Language Models for US Presidential Candidates

CS 229, Autumn 2016; Category: Natural Language

FNU Budianto (budi71@stanford.edu), Jeff Nainaparampil (jeffnain@stanford.edu), Shruti Murali (shru@stanford.edu)

Web UI:
http://tiny.cc/cs229-pclm

Abstract

The 2016 Presidential Elections had proved to be very contentious, with heated language and attacks from both sides. We train language models to better understand the structure of the candidate's spoken words. Specifically, we utilize two types of recurrent neural networks: the Gated Recurrent Unit (GRU) and Long Short-Term Unit (LSTM), and a Hidden Markov Model (HMM). Results indicate the LSTM is the best in terms of perplexity and author classification on the test set, with test perplexity of 37.476 and classification test error of 0.1039.

Dataset

Our project uses two distinct datasets: the v1 dataset are the transcripts from the primary and presidential debates from Clinton and Trump, and the v2 dataset is created by adding tweets from the two candidates to the v1 dataset. Both debates and tweets data are taken from kaggle.com. Lastly, a supplementary Penn Tree Bank corpus of words to is used to generate a pretrained embedding (results not shown in poster). Each dataset is split into 80% for the training set, 5% for validation, and 15% for the test set. The least frequent words are replaced with unknown-token.

Number of Sentences	v1	v2
Clinton	4110	10182
Trump	5330	12066
Total	9440	22248

Number of Words with Punct.	v1	v2
Clinton	86719	163633
Trump	77125	154714
Total	163844	318347

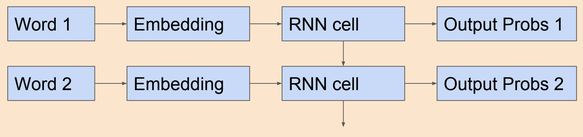
	v1	v2
Number of Unique Words with Punct.	5848	11555
Vocabulary Size	5500	9000

Features and Model: RNN - GRU

An individual unit of a GRU is governed by the following functions, where $h_j^{(t)}$ is the j^{th} output and $x_j^{(t)}$ the j^{th} input at time t:

$$z_j^{(t)} = \sigma\left(W_{z_j} \cdot [h^{(t-1)}, x^{(t)}] + b_{z_j}\right) \qquad \tilde{h}_j^{(t)} = \tanh(W_{\tilde{h}_j} \cdot [r^{(t)} \cdot h^{(t-1)}, x^{(t)}] + b_{\tilde{h}_j})$$
$$r_j^{(t)} = \sigma\left(W_{r_j} \cdot [h^{(t-1)}, x^{(t)} + b_{r_j}\right) \qquad h_j^{(t)} = (1 - z_j^{(t)})\tilde{h}_j^{(t)} + z_j^{(t)} h_j^{(t-1)}$$

We modified an open-source Tensorflow script for the RNN implementations. Included is a word embedding, which maps the one-hot vector of a word to a different vector, where similar words are clustered together. The output of the GRU is affinely mapped to a dimension equal to the vocabulary size and softmax is performed to estimate probability. Training is performed in mini-batches of sequences of words via gradient descent. The GRU uses the same parameters as LSTM.



Features and Model: RNN - LSTM

The LSTM relies a similar recurrent network model as the GRU, but with a cell state $c_j^{(t)}$ separate from output of the model $h_j^{(t)}$. Specifically, the gating functions of the LSTM are the following:

$$I_i^{(t)} = W_{I_i} \cdot [x^{(t)}, h^{(t-1)}] + b_{I_i} \qquad F_i^{(t)} = W_{F_i} \cdot [x^{(t)}, h^{(t-1)}] + b_{F_i}$$
$$J_i^{(t)} = W_{J_i} \cdot [x^{(t)}, h^{(t-1)}] + b_{J_i} \qquad O_i^{(t)} = W_{O_i} \cdot [x^{(t)}, h^{(t-1)}] + b_{O_i}$$

From the gating functions, the next cell states and current outputs can be calculated.

$$c_i^{(t)} = c_i^{(t-1)}\sigma(F_i^{(t)}) + \sigma(I_i^{(t)})\tanh(J_i^{(t)}) \qquad h_i^{(t)} = \tanh(c_i^{(t)})\sigma(O_i^{(t)})$$

From performing a parameters search, we find that the hyperparameters that led the best results are:

- RNN layers = 1
- learning rate = 1.0 with a decay factor of 0.8
- max gradient norm = 5
- mini-batch size = 20
- backpropagation steps = 35
- dropout keep probability = 0.35
- embedding vector size = 650

Features and Model: HMM

We define the hidden states using the part-of-speech (pos) tagger in NLTK toolkit. The parameters {transition distribution, emission distribution, initial state distribution}, are initialized empirically. Laplacian smoothing is applied on the parameters, and we then use the Baum Welch algorithm to train the model. After the model is trained, we utilize Matlab's hmmdecode for perplexity computations. The basic equations that govern hidden markov models are as follows:

$$\alpha_k(x_k) = p_{y|x}(y_k, x_k) \sum_{x_{k-1}} \alpha_{k-1}(x_{k-1}) W(x_{k-1}|x_k)$$
$$\beta_k(x_k) = \sum_{x_{k+1}} \beta_{k+1}(x_{k+1}) p_{y|x}(y_{k+1}, x_{k+1}) W(x_k|x_{k+1})$$

Number of Hidden States	v1	v2
With Punct.	42	44
w/o Punct.	35	35

Sentence Generation and Classification + Naive Bayes and SVM

Sentence generation is performed by doing multinomial sampling using the output probabilities as the probability distribution.

Classification for RNNs and HMM is performed by calculating:

$$\arg\max_{\text{author}} P(\text{author}|\text{sentence}) = \arg\max_{\text{author}} P(\text{sentence}|\text{author})P(\text{author})$$

$$P(\text{sentence}|\text{author}) = P(w_1|s)P(w_2|s,w_1)\dots P(e|s,w_1,\dots,w_n)$$

In RNN models (LSTM and GRU), $P(\text{next word}|\text{previous words})$ comes from the output probabilities of the model, whereas in the HMM, $P(\text{sentence}|\text{author})$ is calculated using the forward algorithm. $P(\text{author})$ is calculated by counting the number of sentences authored divided by total number of sentences in the training set.

The Naive Bayes uses the multinomial event model with Laplacian smoothing and the SVM uses the Gaussian kernel with a similar event model (we use the same overall specifications as demonstrated in Problem Set #2).

Results

Perplexity for dataset v1 (debates only)

Model	With punctuation				Without punctuation			
	Trump		Clinton		Trump		Clinton	
	Train	Test	Train	Test	Train	Test	Train	Test
LSTM	20.063	37.476	27.037	53.893	28.598	60.724	35.349	80.860
GRU	19.263	41.304	25.128	58.388	28.019	68.633	33.362	87.740
HMM	56.039	98.970	80.485	143.695	86.330	175.518	114.718	229.559

Perplexity for dataset v2 (debates + tweets)

Model	With punctuation				Without punctuation			
	Trump		Clinton		Trump		Clinton	
	Train	Test	Train	Test	Train	Test	Train	Test
LSTM	29.235	51.444	30.923	56.245	40.788	79.275	41.257	84.494
GRU	28.370	55.833	30.142	60.337	40.115	85.277	39.962	89.426
HMM	93.243	151.390	104.816	165.186	136.888	231.379	151.323	242.429

Sentence Generation

Using LSTM and Dataset v2 with punctuation

Clinton: "we are also going to vote for the global economy again …"

Trump: "we will make america great again."

Author Classification Test Error

Model	with punctuation		w/o punct.	
	v1	v2	v1	v2
LSTM	0.1253	0.1039	0.1279	0.1252
GRU	0.1325	0.1093	0.1403	0.1228
Naive Bayes	0.1436	0.1515	0.1495	0.1755
HMM	0.2898	0.1881	0.3153	0.1833
SVM	0.2931	0.3282	0.3035	0.3591

Perplexity = $\exp\left(\frac{1}{N}\sum_{i=1}^{N} \log \hat{P}(w_i|w_1, w_2, ..., w_{i-1})\right)$

Classification Error =
#correct_prediction / #sentences

Discussion

- RNN models perform better than HMM in terms of perplexity, and LSTM is slightly better than GRU. This might be due to RNN models making decisions based on longer histories of words preceding the current word.
- In the author classification task, RNN models perform better than HMM, Naive Bayes, and SVM. Again, LSTM has the lowest test error, but GRU comes in a close second.
- Using pre-trained embedding does not give good results, probably because it is trained using a different dataset (Penn Tree Bank corpus) and isn't improved during training.
- Across all models, perplexity and classification error is better on the dataset with punctuation than on the dataset without punctuation.

Future Work

- Continue to add to the Clinton and Trump datasets.
- Try different types of LSTM (e.g. LSTM with peephole)
- Further hyperparameter (embedding vector size, dropout keep probability, etc) tuning.
- Improve the web user interface (e.g. add HMM).

References

- "Recurrent neural networks."https://www.tensorflow.org/versions/r0.11/tutorials/recurrent/index.html.[Online; accessed 12-November-2016].
- S. Hochreiter and J. Schmidhuber, "Long short-term memory,"Neural Comput., vol. 9, pp. 1735–1780, Nov. 1997.
- C. Olah, "Understanding lstm networks."http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Aug.2015. [Online; accessed 12-November-2016].
- K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,"ArXiv e-prints, June 2014.
- D. Ramage, "Hidden markov models fundamentals."http://cs229.stanford.edu/section/cs229-hmm.pdf, Dec.2007. [Online; accessed 21-November-2016].