RESULTS

In the below table, I tried to show my work according to the preprocessing steps and model parameter/hyperparameters.

Tokenizer	Stemming	Lemmatization	Negation	Model	Optimizer	Accuracy
RegexpTokenizer(r'\w+')	-	-	-	Emb+ Linear Layer	Adam Lr=0.001	69.24%
RegexpTokenizer(r'\w+')	SnowballStemmer	-	-	Emb+ Linear Layer	Adam Lr=0.001	70.02%
RegexpTokenizer(r'\w+')	-	WordNetLemmatizer	-	Emb+ Linear Layer	Adam Lr=0.001	71.49%
RegexpTokenizer(r'\w+')	-	WordNetLemmatizer	pos_tag	Emb+ Linear Layer	Adam Lr=0.001	69.90%
RegexpTokenizer(r'\w+')	-	-	-	Emb(75)+ Linear Layer*	Adam Lr=0.001	69.84%
RegexpTokenizer(r'\w+')	-	-	-	Emb(75)+ 2Linear Layers*	Adam Lr=0.0002	64.78%
RegexpTokenizer(r'\w+')	-	-	-	Emb(75)+ 2Linear Layers*	Adam Lr=0.02	64.78%
RegexpTokenizer(r'\w+')	-	WordNetLemmatizer**	-	Emb+ Linear Layer	Adam Lr=0.001	70.09%

In loss part, '*' sign shows that I used truncated frequent words with freaquency is less than 100. Also, I tried to more deeper networks, but I got the RAM error. '**' sign shows that I use lemmatization in postprocessing part. As you can see the results of lemmatization, running as a postprocessing gave a little worse result than the other. On the other hand, while running the preprocessing and different models Lemmatization ones gave much better results.

According to accuracy results, I can conclude that preprocessing implementations are not so different. Actually, they are really close to each other. While using lemmatization and stemming processes, the run time is not worthy to implement these processes. Changing the hyperparameters of the model gives more considerable changes. I would like to have more time to try different preprocessing-postprocessing and hyperparameter in order to see that I get better accuracy.