DLSL- Toxic Comment Classification Challenge

Team 1

Mireia Gartzia, Carlos Arenas, Itziar Sagastiberri

30th of January, 2018

Overview

- **Introduction**
- Baseline architecture
- Experimental setups
- Results
- Conclusions
- References

Introduction

Aim of the challenge

- **Detect** toxic written comments in a forum (Dataset from wikipedia forum)
- **Classify** the kind of toxicity comment:

Toxic	Severe toxic	Obscene	Threat	Insult	Identity hate

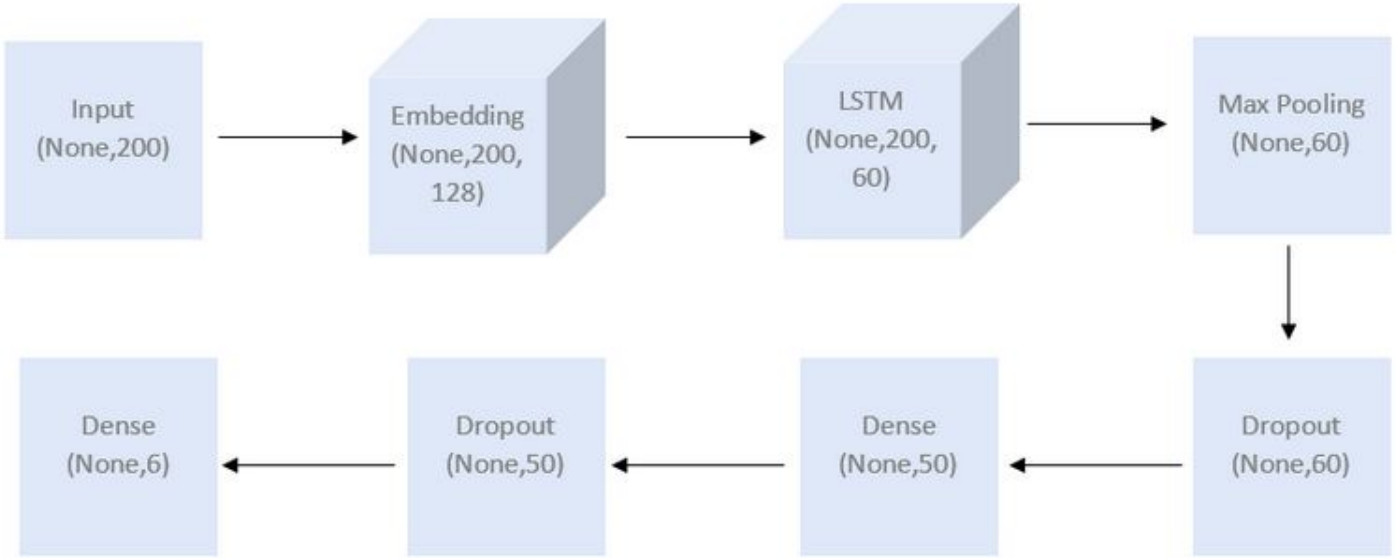
Our personal goal

- **Improve** an already working project by changing the architecture, the hyperparameters…
- **Understand** and **justify** our experiments and results

Overview

- Introduction
- **Baseline architecture**
- Experimental setups
- Results
- Conclusions
- References

Baseline Architecture

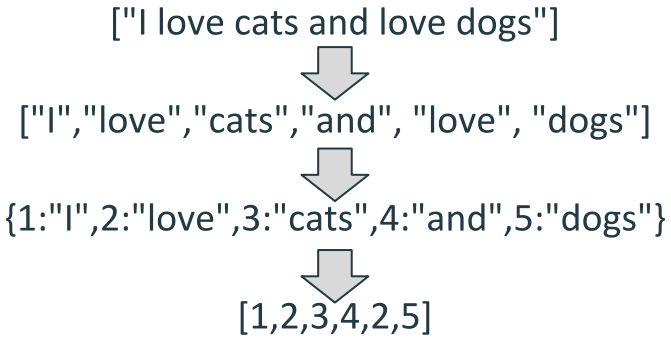


Taken from: https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras

Baseline Architecture

Text pre-processing

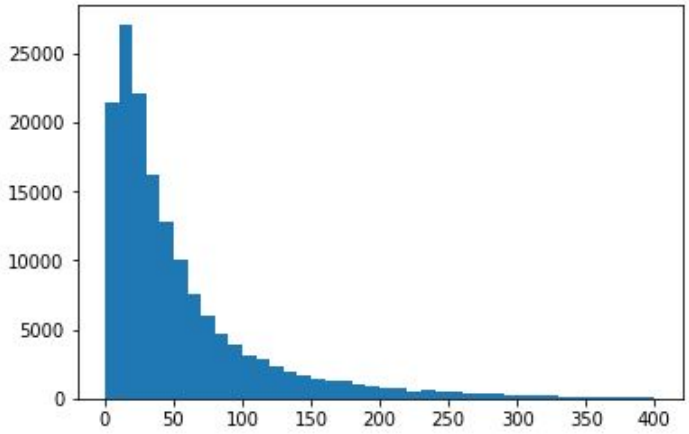
- **Tokenization** and **indexing** the words from the splitted sentences:

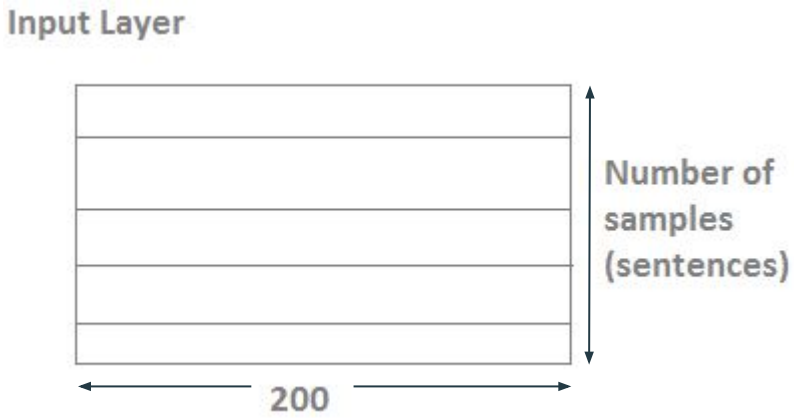


Baseline Architecture

Text pre-processing

- **Padding/truncating** each sentence to feed the Embedding layer with the same input shape (200 words/sentence). The **input** is a **2D tensor (None, 200)**

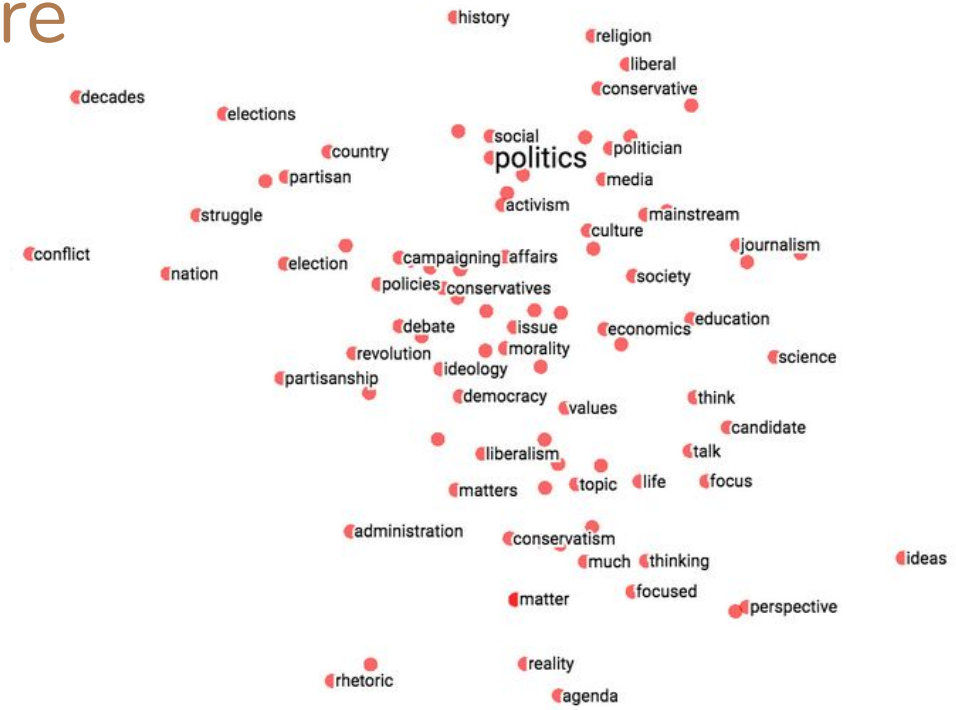




Baseline Architecture

Embedding

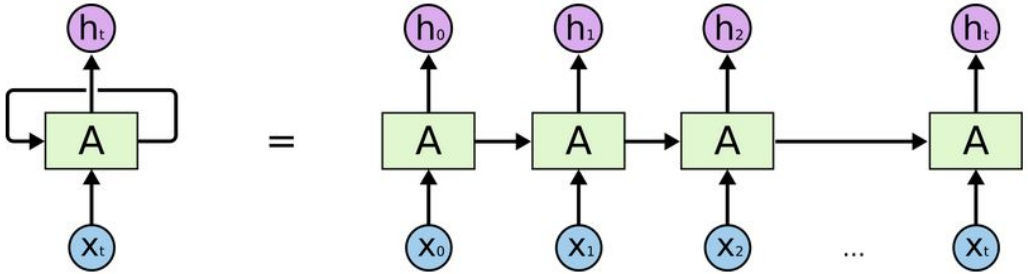
- **Projecting the vocabulary words** to a defined vector space depending on the distance of the surrounding words in a sentence.
- **Reduce** model dimensions
- Vectorial space of **size = 128**
- Output: **3D tensor (None, 200, 128)**



Baseline Architecture

LSTM

- It feeds the input of the current network with the output of the previous step recursively
- We take the outputs (60) of each time step (200), obtaining a **3D tensor (None, 200, 60)**



- A **max pooling 1D layer** takes all outputs from the LSTM to reduce the time dimensionality
- Reshape into a **2D tensor (None, 60)**

Baseline Architecture

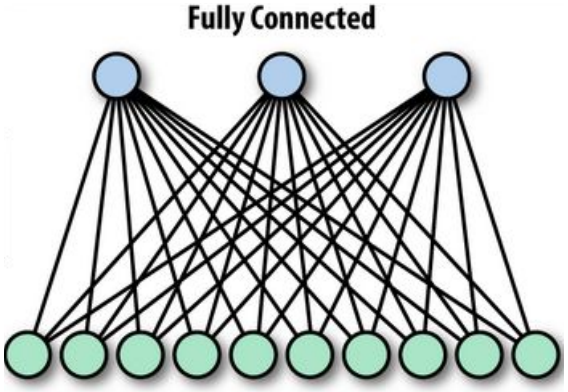
Fully connected (Dense)

First FC

- 60x50
- Activation function : **RELU**

Second FC

- 50x6
- Activation function: **Sigmoid**
  (binary classification for one of each 6 toxicity labels)

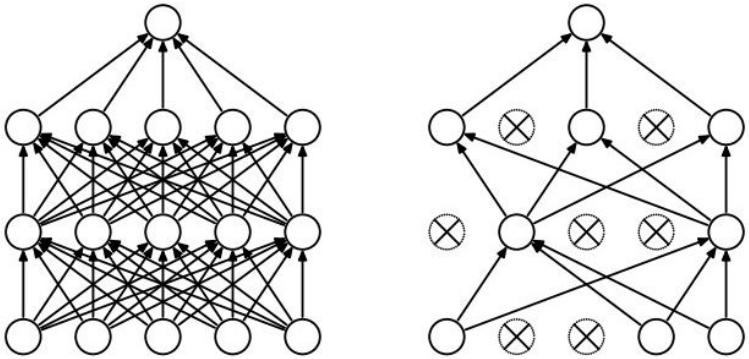


- The network has **6 outputs** (1 per label), where the values are 1 or 0 depending on whether it belongs to a class or not respectively.

Baseline Architecture

Dropout

- It randomly disables the **10%** of the nodes **in both FC layers**
- The whole network could result in **better generalization**



(a) Standard Neural Net

(b) After applying dropout.

Overview

- Introduction
- Baseline architecture
- **Experimental setups**
- Results
- Conclusions
- References

Experimental setups
Hyperparameters fine-tuning

- **Length of sentences (# words/sentence)**
  - In the range [80, 250] the baseline length **(200)** was the best
- **Length of the embedded vector**
  - In the range [64, 256] the baseline length **(128)** was the best
- **Dropout**
  - First layer at **0.2** (instead of 0.1) and second layer remains at **0.1**
- **Batch size**
  - In the range [16, 64] the baseline size **(32)** was best
- **Learning rate**
  - A higher value **0,0025** (instead of 0.001) improves the initial results.

Experimental setups

Early stopping

- Type of regularization to **control overfitting** of the training data
- It stops training once performance on the validation dataset starts to degrade
- Configuration:
  - Maximum # epochs = 20
  - Patience of 2 epochs
  - Monitors the validation loss

Experimental setups
Different architectures

- Add **Batch Normalization** layer after LSTM
- **Bidirectional** LSTM
- **2 LSTM without MaxPooling1D** layer
- 1 LSTM **without MaxPooling1D** layer (we take only the **outputs of the final time step**) 

Experimental setups

Initialization with pre-trained word embeddings

Objective

- Boosting accuracy and take advantage of existing work.
- 100D GloVe (Global Vectors for Word Representation) embeddings of 400k words computed on English Wikipedia.

Approach

1st Prepare an embedding matrix that maps our vocabulary words in the pre-trained embedding vector space

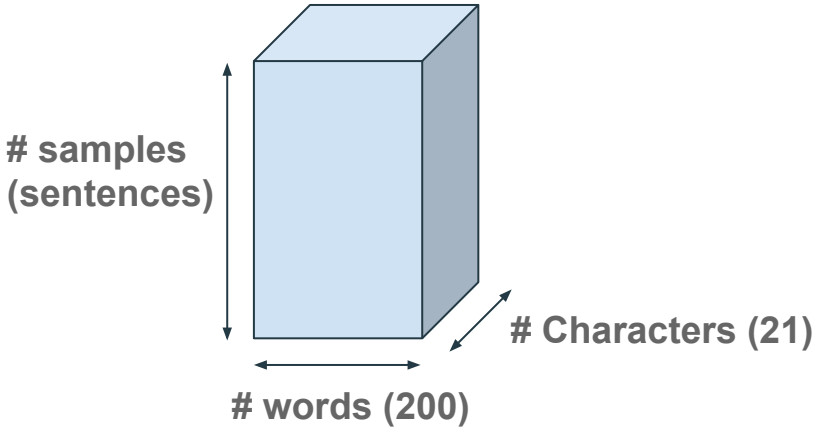
2nd Load this matrix into the Keras Embedding layer, set to be frozen (its weights, the embedding vectors, will not be updated during training)

Experimental setups

Combining multiple inputs

1st Create a new data input based on characters (instead of words)

- **Tokenization** and **indexing** the characters from the splitted words
- **Padding/truncating** each word and sentence to feed the Embedding layer

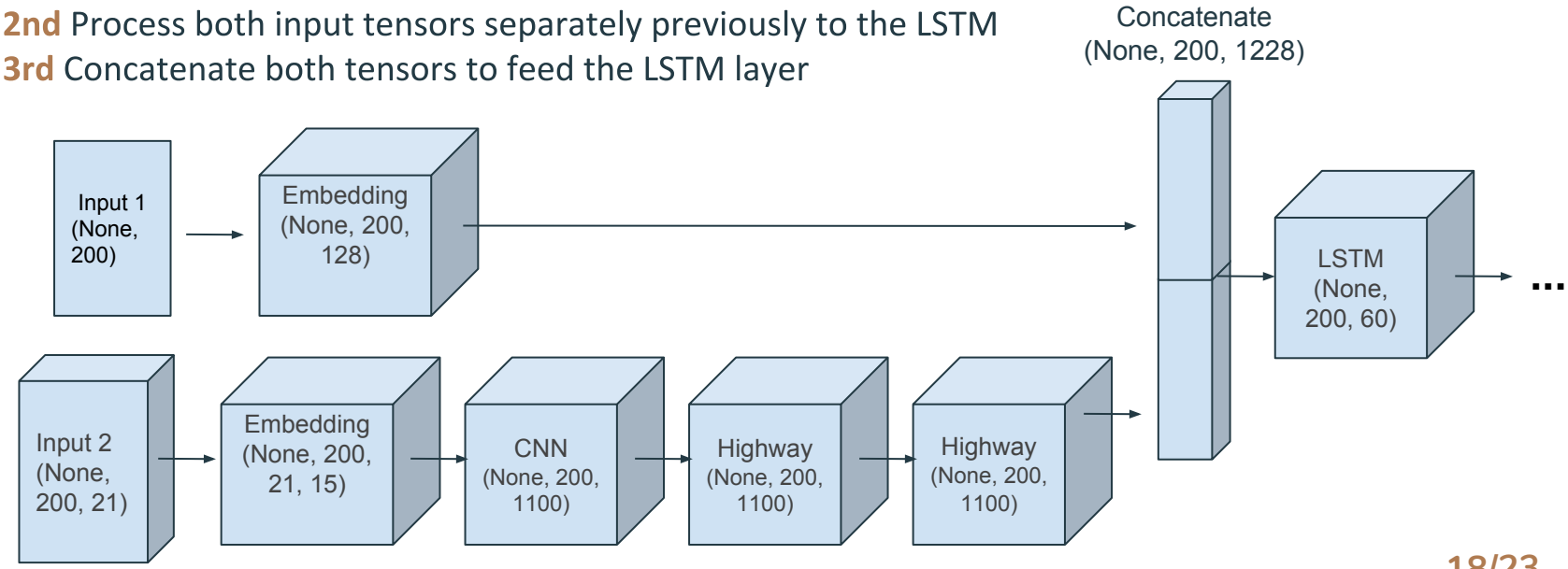


Experimental setups

Combining multiple inputs

2nd Process both input tensors separately previously to the LSTM

3rd Concatenate both tensors to feed the LSTM layer



Overview

- Introduction
- Baseline architecture
- Experimental setups
- **Results**
- Conclusions
- References

Results

Configuration of our final Approach

- Hyperparameters fine-tuned:
  - Dropout 1 with **0.2** and Dropout 2 with **0.1**
  - Learning rate: **0.0025**
- Early stopping which stops at the **second epoch**
- 1 LSTM with the outputs of the final time step (**avoid MaxPooling1D layer**)
- Initialization with **pre-trained word embeddings**

	Validation loss	Validation Accuracy
Baseline Architecture	0.0482	0.9822
Combining multiple inputs	**0.1431**	**0.9627**
Final approach	**0.0470**	**0.9825**

Overview

- Introduction
- Baseline architecture
- Experimental setups
- Results
- **Conclusions**
- References

Conclusions

- This challenge allows to achieve very good results with a quite basic structure
  → More **difficult** to obtain even better results
- Explore deeply almost all possible configurations to our baseline architecture
  → Increasing the complexity of the architecture **not always** ensures better results

Future Work

- Start from **another baseline architecture**
- Include **attention-based mechanism**

References

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

https://www.kaggle.com/sbongo/for-beginners-tackling-toxic-using-keras

https://github.com/jarfo/kchar

https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

Questions?