Sentiment Analysis with RNN and LSTM

Juan David Bahamon
Samuel Hernández
David Peñaranda

Description

Development of sentiment analysis models through supervised learning with Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).

Dataset and Preprocessing

Source

"Sentiment Labelled Sentences Dataset" from the University of California, Irvine.

Some examples...

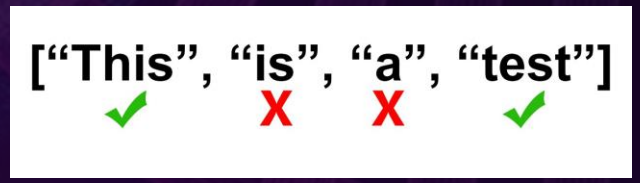
So there is no way for me to plug it in here in the US unless I go by a converter. 0

A very, very, very slow-moving, aimless movie about a distressed, drifting young man. 0

Great for the jawbone. 1

Dataset and Preprocessing

Language Learning Models (LLMs) have revolutionized the field of natural language processing, enabling machines to understand and generate human-like text. At the core of LLMs lies the concept of tokens, which serve as the fundamental building blocks for processing and representing text data. In this blog post, we'll demystify tokens in LLMs, unraveling their significance and exploring how they contribute to the power and flexibility of these remarkable models.



Tokenization, lowercase conversion, and removal of stopwords using the NLTK library.

Methodology

Baseline

Dummy classifier results

Hyperparam tuning

Bathc size, dropout rate, epoch, learning rate and lstm units

Measuring the results

Accuracy

Precision

Recall

F1 Score

Cohen's Kappa

Sentiment Analysis Models

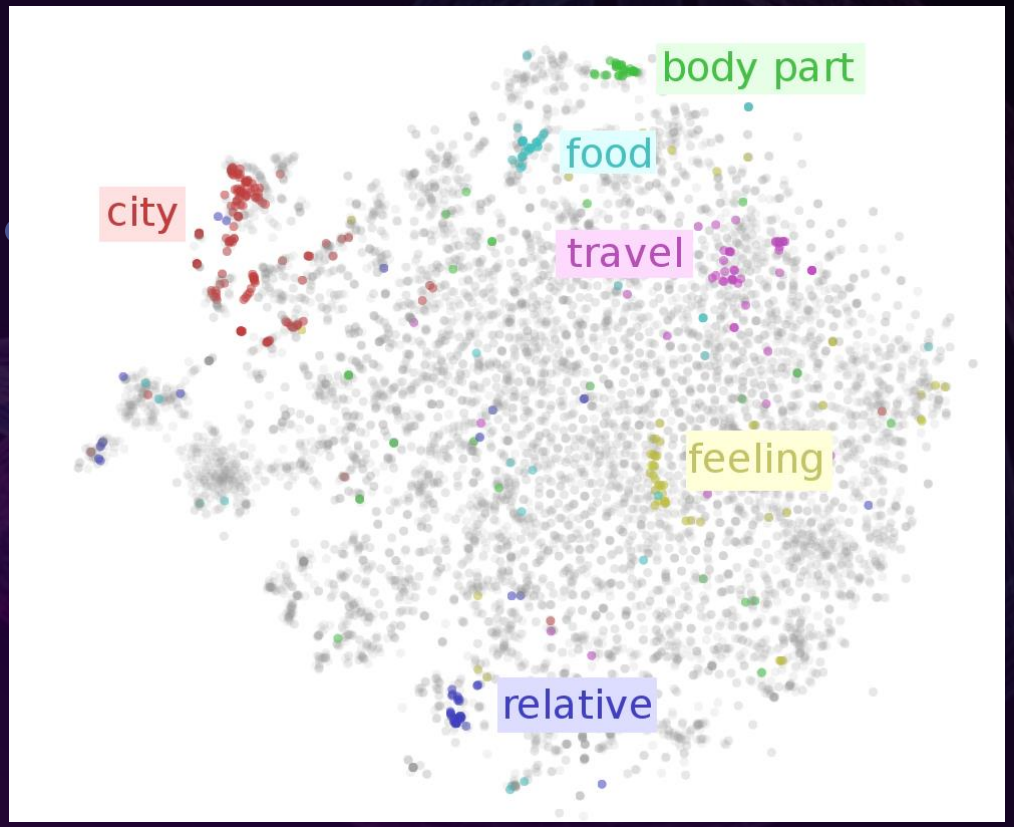
	Basic RNN	LSTM
Structure	Linear chain of recurrent layers.	Memory cells, input gates, forget gates, and output gates.
Functionality	Capturing sequential patterns in sentiment data.	Ability to retain long-term information for contextual understanding.

Architectures

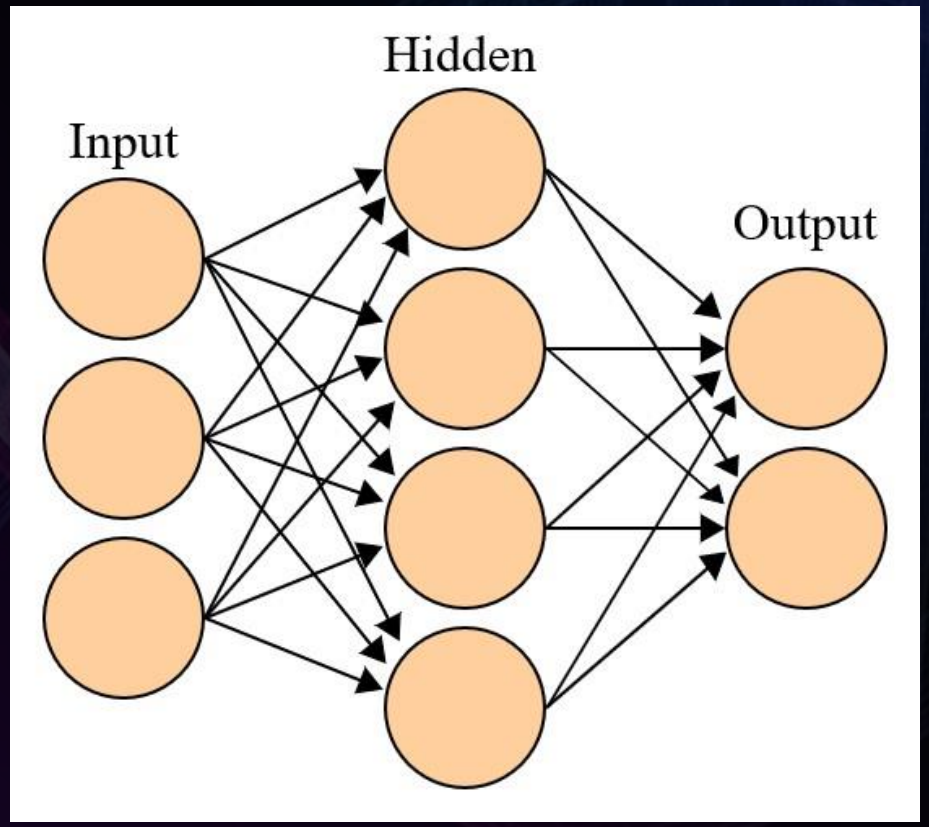
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 16)          160000

 simple_rnn (SimpleRNN)      (None, 8)                 200

 dense (Dense)               (None, 1)                 9

=================================================================
Total params: 160209 (625.82 KB)
Trainable params: 160209 (625.82 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         640000

 lstm (LSTM)                 (None, 50)                35800

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 675851 (2.58 MB)
Trainable params: 675851 (2.58 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Layers Explanation

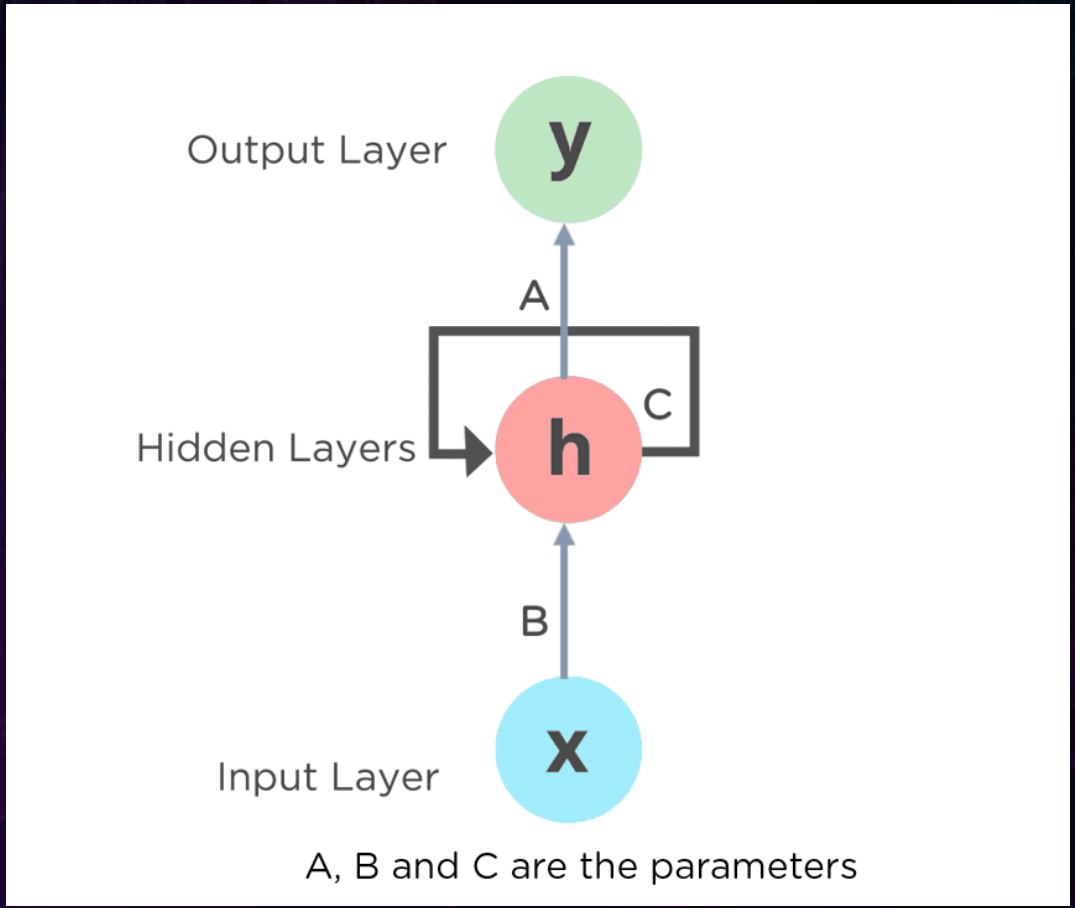


Embedding Layer

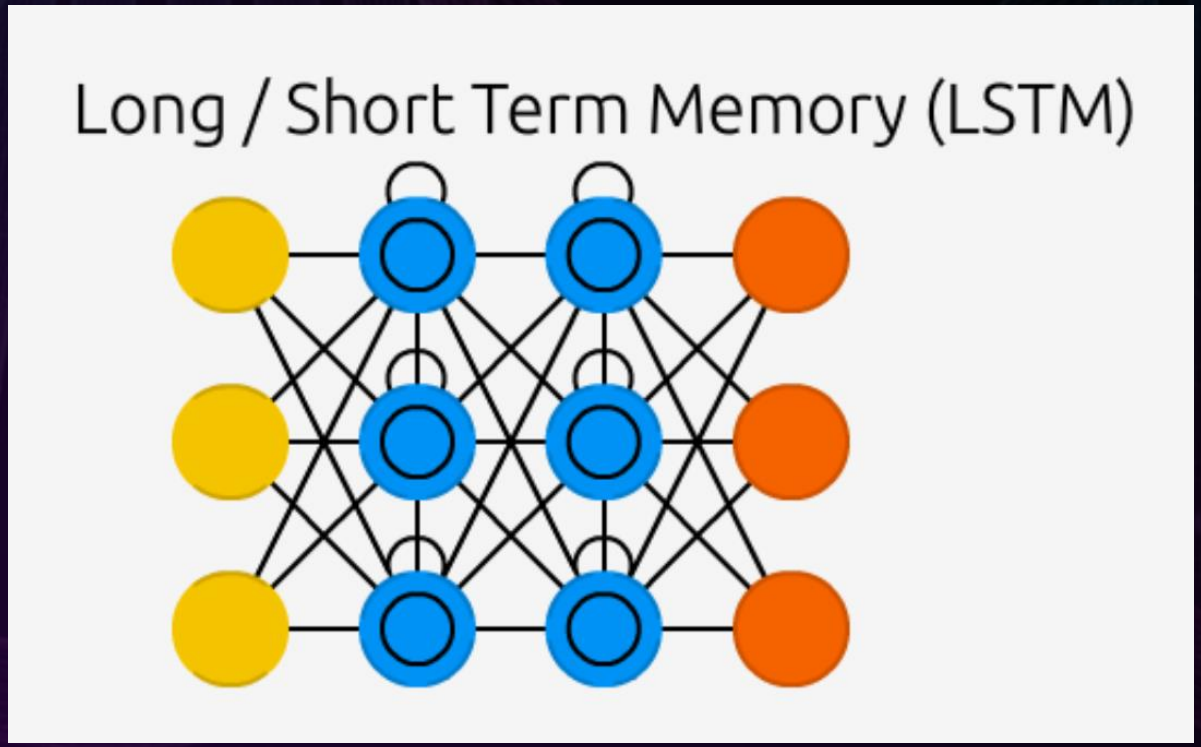


Dense Layer

RNN


Output Layer
y
A
C
Hidden Layers
h
B
Input Layer
x
A, B and C are the parameters

LSTM



Performance Evaluation

	DummyClassifier	RNN	RNN WITH HYPERPARAMETERS	LSTM	LSTM WITH HYPERPARAMETERS
Accuracy	0.5091	0.5915	0.7333	0.7963	0.7975
Precision	0.5091	0.6009	0.7475	0.7863	0.8019
Recall	1.0000	0.5880	0.7190	0.8238	0.8
F1 Score	0.6747	0.5944	0.7330	0.8046	0.8009
Cohen's Kappa	-	0.1830	0.4668	0.5922	0.5950

Better RNN parameters : {'batch_size': 32, 'dropout_rate': 0.5, 'epochs': 5, 'learning_rate': 0.001, 'lstm_units': 100}

Better LSTM parameter : {'batch_size': 32, 'dropout_rate': 0.2, 'epochs': 5, 'learning_rate': 0.001, 'lstm_units': 50}

Key insights

DummyClassifier < RNN < LSTM

Hyperparam tuning is not a perfect method (but is necessary. E.g. more epochs != better predictions)

Data is valuable only if it is processed (and analyzed) properly

Thanks!