UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

Development of a Deep learning based Sentiment Analysis and Evaluation Service

Ignacio Corcuera Platas

2018

TRABAJO DE FIN DE MASTER

Título:	Development of a Deep learning based Sentiment Analysis and Evaluation Service
Título (inglés):	Development of a Deep learning based Sentiment Analysis and Evaluation Service
Autor:	Ignacio Corcuera Platas
Tutor:	TUTOR
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

Development of a Deep learning based Sentiment Analysis and Evaluation Service

febrero 2018

Resumen

Palabras clave:

Abstract

Keywords:

Agradecimientos

A Gauss

Contents

Resumen **VII**

Abstract **IX**

Agradecimientos **XI**

Contents **XIII**

List of Figures **XV**

1 Introduction **1**

1.1 Context 1

1.2 Project goals 1

1.3 Structure of this document 1

2 Enabling Technologies **3**

2.1 Introduction 3

2.2 TensorFlow and Keras 3

2.3 GSITK 4

2.4 More stuff 4

3 Deep Learning Analysis **5**

3.1 Introduction 5

3.2 Artificial Neural Networks 5

3.2.1 Activation functions 8

3.2.1.1 Sigmoid function 8

3.2.1.2 Tanh function 9

3.2.1.3 ReLU function 10

3.2.1.4 Softmax function 10

3.2.2 Architecture 11

3.2.2.1 Feedforward neural network 11

3.3 Use cases 12

3.3.1 System actors 12

4 Architecture **13**

4.1 Introduction 13

5 Case study **15**

5.1 Introduction 15

5.2 Rule edition 15

6 Conclusions and future work **17**

6.1 Conclusions 17

6.2 Achieved goals 17

6.3 Future work 17

List of Figures

3.1 Artificial Neural Network Diagram 7

3.2 Neurol Model 8

3.3 Softmax function 9

3.4 Hyperbolic tangent function 9

3.5 Rectified linear function 10

3.6 Single-Layer Perceptron 11

3.7 Multi-layer Perceptron 12

CHAPTER 1

Introduction

Context

Project goals

- G1

Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 ...

CHAPTER 2

Enabling Technologies

Introduction

Evaluation is one of the most important task in a machine learning analysis.

TensorFlow and Keras

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using **TensorFlow** can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms. The project will be focused on building

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent and simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error. This makes **Keras** easy to learn and easy to use. This ease of use does not come at the cost of reduced flexibility: because Keras integrates with lower-level deep learning languages (in particular TensorFlow), it enables you to implement anything you could have built in the base language. In particular, as tf.keras, the **Keras API** integrates seamlessly with your TensorFlow workflows.

The main idea is to use the **Keras API** with **Tensorflow** as backend, in order to build the different neural network for performing the deeplearning analysis.

GSITK

GSITK is a library developed by the Intelligent systems group (GSI) on top of scikit-learn that eases the deveopment process on NLP machine learning driven projects.**GSITK** manages datasets, features, classifiers and evaluation techniques, so that writing an evaluation pipeline results fast and simple.

GSITK is divided in four main components:

- Datasets and DatasetManager. This module is based on datasets, they allow to load the information quickly . GSITK provides a series of default datasets that can be used, furthermore it allows you to import your own datasets by defining two files: a python and a yaml. The DatasetManager handled the datasets, it has persistence allowing you to load a dataset quickly once it has already been loaded a first time. All of the datasets are processed as a pandas dataframe. The way of creating and adding a dataset will be shown below.

- Feautures. This module can be used to extract features in a format supported by machine learning algorithms from datasets. Mostly of the components are build following the sklearn api, this allows them to be used as a part of a pipeline. Moreover, it provides some sentiment and deeplearning techniques for extracting features.

- Preprocess. This module provides tokenize and preprocess methods for cleaning the data before applying a machine learning technique.

- Evaluation. This module allows to create evaluations, this consists on evaluate one or more pipelines with one or more datasets. It will return the accuracy,f-score,prediction and recall of each pipeline-dataset evaluation. This allows the user to compare the different systems.

More stuff

CHAPTER 3

Deep Learning Analysis

Introduction

Machine learning techniques explores the construction and study of algorithms that can learn from and make predicts on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Deep learning[?] allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction Although there are many implementations, one of the most common is the neural network. A neural networks has been proved to be a powerful tool in speech recognition as it can be seen in this study[?]. In this chapter will give an introduction to the neural networks, and it will explain all the networks developed for all the use cases.

Artificial Neural Networks

A neural network is a mathematic tool that models, at a very low level, the functionality of the neurons in the brain. However, unlike a biological brain where any neuron is connected to any other neuron by a phisical distance, these **artificial neural networks** (**ANNs**) have discrete layers, connections, and directions of data propagation.

ANNs offer the following properties and capabilities[**?**]:

1. **Non-linearity**, a neuron is basically a non-linear device; therefore, a neural network will be equally non-linear.

2. **Transformation input-output**, the learning process of a neural network involves the modification of the connections between the different artificial neurons that make up the network. Therefore, the use of a neural netowrk will imply a transformation of the input parameters for getting a specific exit.

3. **Adaptability**, this kind of algortihms have a great capacity for adaptation by adjusting its parameters. Therefore, a trained neural network trained in a specific environment could be conditioned to operate in another similar environment.

4. **Indicative response**, in the context of patter recognition, a neural network can be designed to provide information not only of the pattern selected, but also the degree of confidence in the decision made. This information can be used later to reject old patterns.

5. **Contextual information**, the knowledge state comes from the activation of the neural network. Basically, each neuron is affected by the goblal activity of the other neurons that compose the network.

6. **Fault tolerance**, due to the massive interconnection, a neural netowrk physically implemented is tolerant to failures and dispite some neurons or connections are pontentially damaged, the total performance is not seriously affected.

The diagram 3.1 shows a simple overview of the **ANNs** architecture. Each circle represents a neuron, that are grouped in layers. The layers can be divided in three main categories:

- Input layers produce information from some source. The input could be of any type: text, image, audio,etc. These layers generates the information that feeds other nodes. As the diagram shows, they have no input-connectors with the **ANN**; only outputs.

- Hidden layers contain intermediate calculations of the network. Its functionality is to transform the inputs received by the previous layer (input or hidden) into something that the following layer can use for the next step.

- Output layers combines and modify the data once the network has performed all the transformations. They are in charge of produce the output scale that is desired.

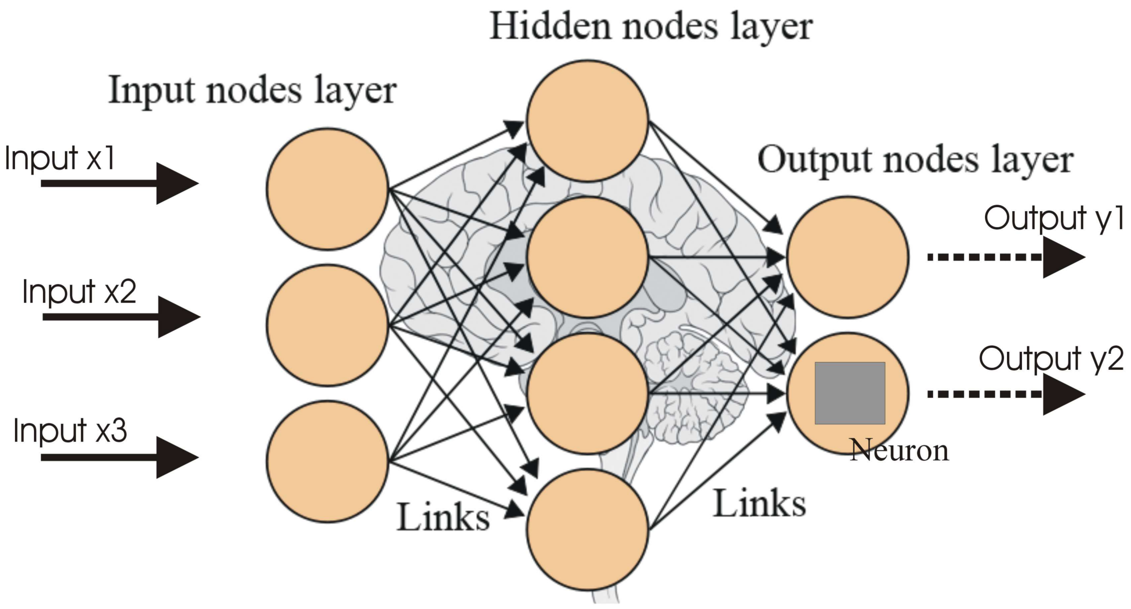

Figure 3.1: Artificial Neural Network Diagram

Neural networks can have any number of layers, and any number of nodes per layer. Most applications use the three layer structure. The hidden and output layers are categorized as 'active' due to they modify the data.

As it can be seen in the figure 3.2, a neuron is formed by the next components:

- Set of synaptic connections: each one characterized with a weigth
- Adder: that will add the components of the multiplied input signals by the weights of the synaptic connections.
- Activation function: responsible for limiting the output of the neuron between the desired values (non-linear transformation).
- Fixed entry: used to introduce a bias (b_k) in charge of increasing or decrease the input value of the activation function.

Basically, the main operation performed by the network of neurons is to multiply the values of a neuron by the weights of its outgoing connections. Each neuron in the next layer receives numbers from several incoming connections, and the first thing it does is add them all together. Them, it applies the activation function at the sum. Mathematically, the

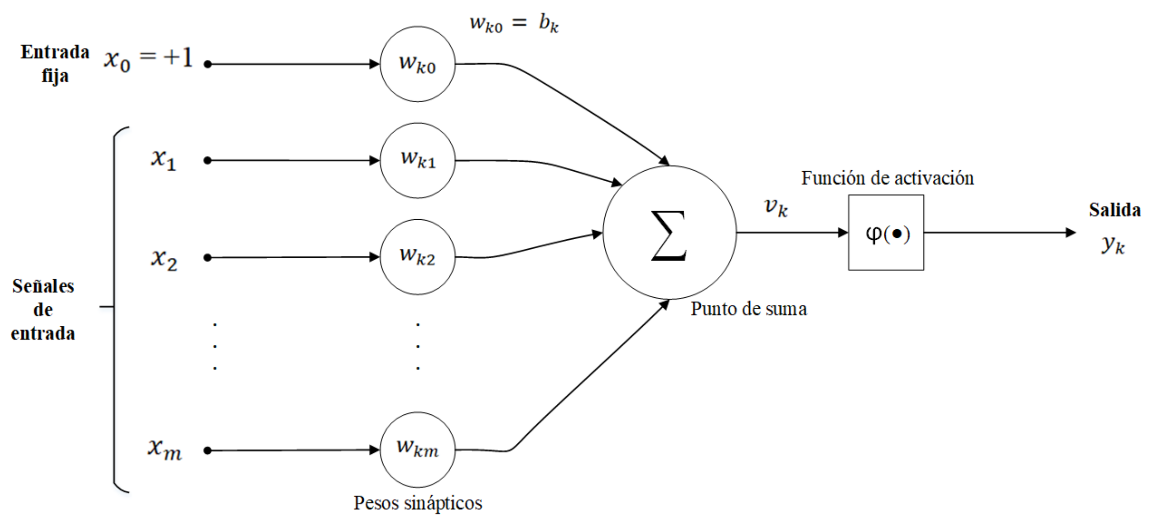

Figure 3.2: Neurol Model

model of a neuron is shown by the next formulas **(Fallos al añadir la ecuación revisar mas tarde)**.

$$u_k = \sum_{j=1}^{m} w_{kj} * x_j \tag{3.1}$$

$$y_k = \varphi(u_k + w_0 \tag{3.2}$$

Activation functions

In computational networks, the activation function of a node defines the output of that node given an input or set of inputs. Once the neuron add the weights and all the values together, it gots a value v_k that could be anything ranging from -∞ to ∞. The neuron doesn't know the bounds of this value or where it has to encamiante it. The activation function takes the decision of whether or not to pass the signal and transform that signal to something with bounds. This subsection will explain the different activation functions that will be related to the development of the deep learning analysis.

There are two main types of functions:

- **Linear Activation Function**, it is a simple linear function of the form $f(x) = x$, where the input passes through it without any modifcation.
- **Non-Linear Activation Functions**, these functions are used to separate the data that is not linearly separable and are the most used activations functions. TANH, RELU, SOFTMAX and SIGMOID will be explained below.

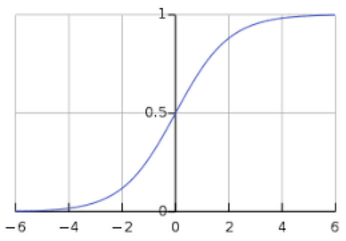

Figure 3.3: Softmax function

Sigmoid function

It consists in a special casa of the Logistic Activation Function. Basically, it transform the input in an output with bounds between 0 and 1 so it can be interpreted as a probability, it converts large negative number to 0 and large positive number to 1. The matematical and graphical represetantion are the following:

$$\sigma(x) = \frac{1}{1 + exp^{-x}} \tag{3.3}$$

Sigmoid are one of the most widely used activations functions. However, they have VANISHING GRADIENTS, which could affect at the learning process of the neural network. At 3.3 can be apreciated the representation of a sigmoid function. The order axis (Y) tends to respond very less to changes in the abscissas axis (X). This makes the neuron to enter in a saturated regime, making the network to refuse learning further or slowing the process . This problem will appear in one of the use cases (USE:CASE EMOJI). Despite of this problem, there are ways to work around this problem and sigmoid is useful in classification taks.

Tanh function

It is also know as the hyperbolic tangent activation function. It has two main differences with the sigmoid function, as it can be seen in the image. The first one, it is the bounds. The **tanh** transform the outputs with bounds between -1 and 1. The results is that the representation is now centered at zero. The negative inputs considered as strongly negative, zero input values mapped near zero, and the positive inputs regarded as positive.

This function can be writen as two sigmoid functions put togethers. The VANISHING GRADIENT is also present in this activation function.

$$\tanh(x) = \frac{2}{1 + exp^{-2x}} - 1 \tag{3.4}$$

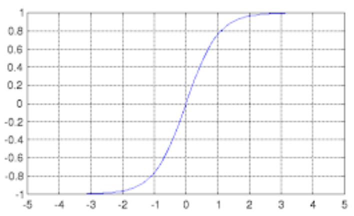

Figure 3.4: Hyperbolic tangent function

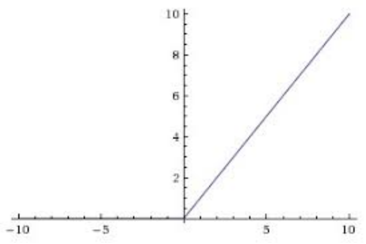

Figure 3.5: Rectified linear function

ReLU function

It is also know as rectified linear unit activation function. This functions gives the input as output if it is positive and 0 otherwise. The ReLu function is as show above**??**.

This activation makes the network converge much faster. It does not saturate which means it is resistant to the vanishing gradient problem at least in the positive region. Due to the horizontal line in **ReLu**(for negative X), the gradient can go towards 0. For activations in that region of **ReLu**, gradient will be 0 because of which the weights will not get adjusted during descent. That means, those neurons which go into that state will stop responding to variations in error/ input (simply because gradient is 0, nothing changes).

For solving this problem, there is a variant of the ReLU called LEAKY RELU. The are more implementations of the ReLU function, this paper describer a evaluation of a convolutional network using the different variants of the ReLU function as activation [**?**].

Softmax function

Sigmoid, tanh, relus are good activation functions for the neural networks. However, when you want to deal with classification probelmas, they seem to have some incovenients. The softmax functions transforms the input into an output with boudns between 0 and 1, just like a sigmoid function. Moreover, it divides each output such that the total sum must be equal to 1. This output is equivalent to a categorical probability distribution.

Matematically, the softmax function is shown below.

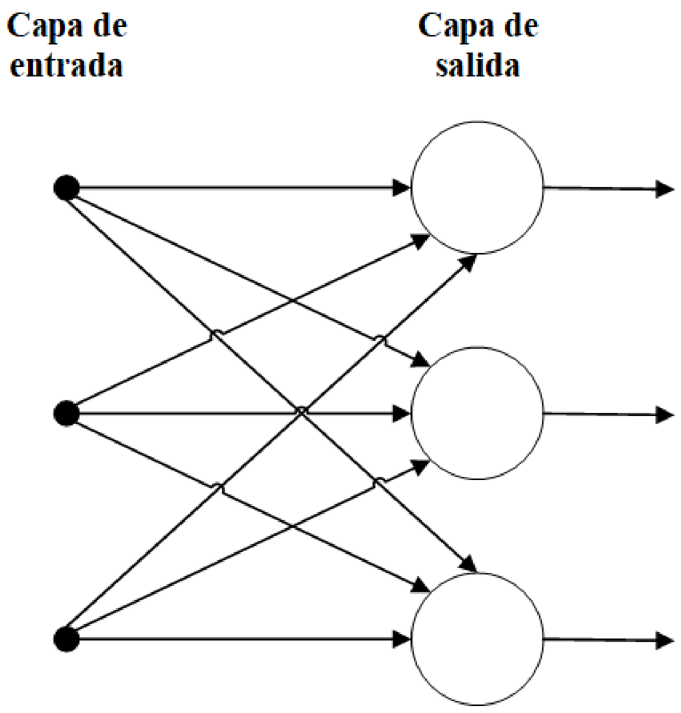

Figure 3.6: Single-Layer Perceptron

$$\sigma(x)_j = \frac{exp(x_j)}{\sum_{k=1}^{k}} forj = 1, ..., K \tag{3.5}$$

Architecture

This section will cover the different architectures of the neural networks. It will give an overview of the different types of general architectures and then it will explained some specific cases that have been used in the deep learning analysis. Typically, neural networks are organized in layers.

Feedforward neural network

A feedfoorward nueral network is an artificial neural network wherein connections between the units do not form a cycle. This kind of networks was the first and simples type of aritfical neural network devised. These networks are grouped into two types:

- **Single-layer perceptron**, the simples kind of neural network consists of a single layer of outputs nodes; as it can be seen in the diagram below **??** the different input neurons feed directly the output neurons.

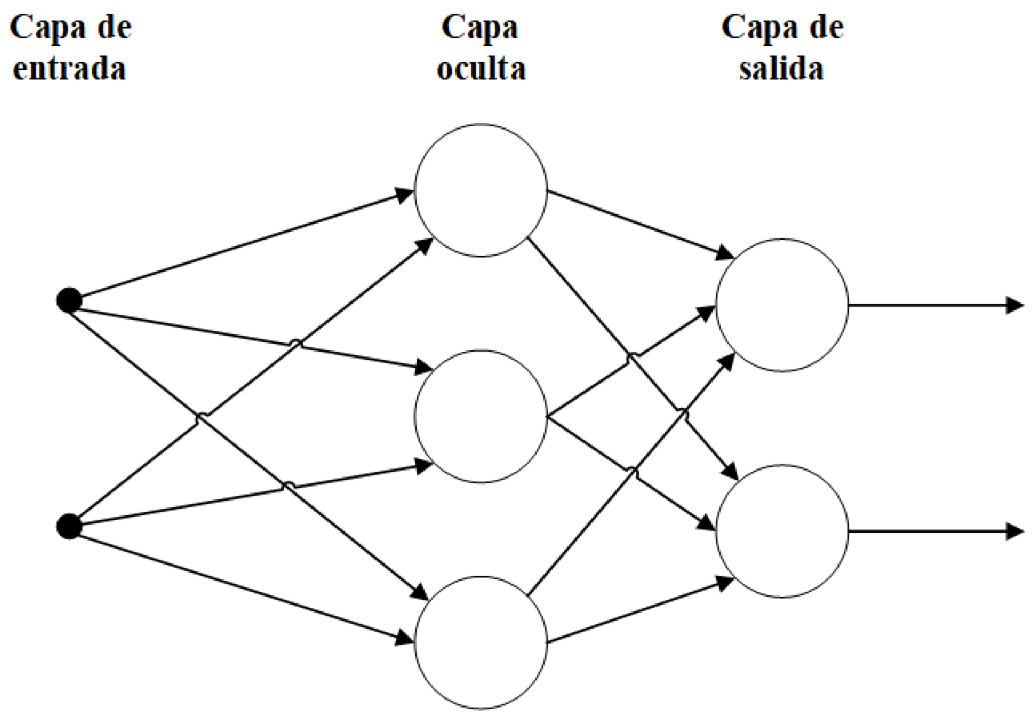

Figure 3.7: Multi-layer Perceptron

- **Multi-layer perceptron**, this kind of neural networks are similar to the single-layer perceptron but they at least one hidden layer between the input and the output layer. The incorporation of the hidden layer allows the network to obtain higher order statistics than its inputs, this fact allows the network to acquire a goblar perspective despite its local conectivity. If each one of the nodes that compose a layer are connected to the nodes of the following layer, moreover, this happend in all the layer of the neural network, the neural network is fully connected. In case this doesn't happend, the network will be parcially connected.

Use cases

System actors

CHAPTER 4

Architecture

Introduction

In this chapter, we cover the design phase of this project, as well as implementation details involving its architecture. Firstly, we present an overview of the project, divided into several modules. This is intended to offer the reader a general view of this project architecture. After that, we present each module separately and in much more depth.

CHAPTER 5

Case study

Introduction

In this chapter we are going to describe a selected use case. This description will cover the main Wool features, and its main purpose is to completely understand the functionalities of Wool, and how to use it.

Rule edition

...

CHAPTER 6

Conclusions and future work

In this chapter we will describe the conclusions extracted from this project, and the thoughts about future work.

Conclusions

Achieved goals

N1

Future work

- F1