Carla Vega Molina 231678
Paulina Campero Baghirzade 233299
Klaus Ditterich Martín 231665

REPORT FINAL LAB

Our project consists in designing a Recurrent Neural Network for sentiment analysis. For this supervised learning process we have chosen a dataset IMDB movie reviews. The dataset is composed by two columns:

- Review which consists of a brief explanation of the opinion of a viewer regarding a movie. Each review is different from the others.
- Sentiment which refers to the overall score, i.e, if it was bad or good, or in other words, if it causes a positive or negative sentiment.

The first thing we are going to do in our project is to clean our dataset and organize it correctly. Concretely, we modified the review column which had a lot of insignificant information and formats that can affect and complicate the prediction of the sentiment. Basically we have many reviews and lots of words where in many cases they are the same written in a different form or conjugation of the same. To facilitate the process we will focus on eliminating all these differences, so that the program sees it as a single word that is repeated in the dataset. Also, we have the case that there are many words that are repeated in our language just because they are essential to unite the words that do contribute meaning to what we want to communicate. In this cleaning we will take care of improving this. This part has many substages:

1. Withdraw the line breaks tags and the numbers of all the samples.
2. Remove extra white spaces.
3. Set all the words to lowercase so that we can recognize the same word but written in different formats.
4. Remove the short forms so that we consider all the words as complete and because contraction gives more complexity to our problem and we want to avoid it.
   In the following image we can see the different replacements that we consider to our problem:

```
text=re.sub("n't"," not",text)
text=re.sub("let's",'let us',text)
text=re.sub("'s",' is',text)
text=re.sub("won't",'will not',text)
text=re.sub("'re",' are',text)
text=re.sub("'ve",' have',text)
text=re.sub("'d",' would',text)
text=re.sub("'ll",' will',text)
text=re.sub("'m",' am',text)
text=re.sub("'d"," did",text)
text=re.sub('theres','there is',text)
text=re.sub(' u ',' you ',text)
```

5. Remove punctuation because when we are going to split the several reviews, the punctuation is added to the word which is next to. Therefore we could have the same words but the program sees it as if they were two different ones. Also it gives no specific semantic meaning. So in order to prevent it, it is better to eliminate it. Some examples can be: ‘ !”#$%&\’()*+,-./:;<=>?@[\\]^_`{|}~ ’
6. Apply the function *text_to_word_sequence* which is in charge of splitting each review in a list of words.

7. Apply the lemmatize function to remove all the extra parts of the words that are insignificant such as verb tenses, declensions, etc. to conserve only the root part.
8. Remove stopwords. Stopwords are those ones, which we previously mentioned, that are useful to link the different parts of the sentence and are essential but they don't give any meaning to what we want to express.
   Using the function *stopwords.words('english')* we achieve to remove all this insignificant part. Those words are the following seen in the screenshot:

{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

So after applying all those changes we reach the following results.

Before cleaning:

*"One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with me.

The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. Trust me, this is not a show for the faint hearted or timid. This show pulls no punches with regards to drugs, sex or violence. Its is hardcore, in the classic use of the word.

It is called OZ as that is the nickname given to the Oswald Maximum Security State Penitentary. It focuses mainly on Emerald City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.

I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. Forget pretty pictures painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was surreal, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence. Not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side."*

After cleaning:

"one reviewer mention watch oz episode hook right exactly happen first thing strike oz brutality unflinching scene violence set right word go trust show faint hearted timid show pull punch regard drug sex violence hardcore classic use word call oz nickname give oswald maximum security state penitentary focus mainly emerald city experimental section prison cell glass front face inwards privacy high agenda em city home many aryan muslim gangstas latino christian italian irish scuffle death stare dodgy dealing shady agreement never far away would say main appeal show due fact go show would dare forget pretty picture paint mainstream audience forget charm forget romance oz mess around first episode ever saw strike nasty surreal could say ready watch develop taste oz get accustom high level graphic violence violence injustice crook guard sell nickel inmate kill order get away well mannered middle class inmate turn prison bitch due lack street skill prison experience watch oz may become comfortable uncomfortable view thats get touch darker side."

Carla Vega Molina 231678
Paulina Campero Baghirzade 233299
Klaus Ditterich Martín 231665

After doing all the data cleaning we are going to encode the input and output. So as we are in a binary classification problem we substitute positive with 1 and negative with 0. Regarding the input we also encoded it. The way in which we encode the different words corresponds to the frequency of the word itself. If the word is the most common one we encoded with a 1 and so on. To achieve it we construct a list of all the words in the dataset and we use the function *most_common*.

Then, after that, we have a large amount of encoded reviews, but each one has a specific length. So, as the input needs to always have the same dimension, we are going to compute the average length of all the reviews and truncate or add some dimensions to each review. We are also considering the possibility of using the maximum length so that we don't remove information of certain reviews and add with 0s the free spaces. Finally, we construct our input matrix.

Then, we are going to do the embedding part. This section consists of initializing the weights of the embedding layer. The embedding layer enables us to convert each word into a fixed length vector of a defined size. In our problem we decide to use a pre-trained word vector to train our RNN model. The most commonly used models for word embeddings are word2vec and GloVe. In our lab we use the GloVe embedding. We have chosen GloVe instead of word2vec, since GloVe also takes into account the frequency of the words rather than only relating the word with its context, which is additional information that could be useful. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Now, we are able to build our model and start doing the training and testing and predict the sentiment of certain reviews. We are going to use an LSTM model, because reviews are long and we need to have a model that either drops or keeps the information in the hidden state. We want to notice that we have a final output neuron that uses sigmoid function as we have a binary classification problem.

Our model has the following layers:

```
self.embedding=nn.Embedding(vocab_size, embedding_dim)
self.embedding.weight = nn.Parameter(torch.tensor(word_embedding,dtype=torch.float32))

self.lstm=nn.LSTM(embedding_dim, hidden_dim, n_layers, dropout=drop_prob, batch_first=True)

#dropout layer
self.dropout=nn.Dropout(0.3)

#Linear and sigmoid layer
self.fc1=nn.Linear(hidden_dim, 64)
self.fc2=nn.Linear(64, 16)
self.fc3=nn.Linear(16,output_size)
self.sigmoid=nn.Sigmoid()
```

Carla Vega Molina 231678
Paulina Campero Baghirzade 233299
Klaus Ditterich Martín 231665

Finally we are going to test different parts in order to achieve the maximum accuracy like changing hyperparameters such as the embedding dimensions available in Glove (100, 200 and 300), hidden dimensions and dropout probabilities, the type of optimizers and its parameters, other embedding algorithms and the equating dimension of the input. Those results are shown in the results file. Finally, we reach an accuracy of 88,5 with LSTM, using the SGD optimizer with learning rate of 0,06 and a momentum of 0,95. We use 2 LSTM layers and a hidden dimension of 256 with the previous architecture and 6 epochs.