HW1: Classification

Jiafeng Chen* Yufeng Ling Francisco Rivera*

February 4, 2019

1 Introduction

In this write-up, our main focus is on building classifiers of sentiment in sentences. We implemented naive Bayes unigram classifier (Wang and Manning (2012)), logistic regression over word types, a continuous bag-of-word neural network with embeddings (Mikolov et al. (2013)) and a convolutional neural network (Kim (2014)). In addition, we experimented different ensemble methods and made improvements.

2 Problem Description—Old

In general, homeworks will be specified using informal language. As part of the assignment, we expect you to write-out a definition of the problem and your model in formal language. For this class, we will use the following notation:

- $\boldsymbol{b}, \boldsymbol{m}$; bold letters for vectors.
- $\boldsymbol{B}, \boldsymbol{M}$; bold capital letters for matrices.
- $\mathcal{B}, \mathcal{M}$; script-case for sets.
- b_i, x_i; lower case for scalars or indexing into vectors.

For instance in natural language processing, it is common to use discrete sets like $\mathcal{V}$ for the vocabulary of the language, or $\mathcal{T}$ for a tag set of the language. We might also want one-hot vectors representing words. These will be of the type $\boldsymbol{v} \in \{0,1\}^{|\mathcal{V}|}$. In a note, it is crucial to define the types of all variables that are introduced. The problem description is the right place to do this.

*Equal contribution

3 Problem Description

The focus of the problem set is sentiment classification. We are given *sentences* and aim to classify them as either positive or negative sentiment (a binary classification). These predictions can be made in a probabilistic fashion by writing y_i as the event that the ith sentence is positive sentiment, and calculating $p(y_i)$.

Sentences[1] are themselves sequences of words $w \in \mathcal{V}$ in some vocabulary $\mathcal{V}$. In particular, there will be two special words, w_{unk}, $w_{\text{pad}} \in \mathcal{V}$ which represent an unknown word and a padding unit respectively; we address their significance later. A particular sequence of words $w_1, \ldots, w_n$ need not have a fixed length, which varies across sentences.

We represent words in two main ways. The first is as a one-hot encoded vector $\boldsymbol{v} \in \{0,1\}^{|\mathcal{V}|}$. This representation is useful for the Logistic Regression model. Alternatively, we can use a dense embedding. That is, each word gets assigned a vector $\boldsymbol{v} \in \mathbb{R}^d$ where d is the embedding dimension. We use Mikolov et al. (2013)'s pre-trained word embeddings ($d = 300$) for the Continuous Bag of Words and Convolutional Neural Network models.

4 Model and Algorithms

4.1 Naive Bayes

As we alluded to earlier, we can treat our prediction problem as probabilistic. We are interested in the probability of

$$p(y_i \mid w_1^i, \ldots, w_n^i) = \frac{p(w_1^i, \ldots, w_n^i \mid y_i) p(y_i)}{p(w_1^i, \ldots, w_n^i)}$$

which is given by Bayes' rule. The Bayesian approach is formulated as follows:

- We first assign pseudo-counts to each word that act as priors in the Bayesian framework. This results in $\boldsymbol{m}^+, \boldsymbol{m}^- \in \mathbb{R}^{|\mathcal{V}|}$, representing the counts of words in positive/negative sentences. In our model, we initialized both to be vector of ones.
- We train the model by updating $\boldsymbol{m}^+$ and $\boldsymbol{m}^-$. We increment m_i^+ (resp. m_i^-) by the number of occurences of word w_i in positive (resp. negative) sentences.

The predicted probabilities for a sentence with features $\boldsymbol{x}_i$ is given by softmaxing over

$$\begin{bmatrix} \log\left(\frac{\boldsymbol{m}^+}{\|\boldsymbol{m}^+\|_1}\right)^T \boldsymbol{x}_i + \log\left(\frac{N_+}{N}\right) \\ \log\left(\frac{\boldsymbol{m}^-}{\|\boldsymbol{m}^-\|_1}\right)^T \boldsymbol{x}_i + \log\left(\frac{N_-}{N}\right) \end{bmatrix}$$

[1] We use *sentences* to mean a unit of data, which can in principle be multiple grammatical sentences.

where N_+ and N_- are respectively the number of positive sentences and negative sentences in the training set and $N = N_+ + N_-$. In addition, $\boldsymbol{x}_i$ is the max-over-time pooling of one-hot encoding matrix.

4.2 Logistic Regression

In this model, we consider the bag-of-words transform

$$x_i = \phi(w_1^i, \ldots, w_n^i) = \sum_{j=1}^{n_i} \text{onehot}(w_j),$$

and assume that

$$y_i \sim \text{Bern}(p_i) \quad p_i = \sigma(Wx_i),$$

for the sigmoid function

$$\sigma(t) = \frac{1}{1 + e^{-t}}.$$

The model is estimated via maximum likelihood over parameter matrix W. We maximize log likelihood via the Adam optimizer in a batched gradient descent setting, with a learning rate of 10^{-4} and weight decay of 10^{-4} for 20 epochs.

4.3 Continuous Bag of Words

A downside of logistic regression is that we have as many parameters as the size of our vocabulary $\mathcal{V}$. This means that without an extensive training set, we run the risk of overfitting. To address this, we can reduce the dimensionality of our embedding by using Mikolov et al. (2013)'s word embeddings. This model has three parts:

1. We start by using average-pooling over time on the word-embeddings to get a vector of dimension d for each sentence.
2. Then, we pass that vector through a linear transform to get another vector of dimension d_2 which then gets passed through a non-linear transformation (ReLU) to make up the hidden layer.
3. Finally, we pass the hidden layer output through another linear layer to get a vector with two elements, and a softmax transformation to turn these values into the probabilities of positive and negative sentiment.

The model has two parameter matrices of size $d \times d_2$ and $d_2 \times 2$ respectively. We implement the model for $d_2 = 100$. We train using Adam for 20 epochs. In summary, the model's steps are visualized in Figure 1.

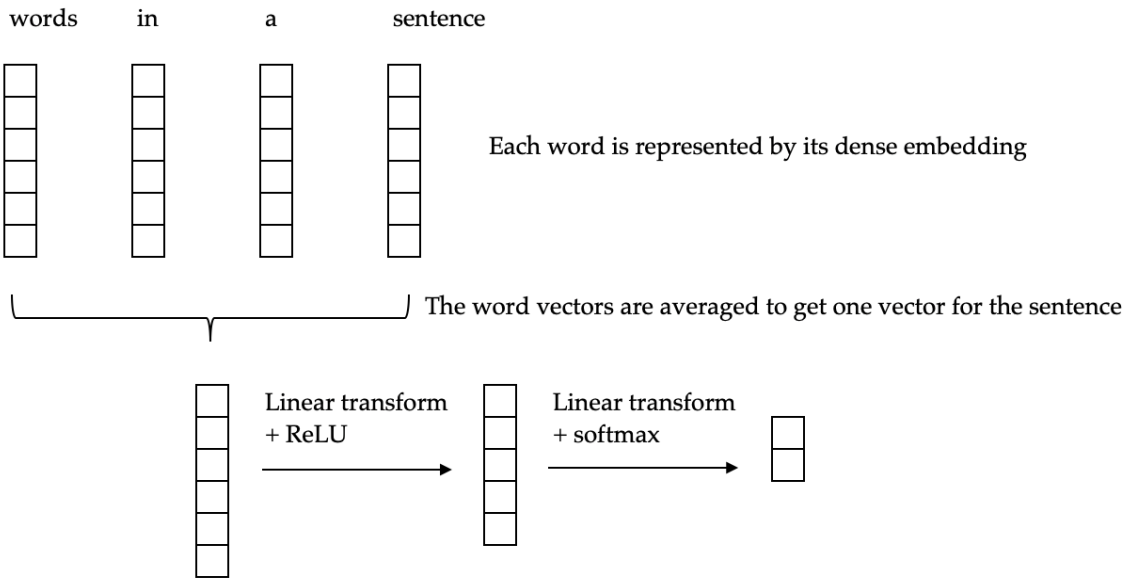

Figure 1: CBOW model pipeline

We call this model a *bag of words* because the order of the words is lost in the average-pooling over time. In other words, any permutation of a sentence will result in the same prediction by this model.

4.4 Convolutional Neural Network

While the Continuous Bag of Words model makes use of the pre-trained dense embeddings, it suffers a limitation from the bag-of-words construction. Because the order of words is not taken into account, the sentences "it bad, not good" and "it good, not bad" cannot be distinguished.

4.5 Ensemble

5 Experiments

Finally we end with the experimental section. Each assignment will make clear the main experiments and baselines that you should run. For these experiments you should present a main results table. Here we give a sample Table 1. In addition to these results you should describe in words what the table shows and the relative performance of the models.

Besides the main results we will also ask you to present other results comparing particular aspects of the models. For instance, for word embedding experiments, we may ask

you to show a chart of the projected word vectors. This experiment will lead to something like Figure **??**. This should also be described within the body of the text itself.

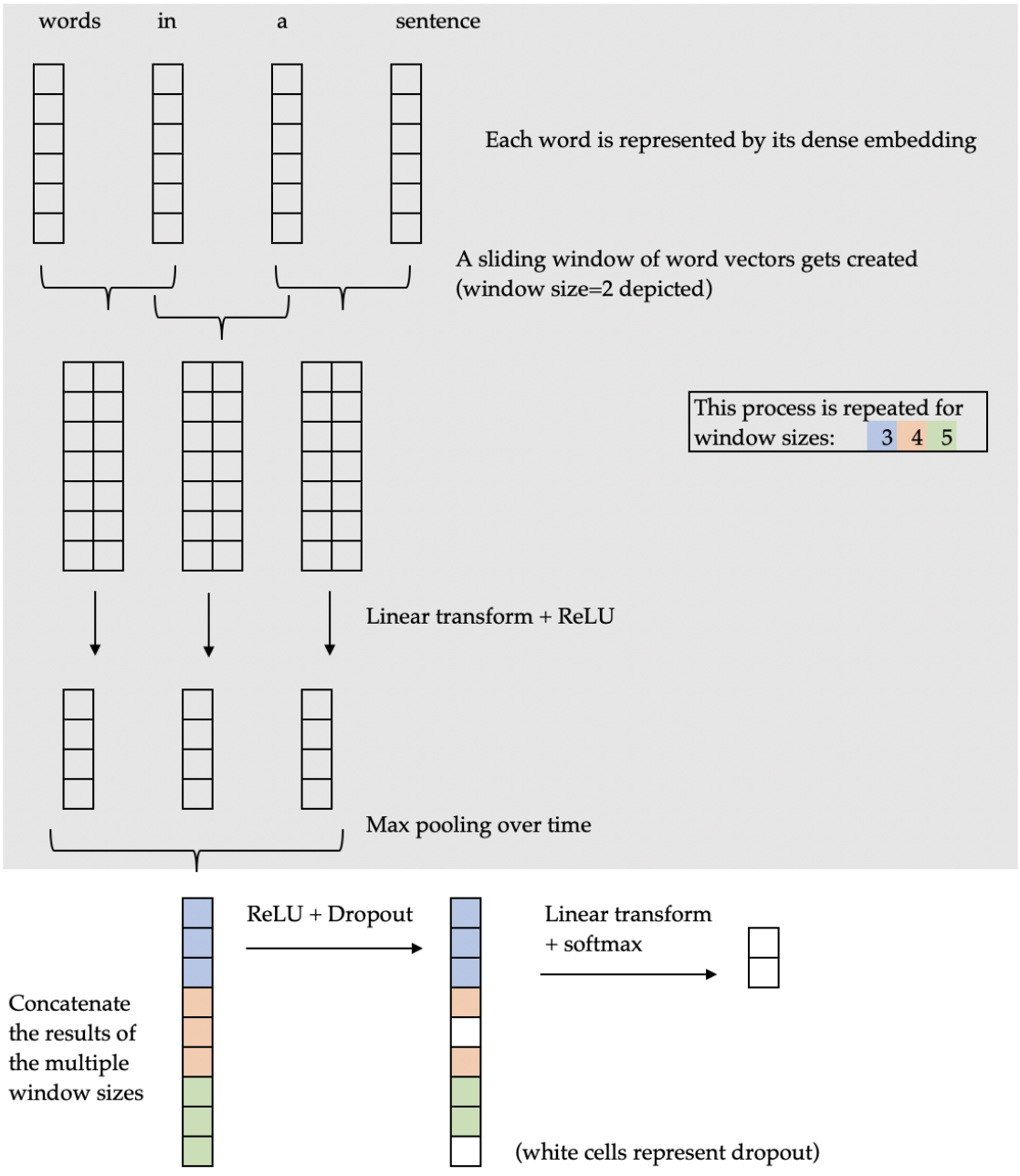

Figure 2: Convolutional Neural Network Model

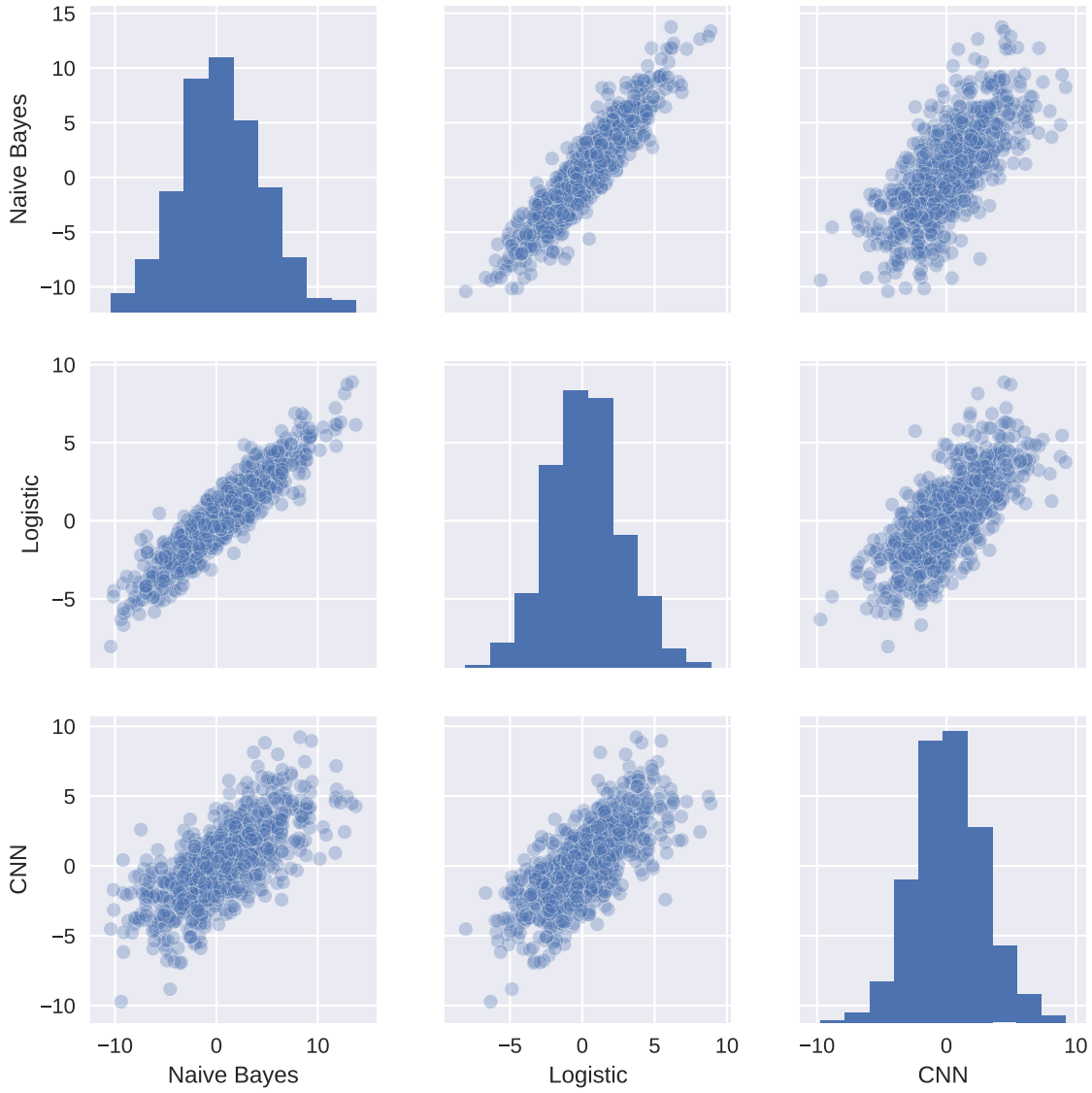

Figure 3: Correlation between Naive Bayes, Logistic, and CNN predictions on the validation set. We plot logit of the probabilities that a sentence is negative sentiment against each other (we clip $\pm\infty$ values in logit-space at ± 10).

	Training Accuracy	Training Average Loss	Validation Accuracy	Validation Average Loss
Naive Bayes	95.0%	0.0139	79.4%	0.0504
Logistic	98.8%	0.0131	78.2%	0.0487
Embedding NN	75.5%	0.0497	70.8%	0.0605
CNN	98.3%	0.0122	76.5%	0.0523
Ensemble	98.5%	0.0121	80.3%	0.0429
Ensemble-votes	98.7%	—	80.2%	—

Table 1: Accuracy and average loss (under cross-entropy loss function) for models considered. Note that for Ensemble, the ensemble weights are trained over the validation set and for the model Ensemble-votes, the loss cannot be computed since model does not output likelihood.

6 Conclusion

End the write-up with a very short recap of the main experiments and the main results. Describe any challenges you may have faced, and what could have been improved in the model.

References

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.