Sentiment analysis on Alexa Amazon reviews

Carlo Mugno CARLO.MUGNO@OUTLOOK.IT
Claudio Ragaglia CLAUDIORAGAGLIA@GMAIL.COM
Alessio Migliorino MIGLIORINOALESSIO@GMAIL.COM

1. Model Description

We describe here a **Recurrent Neural Network** we have built to classify Alexa Amazon reviews through **Sentiment Analysis**.

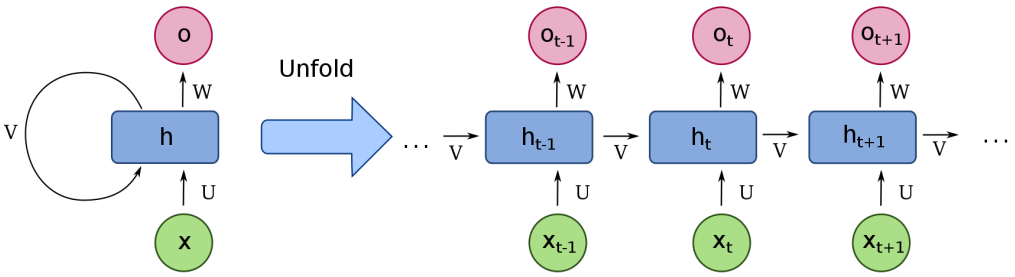

Figure 1: A simple representation of a general recurrent neural network.

Sentiment analysis (also known as opinion mining or emotion AI) is the use of natural language processing, text analysis, computational linguistics, and bio-metrics to systematically identify, extract, quantify, and study effective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. Our model uses three modules: Embedding, RNNCell (Recurrent Neural Network Cell) and Linear Classification in two class. The first one is often used to store word embeddings and retrieve them using indices. The input of the module is a list of indices, in particular the size of our vocabulary number; the output is the corresponding word embeddings. The second module represents only a single time step and it has as inputs the previous module's output (the size of the feature embedding), and the number of neurons in the recurrent layer. The last module applies a linear transformation to the incoming data.

2. Dataset

The dataset was provided by the Kaggle site: "https://www.kaggle.com/datasets/sid321axn/amazon-alexa-reviews". This dataset consists of a nearly 3000 Amazon customer reviews (input text), star ratings, date of review, variant and feedback of various amazon Alexa products like Alexa Echo, Echo dots, Alexa Firesticks. The reviews are extracted from Amazon's website. Thanks to the sentiment analysis we can analyse the reviews: how many positive? how many negative? We decided to drop almost all variables, except the reviews and their feedbacks, because for our analysis we just need those. Feedbacks are "1" for positive, with 2748 samples, and "0" for negative, with 216 samples; clearly the dataset is strongly unbalanced, and this is an important aspect that we will manage later. First we load the dataset in Google Drive because we can easily import it in the Google Colab environment. The dataset has several missing values: reviews without labels and viceversa; so we decided to drop them. Before to converting our reviews in numbers for training process, we split the dataset in two subsets: description (reviews) and labels (feedbacks). After we build our vocabulary, composed by description converted in number, we create two lists: the first one with all reviews labeled "0", and the second one with all reviews labeled "1". We proceed in this way because we want to create a balanced dataset. In the end we obtain a test set of 40 observations, a validation set of 20 observations and a train set of 372 observations. We also create an unbalanced train set with 2904 observations, which utility will be explained in the next section.

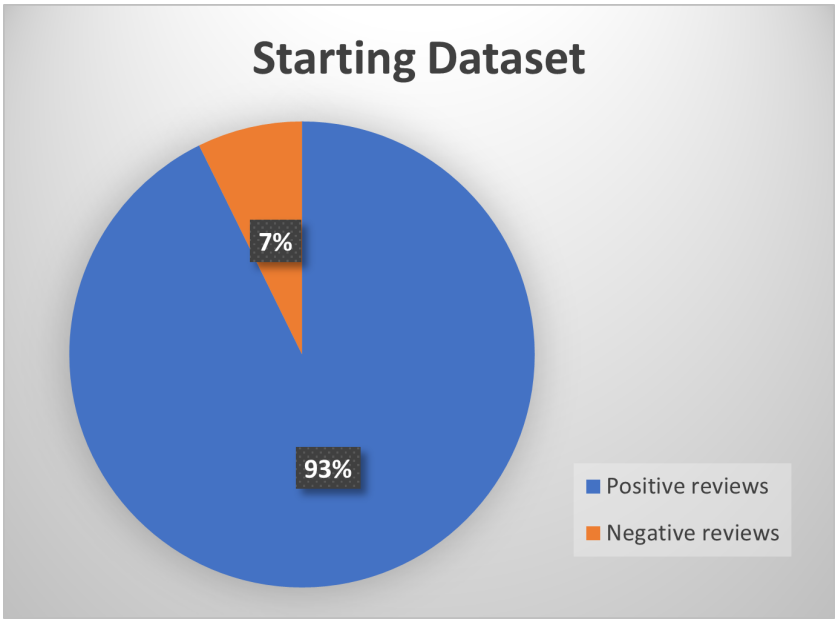

Figure 2: A pie chart of the unbalanced starting data distribution between classes.

3. Training procedure

This section is about the training options we decide to use for the model. Before starting the recurrent neural network, previously presented, we divide the reviews of each data loader (containing train, validation and test) in batches of 8 element size. This means that for data loader of the train we get 372/8 batches, for the validation one 20/8 batches and for test one 40/8 batches. We also create an other data loader containing the same validation and test used before but a different unbalanced training set with 2904/64 batches. We did different trials using an optimizer with four different learning rates (0.01, 0.001, 0.05, 0.005). For balanced training we define the **CrossEntropy** criteria for the loss, while for unbalanced training we used a **weighted CrossEntropy** criteria, giving more "weight" to the class with lowest number of samples. We process the model in 100 epochs, without a stopping criteria: reached the hundredth epoch with the balanced data, the algorithm also process the unbalanced data, repeating it for each learning rate, returning us the epochs with the maximum values of Validation Accuracy recorded every time.

4. Experimental Results

Several experiments were conducted to test the effectiveness of the proposed RNN architecture to develop a sentiment analysis. Table 1 shows the results for each run. From the below table we can notice that the best results are when the learning rate is 0.05 with both balanced and unbalanced data are used, with a Validation Accuracy equal to 0.791; but the best Test Accuracy, 0.70, is when we just use balanced data. Nevertheless the model with 0.005 learning rate and unbalanced data obtain a lower Validation Accuracy (0.75) the others parameters are better: Validation and Test Loss are lower and the Test Accuracy is equal to 0.725.

Learning Rate	Validation Accuracy (Test Accuracy)
– 0.01 (balanced)	75% (52.5%)
– 0.01 (unbalanced)	70.8% (72.5%)
– 0.001 (balanced)	75% (72.5%)
– 0.001 (unbalanced)	70.8% (60%)
– 0.05 (balanced)	79.1% (70%)
– 0.05 (unbalanced)	79.1% (57.5%)
– 0.005 (balanced)	70.8% (70%)
– 0.005 (unbalanced)	75% (72.5%)

Table 1: Performances of the models when using different learning rate and different balances of the data.

At the end we have run again the best model (0.005 learning rate and unbalanced data) and use it trying the classification of several sentences reviews written by us. For example when we give the sentence 'I like this product so much' it returns the 99.7% of being a positive sentence review.