Gated Recurrent Units for Airline Sentiment Analysis of Twitter Data

Yixin Tang
Department of Statistics
Stanford University
Stanford CA, 94305
yixint@stanford.edu

Jiada Liu
Department of Statistics
Stanford University
Stanford CA, 94305
jiada@stanford.edu

Abstract

We explore the use of a bi-directional gated recurrent unit (GRU) network for sentiment analysis of Twitter data directed at U.S airlines. The performance of the GRU is then compared with a convolutional neural network (CNN) and also the comparatively simpler vanilla single layer network. We wish to not only identify which model has the highest test accuracy, but also its precision, recall rates as well as its robustness to unbalanced classes in the training set. We also discuss the common practice of re-weighting data for unbalanced classes and its effect on a model's prediction from a Bayesian viewpoint. Indeed, as it turns out, there is a balance to be found between using local information, that is information contained in each tweet and global information, the information present in the entire training set or test set when making predictions. We also implement a Long Short Term Memory (LSTM) network for text generation of some positive or negative airline tweets and the relatively coherent text demonstrate the power of these networks.

1 Introduction

In recent years, Twitter has become the de facto online customer service platform. Thus, a company's image on Twitter is of central importance and this is especially true for the airline industry given that many tweets are travel related in nature. Indeed, for the airline industry, research has shown that responding to tweets has revenue generating potential, drives higher satisfaction than other customer service channels, and perhaps most importantly, satisfied Twitter users spread the word. In this project, we wish to give airlines a quick snapshot of their image on twitter based on the sentiment of tweets sent to them. Our goal is to classify each tweet into three classes; negative, neutral and positive. The dataset we use contain 14,605 real tweets directed at various US airlines such as United, Virgin America and American Airlines. We train our own word vector through a skip-gram model, initializing with exiting GloVe model by Socher et al. (2014). Then we feed our tweets into a bi-directional GRU, tweet by tweet, where each tweet is concatenated from its individual word vectors. We also try some tree based parser on our trained word vectors in addition to simple concatenation. The performance of the GRU model is then compared to that of a model where each tweet's prediction in the test set cannot influence one another, for example, a CNN or RNN where the inputs are word vectors instead. Previous approaches has mostly relied on word clustering and using the corresponding word frequency as features for prediction. We improve on this by capturing context, providing higher accuracy.

2 Related Work

Our work draws from three areas of NLP, sentiment analysis, semantic vector spaces and deep learning. Twitter user's frequent use of emoticons, abbreviations and slang require custom trained word vectors. Emoticons can contain powerful emotions, and indeed sentiment analysis on Twitter data often use custom dictionaries linking emoticons and slang to word vectors (Agarwal et al. 2011). Indeed, GloVe models for twitter data needs to be custom trained

(Pennington et al. 2014) and it is this model that we initialize our own embedding matrix with. As we soon see, without training our own word vectors, we cannot achieve a very high accuracy. Recent advances in deep learning with compositional models such as recursive neural networks (RNN) (Socher et al., 2011b), matrix-vector RNNs (Socher et al., 2012) help us to capture context and takes advantage of the nested hierarchy present in many sentiment tasks. The advent of parse trees for combining word vectors into phrases is of great use to us, as this allows us to make a network where tweets can influence each other (Chen and Manning, 2014). The model we use is a bi-directional gated recurrent unit network (Cho et al. 2014; Chung et al. 2014), which uses a gating unit for controlling flow from pairs of recurrent layers. An LSTM (Hochreiter and Schmidhuber 1997) model is used for tweet generation, and we find that the gate units used to control error flow leads to impressive generation abilities. The unbalanced classes in our training set can be thought of in the framework of compound decision theory (Robbins 1956), and is closely related to empirical Bayesian theory also pioneered by Robbins.

3 Technical Approach and Models

The first task revolves on training our word vectors. We first do minimal preprocessing of the data, such as making words case-insensitive and removing repeated punctuation. Afterwards, we initialize our embedding matrix and GloVe model for twitter data, and let this embedding matrix update itself as our GRU model run. We also train a skip-gram model for our word vectors but either way, we let our embedding change through the training of the network.

3.1 Single Layer Baseline

We implement a single layer neural network as a baseline first. Here, we train our own word vectors using skip-gram and these word vector stays fixed throughout, unlike the GRU network. The precise equations are
$h = \sigma(xW_1 + b_1), \hat{y} = \text{softmax}(hW_2 + b_2)$.

3.2 Bi-directional GRU

Once we have an embedding, our main approach will be a bi-directional GRU where we concatenate the words of a tweet into one word vector. Placeholders are added as necessary to ensure that all tweets have a common length. Specifically, the equations for this network is as follows.

Right direction $\overrightarrow{h}_t^{(i)}$:

$$\overrightarrow{z}_t^{(i)} = \sigma(\overrightarrow{W}_{(i)}^{(z)} x_t^i + \overrightarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)})$$

$$\overrightarrow{r}_t^{(i)} = \sigma(\overrightarrow{W}_{(i)}^{(r)} x_t^i + \overrightarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)})$$

$$\overrightarrow{\tilde{h}}_t^{(i)} = tanh(\overrightarrow{W}_{(i)} x_t + r_t \circ \overrightarrow{U}_{(i)} h_{t-1})$$

$$\overrightarrow{h}_t^{(i)} = z_t^{(i)} \circ h_{t-1}^{(i)} + (1 - z_t^{(i)}) \circ \tilde{h}_t^{(i)}$$

Left direction $\overleftarrow{h}_t^{(i)}$:

$$\overleftarrow{z}_t^{(i)} = \sigma(\overleftarrow{W}_{(i)}^{(z)} x_t^i + \overleftarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)})$$

$$\overleftarrow{r}_t^{(i)} = \sigma(\overleftarrow{W}_{(i)}^{(r)} x_t^i + \overleftarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)})$$

$$\overrightarrow{\tilde{h}}_t^{(i)} = tanh(\overleftarrow{W}_{(i)} x_t + r_t \circ \overleftarrow{U}_{(i)} h_{t-1})$$

$$\overleftarrow{h}_t^{(i)} = z_t^{(i)} \circ h_{t-1}^{(i)} + (1 - z_t^{(i)}) \circ \tilde{h}_t^{(i)}$$

Output

$$y_t = softmax(U[\overrightarrow{h}_t^{(top)}; \overleftarrow{h}_t^{(top)}] + c)$$

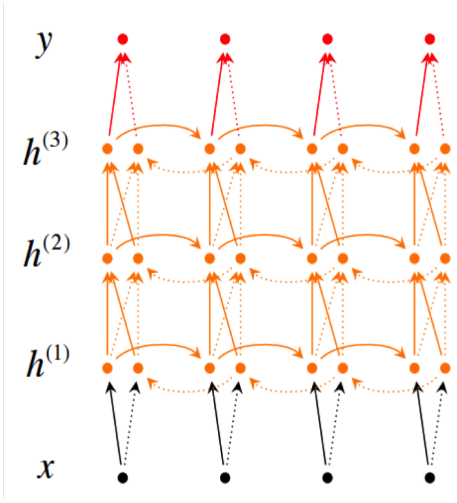


Figure 1: A GRU network on a high level

where x_{t-1} denote input, r_{t-1} is reset gate, z_{t-1} is the update gate, $\tilde{h}_{t-1}$ is the reset memory, and h_t is the new memory.
Observe that by feeding a tweet into each x, we allow each batch of tweets to influence each other's prediction, an idea that may seem counterintuitive at first because each tweet is assumed to be independent from one another. However, our data is sorted in a way such that tweets common to an airline are batched together, that is the first 1000 tweets may all be directed at Virgin America, the next 1500 at American Airlines and so on. Due to this structure, we are motivated to use a network with bi-directional flow. As we will explore later, this is part of a larger framework in empirical Bayes theory, where global information can be of help to local prediction, even if each sample is assumed to be independent of one another.

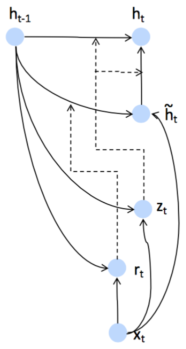
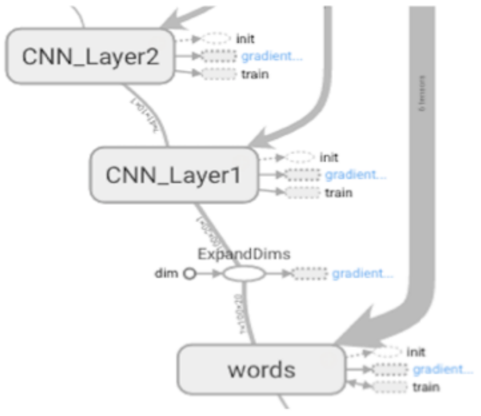


Figure 2: A GRU network on a low level

3.3 Convolutional Neural Network

Similar to our baseline, CNN also does not change its embedding matrix once initialized. We train our word vector in the same way as before. Our convolutional neural network has two convolutional layers, one fully connected layer and two pooling layers to make the dimensions match. The equations for CNN is as follows, along with a diagram courtesy of TensorBoard.

$$
\begin{aligned}
\boldsymbol{x}_{1:n} &= \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus \ldots \oplus \boldsymbol{x}_n \\
&\{\boldsymbol{x}_{1:h}, \boldsymbol{x}_{2:h+1}, \ldots, \boldsymbol{x}_{n-h+1:n}\} \\
c_i &= f(\boldsymbol{w}^T \boldsymbol{x}_{i:i+h-1} + b) \\
\boldsymbol{c} &= [c_1, c_2, \ldots, c_{n-h+1}] \\
\hat{y} &= \text{softmax}(W^{(S)}(r * z) + b)
\end{aligned}
$$



where each $\boldsymbol{x}_i, i \in \{1, 2, ..., n\}$ is the vector of each word, $\boldsymbol{x}_{1:n}$ denotes the concatenation of the first n word vectors, $\boldsymbol{w}$ and b are the parameters of filters which have the inner product with the corpus of training data. After the final *softmax* function, we get the probability of each class and use cross entropy to train the network. Note that unlike the previous GRU, the input to our CNN are word vectors, and each prediction is made independent of one another, and the connection between layers is contained within each tweet.

3.4 LSTM for Text Generation

Finally, we also implement a LSTM to generate text. This method selects the most probable word each time, depending on which class we are in. The following equations govern our training.

$$
\begin{aligned}
z_t &= \sigma(W_z[h_{t-1}, x_t]) \\
r_t &= \sigma(W_r[h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W[r_t * h_{t-1}, x_t]) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

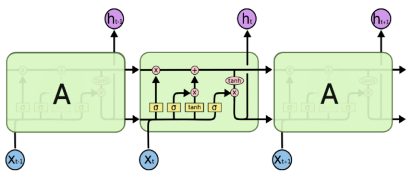


Figure 3: The LSTM model

where $z_t, r_t, \tilde{h}_t, h_t$ denotes the update gate, reset gate, reset memory and new memory respectively. The following is a real tweet generated by our model and we give its process below.

$$< start > \rightarrow h_1 = (p_1.p_2, ..., p_{|V|} \rightarrow \textit{Definitely!} \rightarrow h_2 = (p_1.p_2, ..., p_{|V|} \rightarrow \textit{Customer} \rightarrow h_3 = (p_1.p_2, ..., p_{|V|} \rightarrow \textit{for}$$
$$\rightarrow h_4 = (p_1.p_2, ..., p_{|V|} \rightarrow \textit{ever!} \rightarrow < end >$$

4 Experiments and Findings

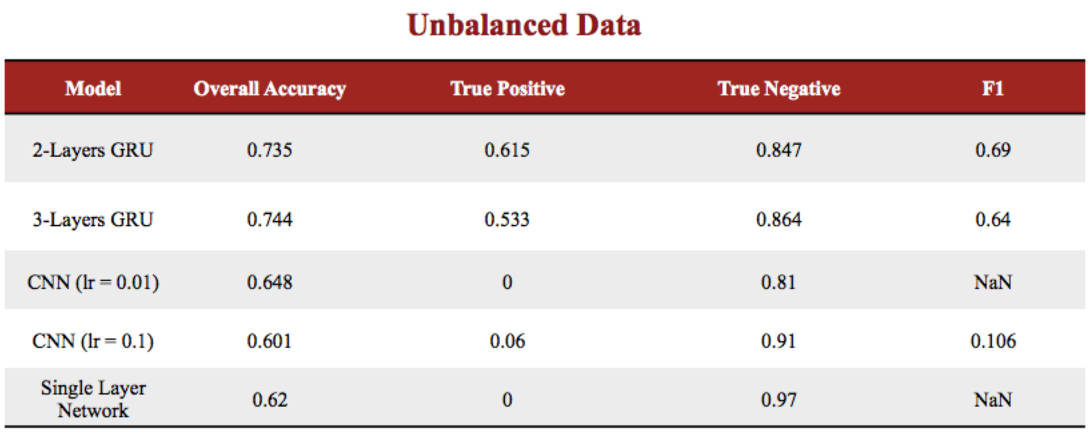
We judge our classification models by overall accuracy, precision, recall and F1 scores. All of our models are trained using cross entropy loss. Before doing so, we first digress into a discussion about our dataset.

4.1 Data

Our dataset consists of 14605 real tweets directed at various US airlines. Each tweet is given a label; negative, neutral or positive manually assigned by a human. Since travel related tweets are generally salty in nature, most tweets are negative. We split our dataset into a training set and test set by randomly picking 10% of tweets from our dataset to be the test set. Previous approaches to our problem has primarily focused on word clustering, where each word is assigned a polarity score and the word frequencies are then used as features. The picture below illustrates the unbalanced nature of our training set and also the word clouds that these previous approach are based on. These methods fail to capture context and as such is often limited to polarizing key words like ”cancelled”, ”delayed” for negative sentiment and ”thanks” or ”awesome” for positive sentiments.

4.2 Results with Unbalanced Data

The result of the GRU and CNN model is summarized in the table below.



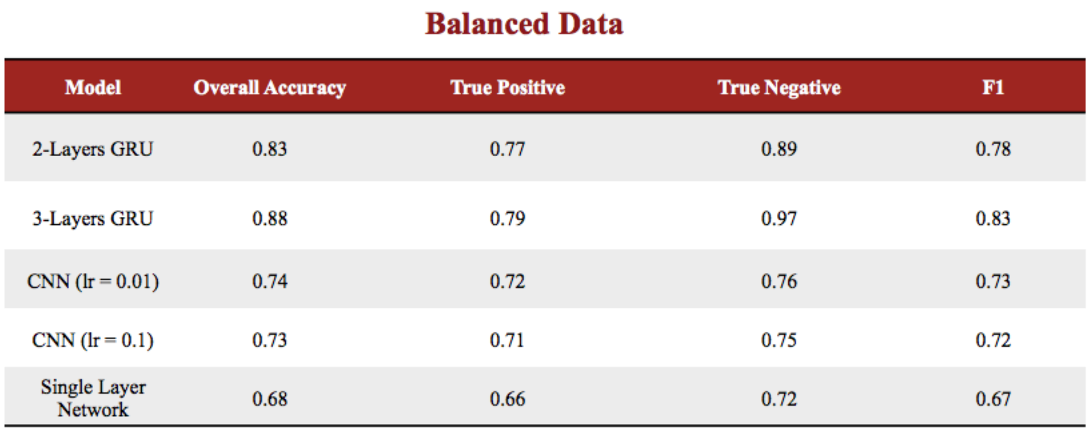
Unbalanced Data

Model	Overall Accuracy	True Positive	True Negative	F1
2-Layers GRU	0.735	0.615	0.847	0.69
3-Layers GRU	0.744	0.533	0.864	0.64
CNN (lr = 0.01)	0.648	0	0.81	NaN
CNN (lr = 0.1)	0.601	0.06	0.91	0.106
Single Layer Network	0.62	0	0.97	NaN

Observe that the 3-Layer GRU has the highest accuracy in the test set, but its true positive rate is much lower than the 2-Layer GRU and as such has a lower F1 score. As our GRU model becomes more complex, it picks up the prior information that the majority of tweets are negative in nature and uses this knowledge to improve its negative class prediction, while ignoring the positive class. We see that the CNN and one layer neural network take this strategy to a new extreme, heavily biasing the majority class. The reader may conclude that the GRU is more robust to unbalanced data than the CNN, but this is also due to our model architecture. Recall that the inputs to the GRU are tweets or concatenated word vectors, and so the model makes decision globally, while the inputs to the CNN are word vectors, and so the model makes decision one at a time. The latter tend to escalate the effect of the negative prior, since it has no knowledge about its other predictions. In fact, because our GRU makes its decision globally, the distribution of its predictions closely matches that of the training set.

4.3 Results with Balanced Data

The result of the GRU, CNN and vanilla single layer model is summarized below. Unsurprisingly, they are much better than the unbalanced version. We balanced our data by adding a small amount of Gaussian noise to each positive sample, to bring up the number of positive and neutral tweets to comparable numbers of negative tweet. This method is identical to giving positive or neutral samples more weight in the training step.



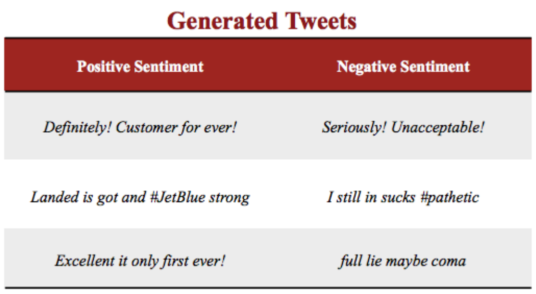
Balanced Data

Model	Overall Accuracy	True Positive	True Negative	F1
2-Layers GRU	0.83	0.77	0.89	0.78
3-Layers GRU	0.88	0.79	0.97	0.83
CNN (lr = 0.01)	0.74	0.72	0.76	0.73
CNN (lr = 0.1)	0.73	0.71	0.75	0.72
Single Layer Network	0.68	0.66	0.72	0.67

The improvement comes from a stronger reliance on local information. Balanced data makes the prior uninformative, which helps CNN and one layer network to make a better prediction. Nonetheless, GRU is still the best model but this time, the 3-Layer GRU wins on both overall accuracy and F1 score. Recall that for the unbalanced case, the 3-Layer GRU also achieved a higher accuracy, but gained its accuracy biasing its prediction to the majority class and thus, losing to the 2-Layer GRU on F1 score. But here, it is better both in overall accuracy and F1 score because the prior is now uninformative and so it's fitting on local information, or the actual tweet itself. We see that GRU still beat out traditional CNN, indicating that global information is still informative, and subsumes more than a simple prior distribution of classes.

4.4 Text Generation

Below are some samples of generated text by our LSTM.



Generated Tweets

Positive Sentiment	Negative Sentiment
Definitely! Customer for ever!	*Seriously! Unacceptable!*
Landed is got and #JetBlue strong	*I still in sucks #pathetic*
Excellent it only first ever!	*full lie maybe coma*

5 A Bayesian approach to unbalanced training classes

The following example from empirical bayesian statistics give an example of how global decisions can perform better than local decisions, even under independence assumptions.

5.1 Coin tossing example

Suppose there are two type of coins c_A, c_B with $\mathbb{P}(H) = \frac{2}{3}$ and $\mathbb{P}(H) = \frac{1}{3}$ respectively. If you observe the outcome of one toss, and wish to guess which type of coin was tossed, it's clear that the optimal decision is to guess c_A if we observe H and c_B if we observe T. This incurs an error of $1/3$.
Now suppose we repeat this N times, where each toss is independent of one another. One may guess that the same local strategy above is optimal, but this is not so. Let us now dispel this notion.
Let ε be the unknown proportion of c_A coins and let δ be the fraction of observed heads. For large N, we have $\delta \approx \varepsilon(2/3) + (1-\varepsilon)(1/3) = (1+\varepsilon)/3$ so $\varepsilon = 3\delta - 1$. Thus, if $\delta < 4/9$, then with high probability, $\varepsilon < 1/3$. Hence, a global decision to guess c_B coin for every coin incurs an error of $\varepsilon < 1/3$, smaller than the 1/3 error incurred by local decisions. Similarly, if $\delta > 5/9$, then with high probability, $\varepsilon > 2/3$ and so a global decision to guess all c_A coins incur an error of $1 - \varepsilon < 1/3$. If the proportion of heads is $4/9 \leq \delta \leq 5/9$ then the optimal decision is the local decision as discussed above. Thus, with global decisions taken into account, the error we incur is actually $\min\{1/3, \varepsilon, 1-\varepsilon\}$ and for some ε, this can be a major improvement over local decisions.

5.2 Unbalanced training classes and global decisions

Models such as CNN and RNN often overwhelmingly predict the majority class if the training set is sufficiently unbalanced. One common way to rectify this situation is to increase the weight of the minority class, but such strategy seem to violate one of the most fundamental assumptions of statistics, that we train and test on the same distribution. Neural networks however, are seemingly immune to this gross violation and the authors suspect this is due to its many parameters and its complex nature. This suggest that models like CNN and RNN relies heavily on local information to do well, and indeed rebalancing data essentially eliminate the information contained in the global prior . This makes intuitive sense since when someone says 'I am mad", the sentiment should be clear from this information alone without relying on what this person has said before. Nonetheless, taking into account of global information can help our predictions, perhaps if the person is known to say 'I am mad" very often, one may change this to a neutral sentiment. In an extreme case, global information can completely override local information, as demonstrated by the coin tossing example above and our CNN on unbalanced data. Usually though, there is a balance to be found and in our dataset, global information subsumes correlation between airlines, time and other features that humans cannot pick out which the GRU model may have used in obtaining its high accuracy.

6 Conclusion and Future Directions

We have implemented a bi-directional GRU, CNN and a single layer network for our airline sentiment analysis. Our results show that bi-directional GRU perform the best and this is most likely due to a combination of the forget gates as well as the global nature in which the data were fed. We found that CNN and single layer network requires balanced data to perform well, and that they tend to make local decisions, ideal for a predicting streams of data. Compared to our GRU, the accuracy and F1 score are both lower even in both balanced and unbalanced cases so it is clear that the bi-directional GRU is the optimal model for this dataset. One interesting aspect is that for unbalanced data, a 2-Layer GRU has a higher F1 score than a 3-Layer GRU because a 3-Layer GRU will encourage the model to focus on the majority class, leading to a higher accuracy but lower F1 score due to its neglect for the minority class. With balanced data however, the 3-Layer GRU wins outright. We have also established the networks behavior when trained on balanced and unbalanced data. The GRU are more robust to an unbalanced training set due to its global predictions, but CNN and RNN tend to choose the global prior over its local information, causing it to favor the majority class. Once the training set is balanced, the results are much better and more sophisticated models lead to better results despite violating a key assumption in statistics of training and testing on the same distribution. We cannot however, conclude that global information is useless, as indicated by the superior performance of GRU, and rush to balance every dataset we have, but instead, shows a need for our network to adequately use both type of information without over-reliance on either one. On the other hand, local information reliance is what makes CNN so adaptable to many different techniques where we could very well, be training and testing on different distributions.

Our aim was to provide airlines a quick snapshot of their image on twitter. A natural next step is to automatically filter tweet for airlines that require a response. Airlines such as Southwest has replied to some customer's urgent tweets in 2 minutes or less, often garnering praise and we wish to help them do more of that by automatically deciding which tweets are important and in need of a response.

Acknowledgments

We extend our warmest thanks to the CS224D teaching staff for their advice and support during office hours as well as Crowdflower for providing the dataset.

References

[1]Chung, Junyoung, et al. ”Empirical evaluation of gated recurrent neural networks on sequence modeling.” arXiv preprint arXiv:1412.3555 (2014).

[2] Socher, Richard, et al. “Recursive deep models for semantic compositionality over a sentiment treebank.” Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[3]Chung, Junyoung, et al. ”Gated feedback recurrent neural networks.” arXiv preprint arXiv:1502.02367 (2015).

[4] Agarwal, Apoorv, et al. “Sentiment analysis of twitter data.” Proceedings of the workshop on languages in social media. Association for Computational Linguistics, 2011.

[5]Hochreiter, Sepp, and Jrgen Schmidhuber. ”Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

[6] Airline Sentiment Data Analysis. https://www.kaggle.com/solegalli/d/crowdflower/twitter-airline-sentiment/airline-sentiment-part-2

[7] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. “Glove: Global Vectors for Word Representation.” EMNLP. Vol. 14. 2014.

[8] Chen, Danqi, and Christopher D. Manning. ”A Fast and Accurate Dependency Parser using Neural Networks.” EMNLP. 2014.