goal: assign sentiment labels to a sentence. $f : \mathcal{D} \rightarrow \mathcal{L}$

- $\mathcal{D} = \{d_0, d_1, \ldots, d_{n-1}\}$ is the set of sentence
- $\mathcal{L} = \{l_0, l_1, \ldots, l_{k-1}\}$ is the set of labels.

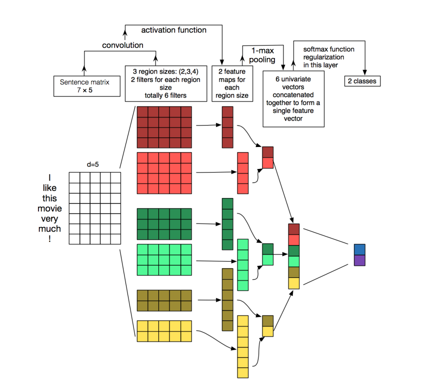
Evaluate using the Stanford Large Movie Review Dataset

- 50,000 highly polar movie reviews
- Each review scored from 1-10
- Conduct both binary and multi-class classification

Movie review is ideal for sentiment analysis

- Most dataset of movie review are already associated with scores
- The scores in dataset are reliable

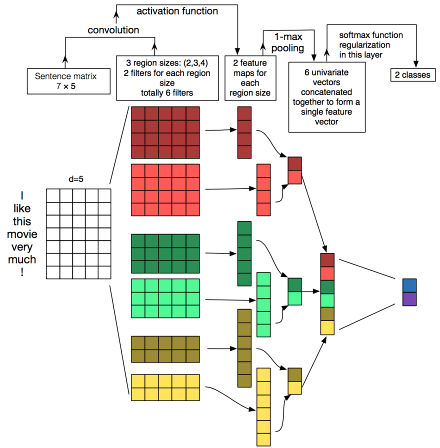
Text Pre-processing



- Tokenization
  - Remove the unnecessary parts of the text
  - Break the text into smaller building blocks like words and phrases
- Filtering
  - emove the part of the text that convey close to zero information
- Lemmatization
  - Groups the words within same role of the text together
- Stemming
  - Find a root of text first and create the tree structure to represent their relationship

[1] figure credit: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Analysis Tools



[1]figure credit:
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

- Historically via naive Bayes, nearest neighbor, decision trees, SVM, etc.
- Build baseline model with Logistic Regression
- Implement CNN model and compare performance
- If time allows, compare with LSTM model too

Text Encoding

- Define the collection of text documents to be

$$\mathcal{D} = \{d_1, d_2, \ldots, d_D\}$$

- Define vocabulary to be

$$\mathcal{V} = \{v_0, v_1, \ldots, v_{m-1}\}$$

- The notation of frequency of a word $v \in \mathcal{V}$ occurred in document $d \in \mathcal{D}$ is $f_d(v)$, hence a document can be represented as a vector

$$\bar{d} = \{f_d(v_1), f_d(v_2), \ldots\}$$

- Define total number of documents $d \in \mathcal{D}$ containing the word w is represented as

$$f_{\mathcal{D}}(v)$$

Text Encoding

Documents and words can also be assigned with other metrics

- Boolean weight

$$\omega_{ij} = \begin{cases} 1 & v_i \in d_j \\ 0 & v_i \notin d_j \end{cases}$$

- Term Frequency-inverse Document Frequency (TF-IDF)

$$q(v) = f_d(v) \log \frac{|\mathcal{D}|}{f_{\mathcal{D}}(v)}$$

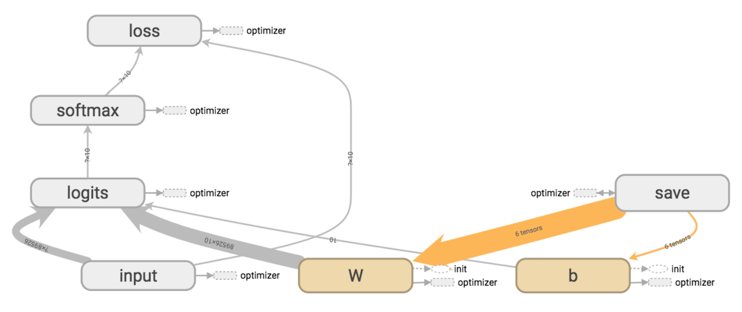
- With these weight metircs, text/document can be represented by a vector

$$w(d) = (w(d, v_1), w(d, v_2), \ldots)$$

Backend: Logistic Regression

- $\hat{y} = \sigma(S(x_i^T W + b))$
- $S(x) = \frac{1}{1+e^{-x}}$
- W is a $N \times C$ weight matrix
- N is number of words, C is number of class
- $\sigma(x)_j = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_i}}$
- b is a $C \times 1$ bias

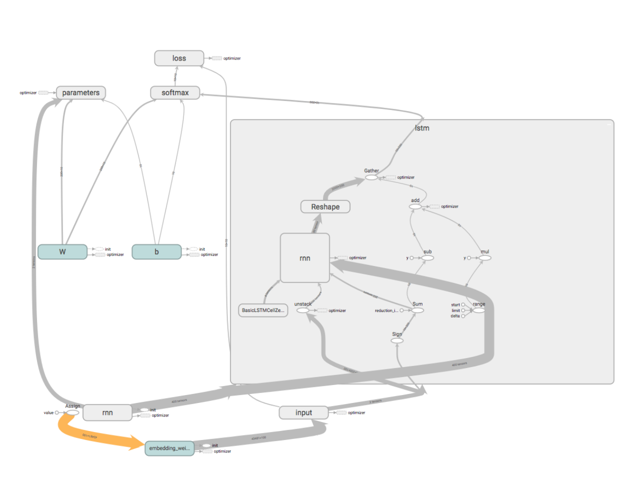
Backend: Logistic Regression



- Multiply input vectors by W, add b
- Convert output into probabilities with Softmax
- pick $i = argmax(\hat{y}_i)$
- Train the model using Adam Optimizer
- Common baseline model and simple to implement

Backend: LSTM Model

- Originally proposed in 1997
- Usually serves as a building block of larger RNN layers
- Build to prevent premature "memory loss" and gradient vanishing



Backend: LSTM Model

Given a word sequence $S = \{v_0, v_1, \ldots, v_{l-1}\}$ with length l, the states of LSTM are updated as:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ c_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} S[h_{t-1}, x_t]$$

- $c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t$
- $h_t = o_t \circ \tanh c_t$

Backend: LSTM Model

To build our LSTM model:

- Define a dictionary
$$D = [w_1, w_2 ... w_N]$$
to contain the set of all words representable by our LSTM network
- The i-th input to our LSTM is a vector
$$v_i = [\pi_{i1}, \pi_{i2} ... \pi_{iL}]$$
corresponding to a movie review r_i of length L, where π_{ij} is the dictionary index of the j-th word of r
- $D = [\text{cat}, \text{dog}, \text{meow}]$
$$r = [\text{dog}, \text{dog}, \text{cat}, \text{meow}] \Rightarrow v = [1, 1, 0, 2]$$

Backend: LSTM Model

In order to help the LSTM learn how to interpret the words in a movie review

- Each sentence's words are embedded inside a weight matrix W_e with dimensions $N \times E$
- E is a hyperparameter known as the embedding dimension and often ranges between 50 and 300

Backend: LSTM Model

After initializing the hidden state to zeros at the beginning of each batch, sequences of embeddings

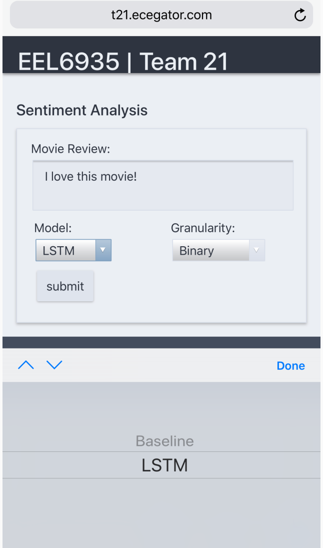
$$s_i = [e_{i1}, e_{i2}, ..e_{iL}]$$

are then sequentially fed into the LSTM layer of our network

- Each e_{ij} is the embedding vector for the word w_{ij} looked-up according to the input index π_{ij}
- Only the final output state of the LSTM is used for prediction
- It is fed into a fully-connected softmax layer

Front End: Flask Web App

- Build a web interface based on Flask
- Allow model selection
- Accept text input and sentiment analysis output
- Hosted on `t21.ecegator.com`



Simulation Platform Specs

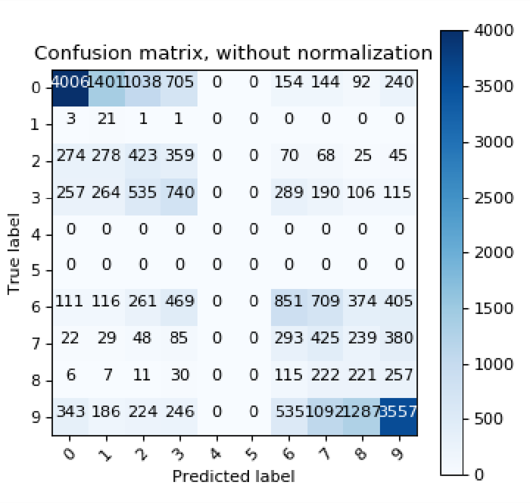
- Propose using the Stanford Large Movie Review Dataset
  - 50,000 highly polar movie reviews
- Score each movie review as either positive or negative
- Use a 60/20/20 split for training, development, and testing
- Code in Python with Tensorflow/Scikit-Learn
- Document composed in LaTeX
- simulations will be conducted on computers with
  - 3.1GHz Intel Core i7 CPU with 8MB cache
  - nVidia GTX 1050 GPU with 4GB Memory
  - 16GB RAM
  - Ubuntu 14.04 LTS

Result

Table 1: System Performance

Method	Epochs	Binomial Training	Binomial Testing	Multinomial Training	Multinomial Testing
scikit-learn LR	N/A	0.9981	**0.8697**	N/A	N/A
tensorflow LR	20	0.8670	0.8583	0.9982	0.3734
larger LSTM	8	N/A	N/A	0.6693	0.3657
smaller LSTM	2	N/A	0.8507	0.5622	**0.4098**

Result



- Logistic Regression quickly converged to the global minimum
- No worry about escaping local optima
- LSTM is very slow to train
- Need even more computational resource to achieve better accuracy

Team Coordination

- Backend: C
- Frontend: J
- Connecting: C
- Web hosting and maintaining: J
- Report writing and making slides: J

The detailed record of contribution history is maintained in the project Git repository `https://github.com/ufjfeng/EEL6935-Course-Project` hosted on GitHub.

[1] C: Caleb Bryant, J: Jixin Feng

Thank You!

Questions?

https://github.com/ufjfeng/EEL6935-Course-Project