U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc

# Deep learning approach to customer feedback understanding

Ricardo Oliveira

MSc thesis presented to
Faculty of Sciences of the University of Porto in
Computer Science

2019

MSc

2.º CICLE

FCUP
2019

U.PORTO

Deep learning approach to customer, feedback understanding

Ricardo Garcia Oliveira

FC

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

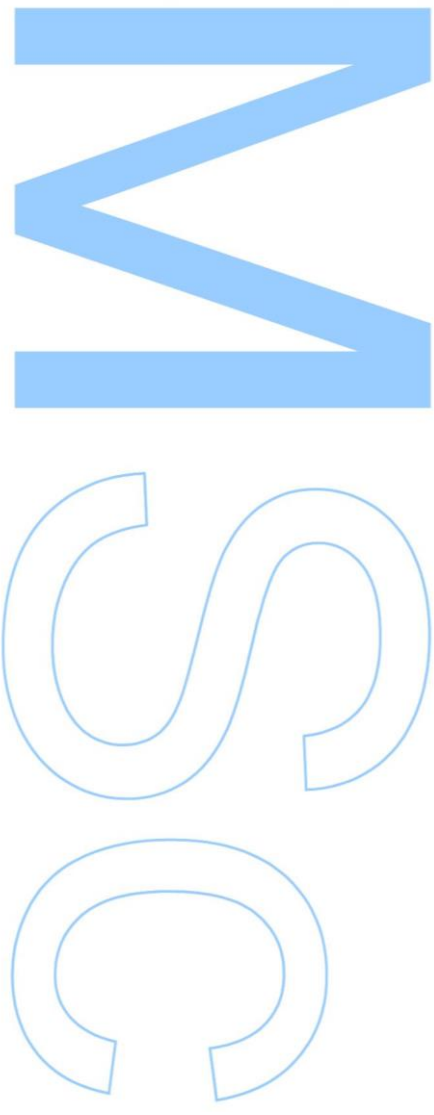
MSc

# Deep learning approach to customer feedback understanding

Ricardo Oliveira
Computer Science
Department of Computer Science
2019

**Orientador**
Nuno Moniz, PhD, FCUP

**Coorientador**
Alípio Jorge, PhD, FCUP



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

# Acknowledgements

I would like to thank my supervisor Nuno Moniz and my co-supervisor Alípio Jorge for all the guidance and patience. I would like to thank everyone at the Telco for receiving me and for allowing me to learn from them, it was a growing experience, both professionally and personally.

I also want to thank my grandmother, my mother and my sister for always being there for me. Thank you for all the unconditional love and support.

# Abstract

Customer feedback is one of the most powerful tools that any company has in order to evaluate the validity of its products and services. In an ideal world, the company could use the feedback from a client to understand its overall needs and wishes and use this insight to provide a better service or product, making both the company and the customer satisfied. Parallel to this is the immense improvement in technology that has been accomplished in the last decade, some of the most promising being in the area of artificial intelligence in particular in the use of deep neural networks to intelligently retrieve and analyze valuable information from data.

Using the data from the call centers of the partner Telecom Company, this work applies deep learning techniques to this data in order to explore and evaluate the degree of valuable information that could be gained from it. It begins by exploring and deciding what would be a good target for a classification problem and then it experiments with various model architectures to make predictions. The classification problem is the probability of a customer to do churn, that is, to abandon the partner Telco services. From traditional methods, like logistic regressions coupled with a TF-IDF approach, to advanced recurrent neural networks, like bidirectional LSTMs coupled with word embeddings, the various models' show surprisingly similar results. The apparent lack of advantage in using deep learning versus more traditional approaches justifies an analysis of the data provided by the partner Telecom Company. This analysis is performed and attempts to justify the results obtained with the particularities of the original data.

***Keywords:*** Neural Networks, Deep Learning, Natural Language Processing

# Resumo

O feedback dado pelos clientes é uma das ferramentas mais poderosas que qualquer empresa possui para julgar a validade dos seus produtos e serviços. Idealmente, a empresa poderia usar o feedback de um cliente para entender as suas necessidades e vontades de uma forma geral e usar esse *insight* para fornecer serviços ou produtos melhores, deixando tanto a empresa como o cliente satisfeitos. Paralelamente a isto, está o tremendo desenvolvimento tecnológico realizado na última década, sendo um dos campos mais promissores a área de inteligência artificial, em particular o uso de redes neurais *'deep'* para descobrir e analisar conhecimento num conjunto de dados.

Usando o conjunto de dados do *call center* da empresa de telecomunicações parceira, este trabalho tenta aplicar algumas técnicas de *deep learning* a estes dados, a fim de explorar e avaliar a qualidade das informações que poderiam ser obtidas com eles. Começa por explorar e decidir qual seria um bom objetivo para um problema de classificação e depois experimenta várias arquiteturas de modelos de redes neuronais para fazer previsões. O problema de classificação escolhido foi a probabilidade de um cliente fazer churn, ou seja, de abandonar os serviços da empresa parceira. Desde métodos tradicionais, como regressões logísticas acopladas a uma abordagem usando TF-IDF, até redes neurais avançadas, como LSTMs bidirecionais acopladas a *embeddings* de palavras, os vários modelos treinados, surpreendentemente, obtiverem resultados similares. A aparente falta de vantagem em usar *deep learning* em vez de métodos mais tradicionais justifica uma análise aos dados fornecidos pela empresa de telecomunicações parceira. Esta análise foi realizada e tenta justificar os resultados obtidos com as particularidades dos dados originais.

***Palavras-chave:*** Redes Neuronais, Deep Learning, Processamento de Linguagem Natural

# Contents

# List of Tables

# List of Figures

# Acronyms

**AUC** Area Under ROC Curve

**BoW** Bag-of-Words

**CBoW** Continuous-Bag-of-Words

**CNN** Convolutional Neural Network

**CPU** Central Processing Unit

**DCC** Departamento de Ciência de Computadores

**DL** Deep Learning

**DNN** Deep Neural Networks

**FCUP** Faculdade de Ciências da Universidade do Porto

**GPU** Graphics Processing Unit

**LSTM** Long Short-Term Memory

**MLP** Multilayer Perceptrons

**NLP** Natural Language Processing

**NN** Neural Network

**ReLU** Rectified Linear Unit

**RNN** Recurrent Neural Network

**ROC** Receiver Operating Characteristic

**Telco** Telecom Company

**TF-IDF** Term Frequency-Inverse Document Frequency

# Chapter 1

# Introduction

## Contents



This chapter provides an introduction to the context in which this thesis was developed and tries to lay out the motivation that drove the execution of the work. The goals and objectives that were aimed at are also exposed and explained along with the description of the structure of the entire thesis.

## 1.1 Context

Customer feedback is one of the most powerful tools that any company has to evaluate the validity of its products and services. A company can obtain feedback from its clients from a variety of channels and forms, one of which is from its call centers. The information received by a company in its customer support services can be difficult to exploit because usually when someone contacts a call center it is to complain or discuss a specific product or a specific service and that means the information received is also very specific and could be hard to extract general knowledge from it. In an ideal world, the company could use the feedback from a client to understand the customer overall needs and wishes and use this insight to provide a better service or product making both the company and the customer satisfied.

Unfortunately, that is not the case and even though most companies have access to millions of customer's calls and records it can still be difficult to translate this huge amount of data into something actionable.

Parallel to this is the immense improvement in technology that has been accomplished in the last decade, and it has since made it possible to process and extract information from the enormous amount of data that is available. Some of the most promising is in the field of data science and machine learning were the use of Deep Neural Networks (DNN) has provided the tools to intelligently analyze and retrieve valuable information from the data that the companies already have. Deep learning can be used to amplify and improve on previous technologies that were used to understand patterns and behaviors from its clients, and use that information to drive companies' decisions on what path to follow.

In the area of the telecommunications the competition can be very fierce, customers can be very fast to switch between service providers if they encounter any problem and given that the realm where a Telecom Company (Telco) operates in inherently digital it is paramount that any Telco use the new machine learning technologies to increase the value and quality of its services otherwise it will quickly be consumed by its competitors.

## 1.2 Motivation

For a Telco, being able to understand the real needs of its customers is of critical importance, in many situations there are services or products that would improve the customer life but he is not aware that they exist, other times the customer knows of the existence of the product or service but its still hesitant in adhering to it. A company can improve dramatically the success of its publicity and marketing campaigns if she knows precisely whom to target.

Being able to identify customers that are dissatisfied with the services provided or that are in need of an upgrade in its services is tremendously valuable. If a Telco recognizes that a customer is considering abandoning its services, known as a customer doing *churn*, in some time in the future then it can start to take preventive measures like give that particular customer a specific promotion or upgrading its services without charging more for it and in doing so it can retain the customer for a longer period of time which in the end will bring more profit for the Telco. Another situation that can occur is if a customer that uses a product or service and is relatively satisfied with it but would be more happy with something better and is willing to pay for it but for some reason it has not yet made the decision to make the upgrade, if a Telco can recognize this situation it can target this particular customer and actively offer him the upgraded service. These kind of actions improves the value of the customers that the Telco already has giving it an increased revenue without the need to acquire more customers.

The Telco's customer support services are a source of data that can be mined to extract information about its clients. Having the ability to process, analyze and retrieve value for the data generated in a Telco call

center is very important, it can help support and strengthen the decisions made by other departments regarding a specific customer. Doing this automatically using DNN can be a cheap and easy way to deal with millions of records and calls that are generated and that would be very difficult to process and analyze using manual labor. This work tries to do exactly that, it aims to explore the best ways to use the most recent advancements in deep learning and apply them to the data from the partner Telco's call center in order to extract valuable insights.

## 1.3 Objectives

The partner Telco possesses data that is produced in its call centers and is interested in exploring data science techniques to evaluate what and how much value can be extracted from this data. In order to do this, a first step is to explore the data itself and analyze some statistics about it. Given that one of the objectives is to ultimately get actionable value from these textual reports it is necessary to define in what categories the customers are going to be classified based on the interaction with the call center.

With the idea of using the textual reports generated by the call center operators to make predictions about the customers, it is important to begin with some solid and more traditional approaches. There have been other studies [Kamath et al., 2018] that made the comparison between traditional and deep learning approaches but a big part of this work is to use the specific data from the partner Telco so it is important to try traditional methods as well. The next step is then to apply methods like the Bag-of-Words (BoW) and the Term Frequency-Inverse Document Frequency (TF-IDF) to the texts in order to create features and then use these features to make predictions using a logistic regression model.

The most challenging step is to use deep learning techniques to generate predictions for the *churn* problem. Since this work has a component of exploration, several different architectures need to be tested to see which ones give the best results. Models based on multilayer perceptrons, based on convolutional neural networks and models based on recurrent neural networks need to be trained to gain a sense of the differences between them and the impact that they have on the results.

The end objective is to, after testing which of the various models perform the best, use it on the reports of the Telco call centers to obtain predictions on which customers are more likely to churn in the six months after the interaction.

## 1.4 Contributions

The main contributions of this work are as follows:

- The Telco partner gained insights about the advantages and disadvantages of applying deep

learning approaches to the data used in this work, which may be relevant in decisions regarding future projects using the reports from the call centers.

- The partner Telco ended with more than one model that predicted if a customer was going to do churn in the next six months with reasonable results. These predictions were based solely on a single interaction of the customer with the Telco's call centers.
- A new case study about the use of deep learning techniques with real-world data is presented, where it shows that the performance of these advanced techniques can be halved when working with a dataset built with practical limitations and constraints.

## 1.5 Thesis Structure

The remainder of this thesis is structured as follows:

- Chapter 2, provides a literature review on the main techniques and subjects used in this work. It presents an overview of the neural networks framework and details the model architectures used in this work. It also explains traditional methods like BoW and TF-IDF.
- Chapter 3, presents the work done on the creation of the dataset and subsequent analyzes and provides a description of the problem tackled in this thesis.
- Chapter 4, presents and explains the experimental work performed, the models trained and the results obtained. The chapter ends with a discussion of the results.
- Chapter 5, concludes the main ideas about this work and gives some suggestions for further work.

# Chapter 2

# Literature Review

## Contents



The purpose of this chapter is to summarize the techniques most relevant to this work. It presents a brief description and explanation of methods and model architectures that are currently being used in the field of machine learning and that were implemented in this thesis. By the end of this chapter, it should be easier to understand some concepts and ideas related to deep learning and the reasoning behind some of the decisions made in this work.

## 2.1 Traditional Methods

When dealing with the classic problem of document representation there has been several methods and approaches over time. Many of them tried to extract a set of features from the words of the documents and then use these features as input to a machine learning model so to obtains the predictions. The extracted features and their representation play a major role in the accuracy given by a classifier [Jurafsky and Martin, 2009] therefore it is important to choose the best way to perform this task.

In this work the methods used to generate the features, to give to the classifier were the following:

- Bag-of-Words (BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)

These two techniques can be seen in sections 2.1.1 and 2.1.2. The features produced were used to make predictions with the help of a logistic regression model, described in section 2.1.3.

### 2.1.1 Bag-of-Words

One of the simplest forms of feature extraction to use with a classifier is the Bag-of-Words (BoW) method [Harris, 1954], [McTear et al., 2016]. It consists of transforming a text or document into a vector with size equal to the vocabulary where each element of the vector contains the number of times a specific word appears in that particular document. An illustrative example can be seen in figure 2.1.

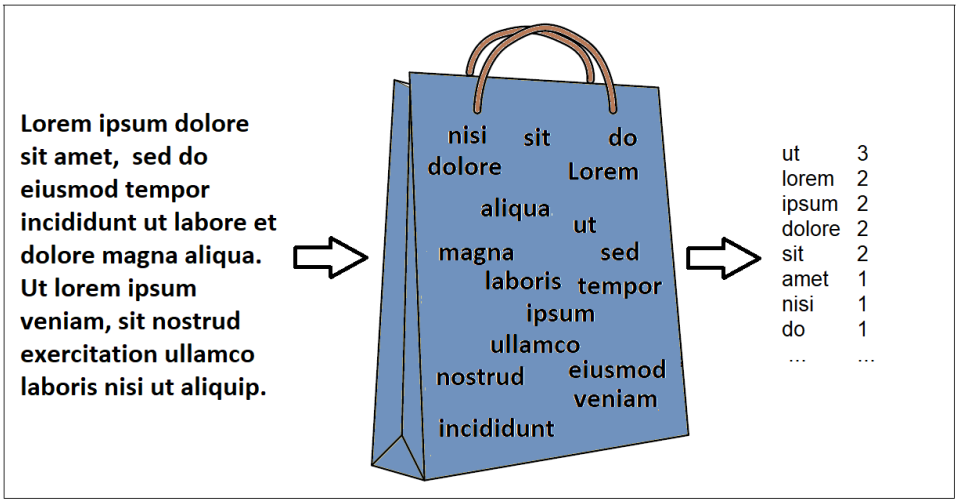



Figure 2.1: Bag-of-Words example[1]

This approach accounts for the number of times the word appears but disregards the order in which the words appear and the entire document grammar which limits its ability in representing the words meaning. The size of the vocabulary also plays an important role because if it is too small it will be unable to represent correctly all the documents but if it is unnecessarily big it can have an impact on the performance of the classifier.

[1] Inspired in [Jurafsky and Martin, 2009]

### 2.1.2 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) [Salton and McGill, 1986] is a technique similar to BoW but that tries to be more sophisticated and accurate by weighting the various words in the vocabulary. TF-IDF is the product of two terms, each term capturing one of these two ideas:

- Term Frequency [Luhn, 1957]: It is, basically, the frequency of the word in the document. This reflects the intuition that if a word appears many times in a document then it must be important. It also helps to control the weight of words for very large documents versus very small documents. See equation 2.1 where *t* denotes a term and *d* denotes a document.

$$TF(t,d) = \frac{count(t,d)}{count(all\_terms,d)} \tag{2.1}$$

  Usually, it is important to down-weight the raw frequency of a word a bit, because if it appears ten times it does not make that word ten times more relevant to the meaning of the document. Resulting in the slightly changed definition for the term frequency weight. Equation 2.2.

$$TF(t,d) = \begin{cases} 1 + \log\ count(t,d) & if\ count(t,d) > 0 \\ 0 & otherwise \end{cases} \tag{2.2}$$

- Inverse Document Frequency [Jones, 1972]: It attributes a higher weight to terms that occur only in a few documents. This reflects the intuition that words that occur frequently across the entire collection of documents are not as helpful. See equation 2.3.

$$IDF(t) = \log\left(\frac{\#\ documents}{\#\ documents\ containing\ t}\right) \tag{2.3}$$

We can then obtain the TF-IDF weight for the term *t* in document *d* by doing $TF(t,d) * IDF(t)$.

### 2.1.3 Logistic Regression

Logistic regression is a model used for classification, where the probabilities for the different outcomes of a single sample are modeled by the logistic function [Defazio et al., 2014]. The logistic function is represented by the equation 2.4 where *e* is the natural logarithm base, $x_0$ is the x-value of the sigmoid's midpoint, *L* is the curve's maximum value, and *k* is the logistic growth rate or steepness of the curve.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{2.4}$$

## 2.2 Word Embeddings

Word embedding is a popular method of representing a term or word or even an entire sentence by a vector of continuous real values, it is used widely in the majority of Natural Language Processing (NLP) tasks and is practically mandatory when working with deep neural networks to solve complex problems.

There are many approaches to create these vectors, one of the most popular being Word2Vec [Mikolov et al., 2013], this method consists of training a neural network on a large corpus of text. It uses one of two predictive tasks to calculate the embedding:

1. SkipGram: It is a task in which, given a word, it tries to predict the words around it, basically it tries to predict the context where the given word appears.
2. Continuous-Bag-of-Words: It is the reverse of SkipGram prediction, given a context it tries to predict the word that belongs to this context which translates to basically, given a short sentence where the middle word is missing the model then tries to predict that missing word.

These tasks have the purpose of providing the embedding with an understanding of context. This means that the embedding representation of the word will contain different meanings depending on the context in which it appears.

Another popular method of calculating word embeddings is GloVe [Pennington et al., 2014]. GloVe is a method were the vector representations for words are obtained by calculating how frequently words co-occur with one another. The training is done using an aggregated global word-to-word co-occurrence statistics matrix with non-zero entries. The intuition behind this idea is the observation that ratios of word-to-word co-occurrence probabilities could encode some form of meaning.

Yet another method of creating word embeddings is to use the FastText [Bojanowski et al., 2016] word representations. This one follows a different approach where it decomposes a word in sub-words made of only a few characters and them attributes a vector to each sub-word in which the original word is then represented by the sum of the vectors of the corresponding sub-words. This approach helps to deal with vocabularies that are very large and contain many rare words, it also solves the problem of out-of-vocabulary words elegantly and efficiently.

## 2.3 Deep Learning

The concept of neural networks [McCulloch and Pitts, 1943] has existed since the middle of the past century but it was never considered very practical. It was only after critical advancements in hardware architectures that were made available at the very end of the XX century, that brought with it an increase in

performance that allowed for this field to reborn. Coupled with the creation of new algorithms, techniques and methods that gave better results for some complex problems a new age was created for the neural networks field leading to an expansion and subsequent re-branding as deep learning.[Goodfellow et al., 2016]

Another critical factor that affected not only the deep learning methods but the entire field of machine learning was the huge volume of labeled data that appeared with the proliferation of computers and the new digital era.

Deep Neural Networks (DNN) has been used and applied across several kinds of applications such as NLP, image data processing, speech recognition, audio processing, and many other well-known applications [Pouyanfar et al., 2018]. It has revolutionized the way people see artificial intelligence and its ability to learn very complex tasks from zero human knowledge [Silver et al., 2017] has made it reach world news.

For this work, three main architectures of deep neural networks were studied:

- Multilayer Perceptrons (section 2.3.1)
- Convolutional Neural Networks (section 2.3.2)
- Recurrent Neural Networks (section 2.3.3)

### 2.3.1 Multilayer Perceptrons

Multilayer perceptrons [Rosenblatt, 1961] also known as feedforward neural networks can be described as a series of layers connected sequentially between themselves where each layer has several perceptrons also known as neurons and each neuron in one layer is connected to every other neuron of the next layer. Figure 2.2 has an illustrative example.

The first layer or input layer receives data and the last layer or the output layer provides the result of the computation made within the network. Between the input layer and the output layer, there can be one or more layers called hidden layers. The flow of the computation goes from the input to the output in a unidirectional fashion without any cycle or feedback loop, hence the name feed-forward neural network. For each connection between the neurons is associated a weight and when the value from a neuron passes to the next layer it is applied a weighted linear summation, using the weights from the connections and then a non-linear activation function. Some popular activation functions are the sigmoid function (equation 2.5), hyperbolic tangent function (equation 2.6) and the rectified linear function (equation 2.7). Figure 2.3 shows an example of each activation function.

$$sigmoid(W^t x) = \frac{1}{1 + e^{-W^t x}} \tag{2.5}$$

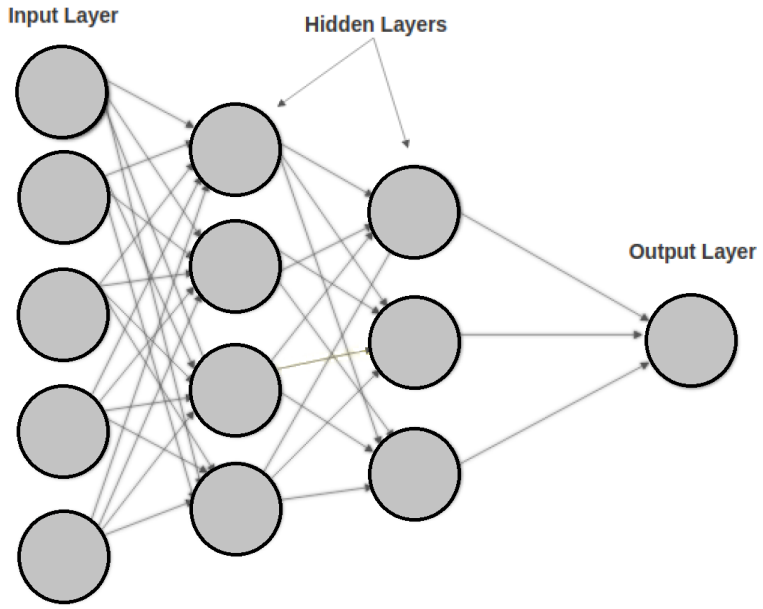



Figure 2.2: Multilayer perceptron example[2]

$$tanh(W^t x) = \frac{e^{W^t x} - e^{-W^t x}}{e^{W^t x} + e^{-W^t x}} \tag{2.6}$$

$$ReLU(W^t x) = max(0, W^t x) \tag{2.7}$$

During the training process, the neural network the connection weight are changed using a technique called backpropagation [Rumelhart et al., 1986]. The aim of this technique is to update the weights based on how much difference there was between the expected result and the actual output. The updating of the weights are applied recursively, layer by layer, from the output layer to the input layer in a backward manner.

### 2.3.2 Convolutional Neural Networks

A convolutional neural network [Lecun et al., 1998] is a particular type of feedforward neural network. CNNs usually are more associated with computer vision where they had great success [Krizhevsky et al., 2012] since then they have been used to perform other tasks, including sentence classification [Kim, 2014].

The convolution operation consists of applying a sliding window, named filter or kernel, over a matrix, while the window is sliding its values are multiplied element-wise with the ones in the matrix and then

[2]Inspired in [Kamath et al., 2018]

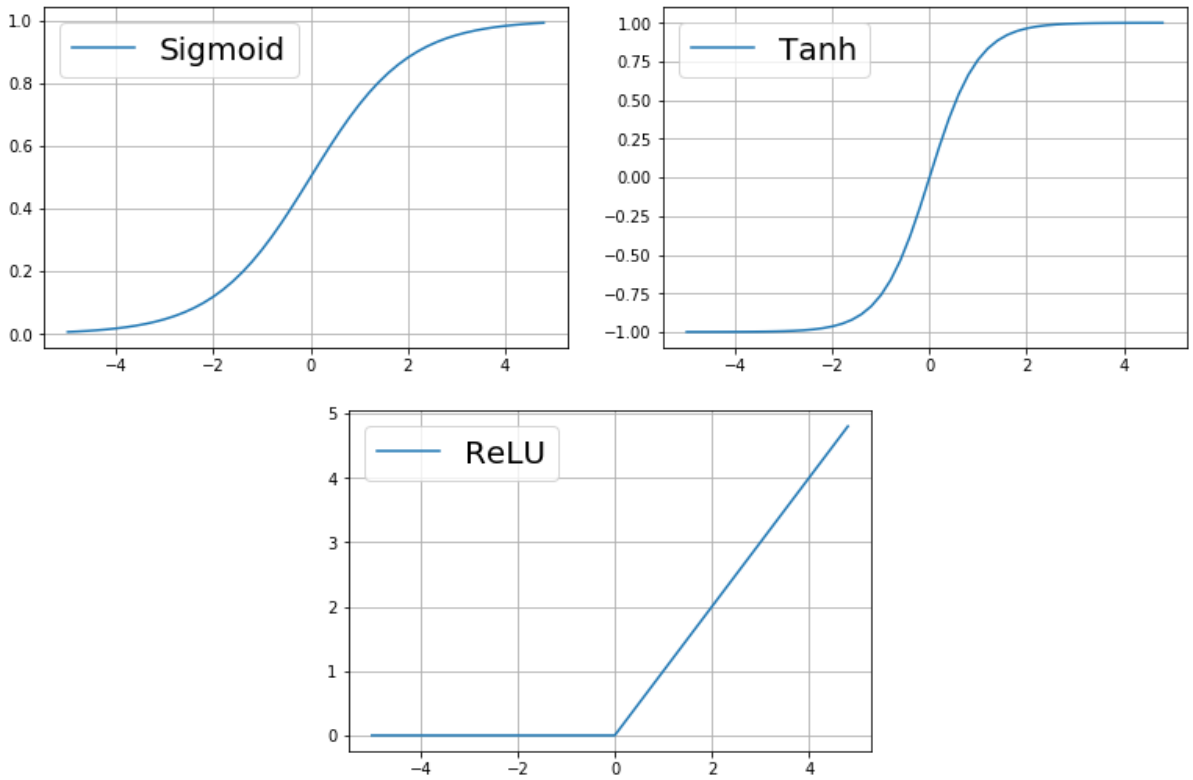



Figure 2.3: Graphs of the activation functions Sigmoid, Tanh and ReLU

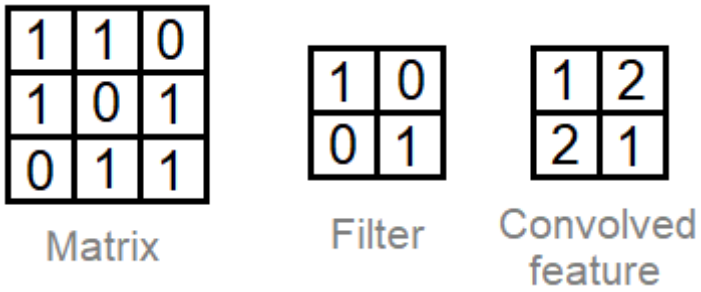



Figure 2.4: Representation of a convolution of a 3*3 matrix with a 2*2 filter[3]

they are summed up. Doing this for every element gives the total convolution on the full matrix. Figure 2.4 shows a convolution of a 3*3 matrix with a 2*2 filter and the produced feature map.

CNNs, usually consist of convolutional layers followed by a subsampling layer called pooling and normally end with a dense layer that maps to the output classes for classification. As the activation function CNNs tend to use non linear functions such as tanh (equation 2.6) or ReLU (equation 2.7). Figure 2.5 illustrates the steps of a convolution neural network.

The process for the convolution layer involves a kernel sliding through the input sequentially and aggregating the information in so-called feature maps. The kernel which has the same number of

[3]Inspired in [Kamath et al., 2018]

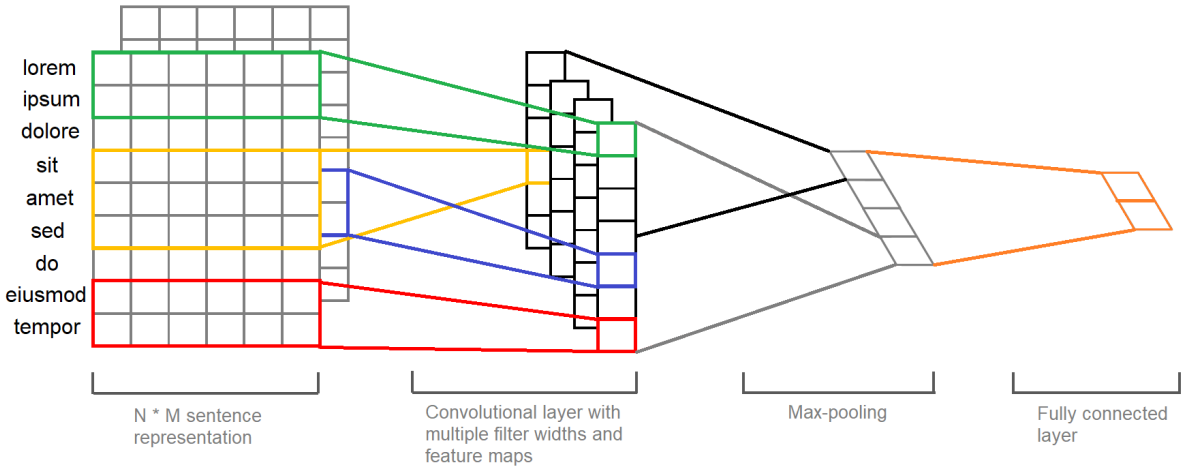



Figure 2.5: Convolutional neural network example[4]

dimensions as the input but is smaller in size joins together adjacent input values and produces a feature that contains localized information, as the kernel slides through the input it calculates localized information for the entire input. Doing this procedure multiple times allows the network to capture different kinds of features for a particular position. The extracted features possess a strong spatial correlation because the method enforces a local connectivity pattern between adjacent neurons and between adjacent layers.

Following this part is typically applied a layer of max pooling that condenses these feature maps and reduces their size which allows the network to train faster while also controlling overfitting [Pouyanfar et al., 2018]. Doing so also permits to find particular features independently of their position which helps the network deal with inputs that only vary in small positional differences.

For a classification problem the network typically has a fully connected layer with softmax, as the final layer.

### 2.3.3 Recurrent Neural Networks

Unlike the previous networks, RNNs [Elman, 1990] can not truly be considered feedforward neural networks because their architecture contains a feedback loop in its internal memory. As the inputs are being consumed the hidden state or the memory of the network remembers the previous elements consumed in the sense that it reuses the same internal weights for all the inputs. This makes RNNs perfect to work with sequential data like text or time-series data. Figure 2.6 presents a general structure of a regular unidirectional RNN.

When processing a sequence, RNNs receive all the inputs at the same time but the computation is

[4] Inspired in [Kamath et al., 2018]

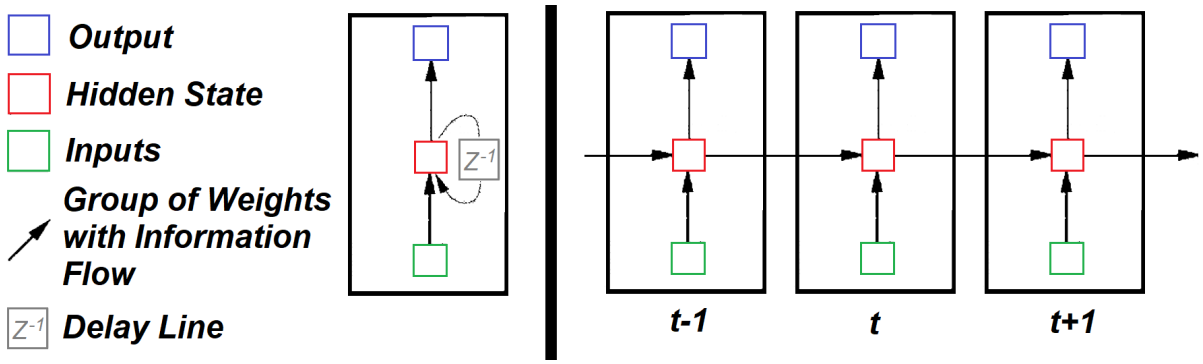



Figure 2.6: Regular unidirectional RNN, folded in the left and unfolded in the right[5]

performed at one element at a time. For every element of the input, it gives an output which means that the output of an RNN has the same length as the length of the input sequence. While the various elements are being processed the internal weights are being reused, because of that, it has an implicit short memory, this means that when the time comes to process a new element the network remembers the information from the previous ones creating a context in which to evaluate this new element.

In theory, RNNs could remember sequences of any length and make use of that information but because it is sensitive to the vanishing and exploding gradients problem [Glorot and Bengio, 2010] it is limited to remember only a few steps.

A powerful way to enhance this architecture is to consider a bidirectional RNN [Schuster and Paliwal, 1997] which is can be thought of as two RNNs placed one beside of the other where the first reads the input from right to left and the second reads the input from the left to the right, allowing the network to create an information context in both directions. In practice, bidirectional RNNs create twice the outputs of a normal RNN because, in reality, it is receiving the input two times. If the output from both directions is combined it means that the evaluation of a particular element in from the input sequence can depend on the previous and next elements of the sequence providing a much stronger understanding of the meaning of the element.

Another technique used it to stack various RNNs one on top of the other creating what are called deep recurrent neural networks [Pascanu et al., 2013]. Accomplishing this task is not as simple as it may appear on the surface as the RNNs are already considered deep neural networks. Attempts to achieve build deep RNNs have already been done more than one time ([Schmidhuber, 1992], [El Hihi and Bengio, 1995]). They all are based on the hypothesis that a deep, hierarchical model can represent functions at an exponentially more efficient way than normal shallow one [Bengio, 2009].

[5]Inspired in [Schuster and Paliwal, 1997]

#### 2.3.3.1 LSTM

Long Short-Term Memory networks [Hochreiter and Schmidhuber, 1997] are an advancement in RNNs in which it tries to solve some of its problems, in particular, the vanishing and exploding gradients problem. RNNs are built around one block that repeats itself by a feedback loop but LSTMs are significantly more complex having four blocks that perform different tasks. Figure 2.7 shows an illustration of the blocks present in an LSTM unit.

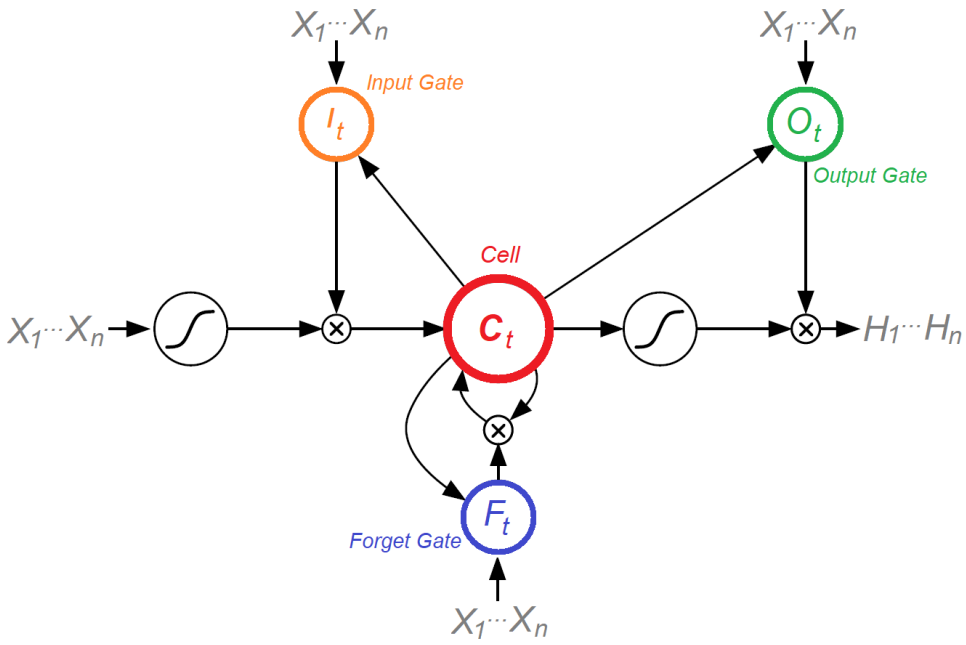



Figure 2.7: LSTM blocks diagram[6]

Every block of the LSTM has a hidden state or memory that it uses to perform its function allowing it to take different actions on different time steps. One block of the LSTM is the *Forget Gate* and its function is to decide what part of the LSTM *Memory Cell* is going to be thrown away which permits the network to select what information is not useful for a given time and forgets it. The *Input Gate* in conjunction with an *tanh* function decide what new information from the input is going to be added to the *Memory Cell* of the LSTM. The last block, the *Output Gate*, chooses which parts of the *Memory Cell* are going to actually be in the output in that particular time step. This process of actively deleting information and updating information helps in controlling the problem of vanishing and exploding gradients that afflicts the RNNs in general.

Like with the normal RNNs, LSTMs can be used in a bidirectional fashion. A bidirectional LSTM [Graves and Schmidhuber, 2005] provides all the advantages of the bidirectional RNN with the added bonus that

[6] Inspired in [Graves et al., 2013]

it possesses all the benefits of LSTMs. This makes them one of the most powerful architectures of neural networks that can be used in problems of Natural Language Processing.

### 2.3.4 FastText

FastText [Joulin et al., 2016] is a model architecture based on deep learning created and released by the Facebook AI research lab. Although it's main feature is the ability to produce word embeddings, the FastText model is also capable of making text classification. To make a prediction, it feeds a linear classifier with an embedding that is the representation of the text on which the prediction is going to be based upon. This latent text embedding is created by making an average of the word embeddings that compose the text, which in turn are produced by summing the individual embeddings of the character n-grams that constitute each word. In the end, a hierarchical softmax is applied to speed up the class prediction task.

Figure 2.8 ilustrates the model architecture of fastText for a sentence with *n* n-gram features *X1*, ... , *Xn*. The n-grams are embedded and their mean is calculated to create the hidden variable.

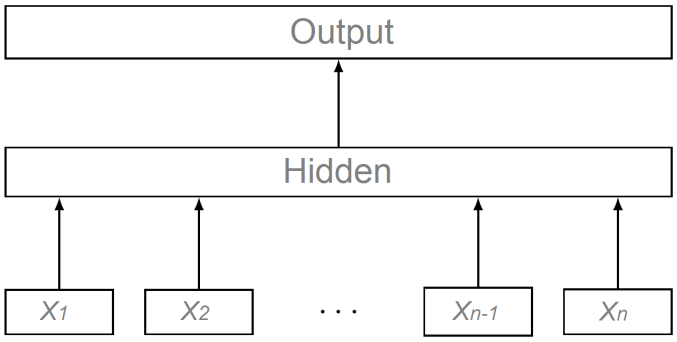



Figure 2.8: FastText model architecture[7]

Given the relatively simple approach that FastText uses coupled with some efficient algorithms to save the internal representations of the n-grams, it manages to be much faster than other deep learning models.

## 2.4 Summary

The velocity at which new developments about deep neural networks appear is sometimes overwhelming. Therefore, considerations made about the state-of-the-art for them have a very limited time window.

[7] Inspired in [Joulin et al., 2016]

Nevertheless, this chapter tries to provide basic knowledge about the main techniques and architectures relevant for this thesis.

At the end of this chapter, it should be easier to understand chapter 4, where all the experimental work is described and detailed without major difficulties.

# Chapter 3

# Dataset and Problem Definition

## Contents



This chapter focuses on presenting the work done in the creation and analysis of the dataset, containing three years' worth of reports. It also states the problem on which this thesis focuses.

## 3.1 Dataset

The work presented in this thesis relies on the data collected in the partner Telecom Company (Telco)'s call centers during the last three years but there was no clear and simple dataset, this meant that a significant portion of the work was the creation and definition of the dataset including the labels or classes in which the models were going to predict and classify the data.

### 3.1.1 Dataset Construction

The dataset used was constructed using the reports made in a call center and associating them to the client that made them. Originally there was a huge amount of reports, around 89 million, but after filtering, processing and linking the reports to a specific client of the partner Telco the total number of reports reduced to approximately 5 million which was still a very large. The table 3.1 shows the number of reports after some of the steps taken during the construction of the final dataset.

| | **Number of Reports** |
|---|---|
| All | 89 067 491 |
| That contain text | 30 501 302 |
| From Telco clients | 16 011 618 |
| After preprocessing | 5 172 980 |

Table 3.1: Number of reports after each stage

A large amount of the reports did not have a textual description of the situation and therefore could not be used, so the first step was to remove any report that didn't contain a text and this step alone eliminated more than 65% of all the reports leaving only around 30 million.

The reports made in the call center were based on phone calls from any person, including individuals that were not clients of the Telco at that particular time. Because one of the objectives of this work was to use the data from the Telco to generate actionable predictions on its clients, the next mandatory step was to associate each report to a specific user that was registered in the Telco database at the time of the report creation. Since many reports originated from phone calls made from persons who were not, at the time, customers of the Telco, another 16% of the total reports were lost, leaving approximately 16 million.

Before the texts could be used in the training of the models three preprocessing steps needed to be done and are briefly summarized below:

1. Anonymization: Because of obvious legal and ethical reasons, a step of anonymization was taken to remove all references to any particular person, also, leaving personal or unique information about the clients withing the texts could potentially create data leakages or make the models biased for some particular people or towards a specific geographic area. Therefore this was an important step to ensure that the models were as general as possible. A more detailed description of this step can be found in section 3.1.1.1.
2. Minimum length: This work has a component of exploration but also it needs to train models to make predictions with reasonable results, therefore, some care was taken to guaranty that the texts had some quality and contained relevant information withing it, therefore all reports that were simple short machine logs and all reports which texts had less than one hundred characters were eliminated.
3. Duplicates removal: For similar reasons as in the previous step, having duplicate reports in the dataset was redundant and useless, besides, it could potentially create some bias in the models or give too much importance to a particular text just because it had too many duplicates, skewing the models towards a wrong prediction.

The final set of reports had a size of approximately 5 million. The main idea behind these steps of filtering and preprocessing was to end up with a number of texts, written by the call center operators, with sufficient information and detail so that it could be said with some confidence that it carried some amount of value.

#### 3.1.1.1 Anonymization

One of the very first preprocessing tasks done to the data was anonymization. This was necessary to fulfill the privacy requirements that the Telco needs to uphold. Because of the large amount of data, it was impractical to make any kind of manual anonymization, therefore, it was required to find a method that could be done automatically and with a minimum amount of human interference.

In order to maintain the maximum information in the texts, all information that needed to be removed was replaced with a tag that identified the type of information that was there, this way, for example, the individual name of a person would be removed but the information that there was a name there would be kept.

The task of entity recognition is an ongoing research task and as such it would not be reasonable to expect perfect names anonymization, since this task, although important, was not the objective of this work and because of time constraints it was decided to approach the anonymization step with a technique that could be relatively simple but efficient. A dictionary of names[1] and surnames[2] was used to match, remove and replace all names from the clients with a tag, although it can not be guaranteed that the removal of names was perfect, from a simple inspection of some texts it was clear that the end results were relatively good.

Personal names were not the only thing that had to be removed, financial information, phone numbers, e-mails, dates, and other numerical values needed also to be cleared. For this part, since this type of data follows a relatively strict format, it was created a series of regular expressions to match, remove and replace, these pieces of information, with a tag. Table 3.2 contains some of the information replaced along with the tag and regular expression used.

#### 3.1.1.2 Target Label Construction

To explore the intrinsic value that these reports had, it was required to attribute a label to a text that had meaning and at the same time could be useful to the Telco. There was no simple way or straight forward method to classify the reports and it was required some exploration of the Telco's database and

[1] https://www.irn.mj.pt/sections/irn/a_registral/registos-centrais/docs-da-nacionalidade/vocabulos-admitidos-e/downloadFile/file/NomesAdmit.pdf?nocache=1214922851.67

[2] https://pt.wikipedia.org/wiki/Lista_de_apelidos_de_família_da_língua_portuguesa

| Information | Tag | Regular expression |
|---|---|---|
| Name | #NAME | |
| Phone number | #PHONE | \b(?:9[1236]\d{7}\|2\d{8})\b |
| Fiscal number | #NIF | nif\s*:?\s*\d{9} |
| E-mail | #EMAIL | [A-Za-z0-9\.\+_-]+@[A-Za-z0-9\._-]+\.[a-zA-Z]* |
| Date | #DATE | \d+[/-]\d+[/-]\d+\|\d{2}[/-]\d{2} |
| NIB or IBAN | #NIB | (?:nib\|iban)\s*[ \-:=]?\s*(?:[a-z]{2}[ \-]?[0-9]{2})?(?:[ \-]*\d{3,5}){4,6}(?:[ \-]*\d{1,3})? |
| Client ID | #CLIENT_ID | [sc]\d{9}\|u\d{8} |
| Service request | #SR_NUMBER | \d-\d{11}\|sr\d{8} |
| Phone's IMEI | #IMEI | \b\d{15}\b |
| Hour | #HOUR | \d{1,2}h\d{0,2}\|\d{1,2}:\d{2}(?::\d{2})? |
| Price | #PRICE | \b\d+[,\.]\d{1,2}\b |

Table 3.2: Regular expressions used to replace sensitive information by generic tags

posterior data processing in order to attribute some labels to the reports. With the intent of making some preliminary analysis on the data and evaluate what would be a proper target class/label to be used during the model testing stage the following five classes were created:

- **Upsell**: This classified a report with the label true if the customer increased its service bill (not including premium services) by at least two euros within the next six months after the phone call, and false otherwise.
- **Premium**: This classified a report with the label true if the customer adheres to a premium service within the next six months after the phone call, and false otherwise.
- **Churn**: This classified a report with the label true if the customer left the Telco within the next six months after the phone call, and false otherwise.
- **Refidelization**: This classified a report with the label true if the customer renews its service fidelization plan with the Telco within the next six months after the phone call, and false otherwise.
- **New report in 30 days**: This classified a report with the label true if the customer made a complaint (a new report) within the next thirty days after the phone call, and false otherwise.

All of these classes were important actions that a customer could take and were impactful to the Telco. As the table 3.3 shows, all had a significant percentage and were actions that the customers often took after contacting the Telco's call centers.

After careful consideration of what would be more important to the Telco and more interesting to study and after making some preliminary models and evaluating the results it was decided that this work would focus on the Churn class and leave the other ones in the background. Although all classes ended being

| Action | Percentage |
|---|---|
| Upsell | 18.0% |
| Premium | 16.5% |
| Churn | 20.8% |
| Refidelization | 16.0% |
| New report in 30 days | 50.9% |

Table 3.3: Percentage of reports with the label True for the various classes

tested in all major model architectures the majority of the work and effort went to testing and exploring with the churn class.

### 3.1.2 Texts distributions

The approximately five million reports that constitute the dataset used in this work were produced during the last three years, it spans over thirty-seven months between January of 2016 and January of 2019. It means that on average, there are more than 135 thousand reports per month which gives a solid representation of the customer base that the partner Telco has and therefore it brings some confidence to the statistical relevance of this work. Figure 3.1 presents the distribution of the number of reports per month.

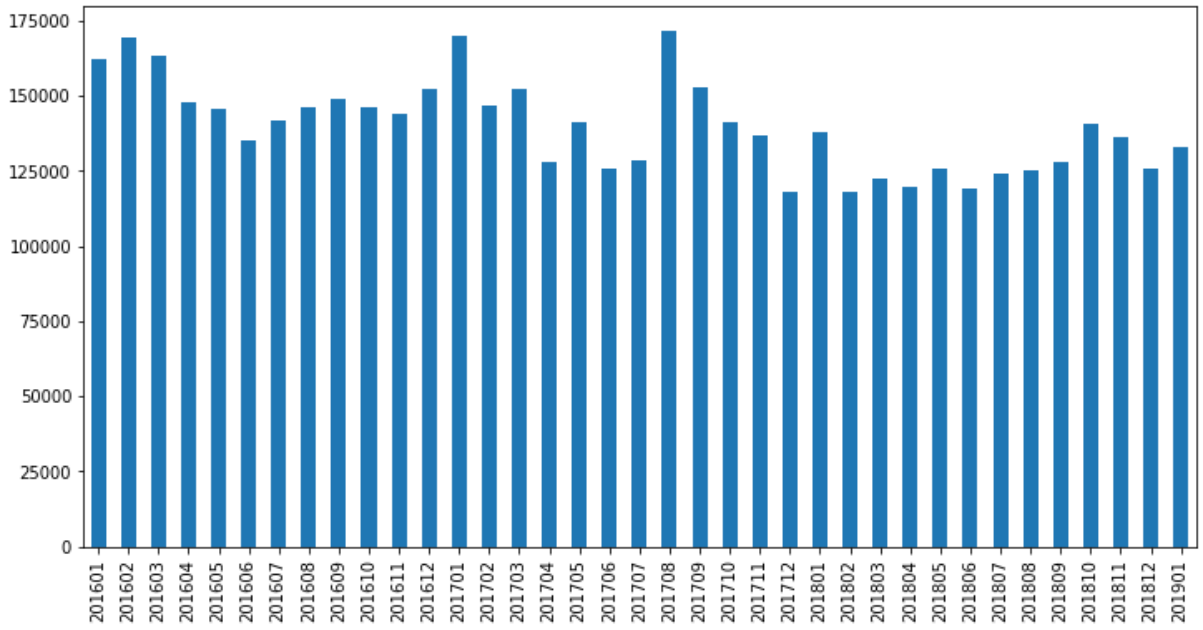



Figure 3.1: Number of reports per month

The texts associated with the reports had a huge variance in quality and size, because these texts were written by human operators and because every person has a different style to express itself, the texts could be very short and succinct with only the most relevant information or could be very long with a detailed description of the phone call and of the customer complain. As can be seen in figure 3.2, that shows the distribution of the number of characters in the texts per month, there is a significant portion of very big texts, some being more than three thousand characters long.

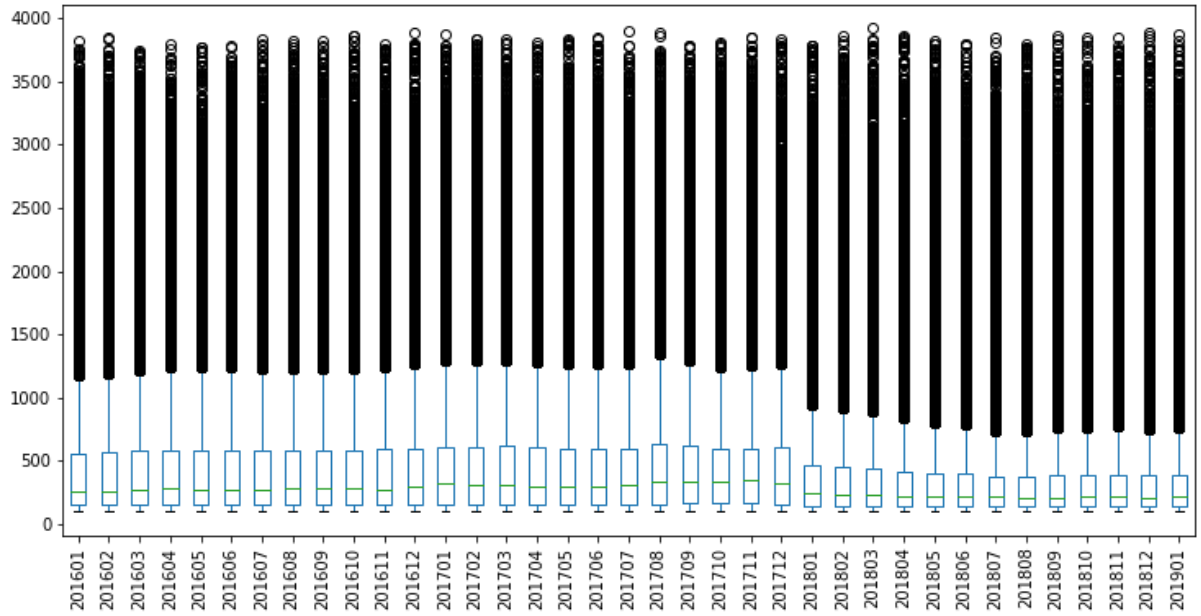



Figure 3.2: Distribution of the number of characters in a report per month with outliers

Although there were some texts very big, the median was relatively small, meaning that the average operator wrote less than four hundred characters per report text, this can be observed more clearly in figure 3.3 where the outliers were removed in order to have a better view of the distribution of the number of characters per report around the median value.

The length of the texts is important in particular when dealing with neural networks, because the texts usually are consumed in an input layer of embeddings this means that very large texts will need to be truncated in order to satisfy the limitations on the size of the embeddings input layer, to be noted here that it is not a limitation on the size of the embedding per word but the length of the input layer.

## 3.2 Problem Definition

Although the objective of this work has a component of data exploration, the problem to tackle during the process of experimenting with the various deep learning techniques needs to be defined clearly.

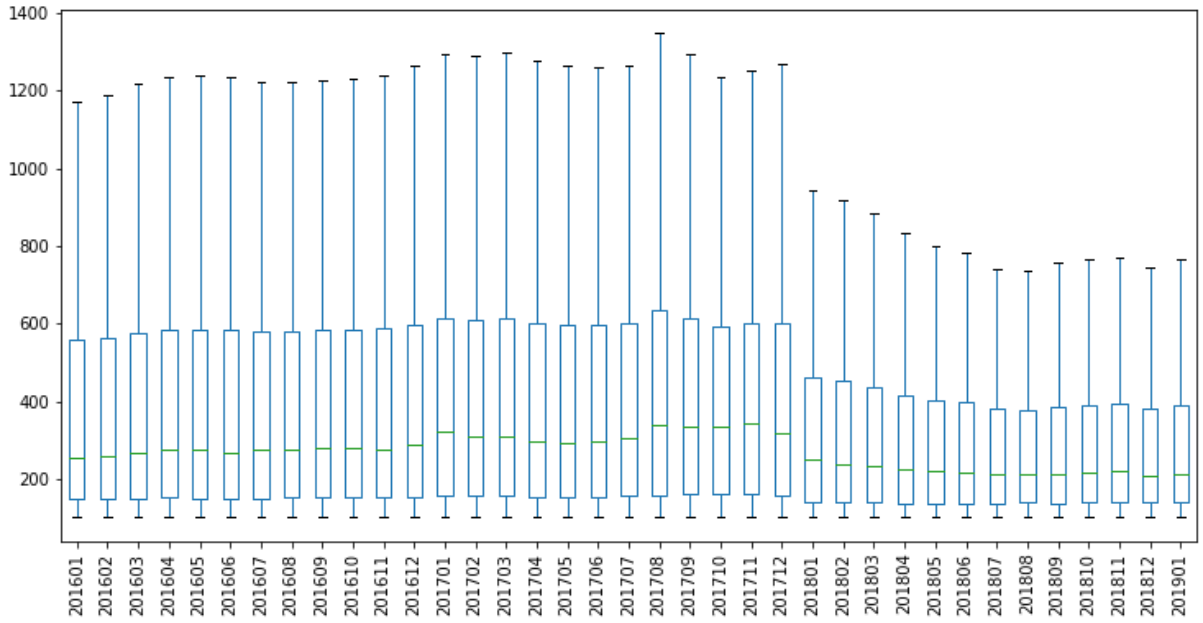



Figure 3.3: Distribution of the number of characters in a report per month without outliers

The problem type chosen for this work is a classification problem. The texts contained in the reports are associated with a customer from the partner Telco. An important goal is to use these texts, representing the interaction of the customer with the Telco, to classify the customer with a label that gives an indication of some particular actions that he or she could potentially take in the future.

In Section 3.1.1.2, it is explained that several target labels were produced for the texts in the dataset. Each target label represents an action from the customer - to churn, to buy some premium service or others. All of them have their own merits, but in the end, the target label chosen to use during the experimental evaluation of our work was the churn label.

Having the ability to predict if a customer is going to churn is, arguably, of tremendous importance. After some discussion with the partner Telco we concluded that if this work could retrieve value from this dataset in regard to predicting the churn of a customer, that would be very relevant. This value could later be integrated with other methods and models, and help with the Telco customer retention.

## 3.3 Summary

In this chapter, a description of the construction of the dataset was given and the problem is defined giving the tools to understand the purpose of the next chapter where the results are exposed and analyzed.

# Chapter 4

# Experimental Work and Results

## Contents



This chapter focuses on presenting all the experimental work that was done and give a description of the models build and trained using both traditional methods and deep learning methods. The results are presented for each model architecture and parameterization tested. The chapter ends with a discussion of the results.

## 4.1 Implementation

The work done was developed with the python language [Rossum, 1995], it is one of the most relevant and used programming languages in the field of data science and deep learning in particular. Python provides access to a myriad of libraries and packages that helps with the most common tasks one needs to do while working in data science. The python programming language is also very easy to install and use, and integrates perfectly with TensorFlow [Abadi et al., 2015].

TensorFlow in conjunction with Keras [Chollet et al., 2015] is the library that was used to create and train all the deep learning models. Using it is possible to, basically, make all the architectures that one could think of in a relatively easy and fast way, it also provides some mechanisms to analyze and follow up during training using the visualizing tool TensorBoard that it is incorporated within TensorFlow. All the different deep learning models developed during this work were made with the help of TensorFlow and Keras.

Worth of note is also the library scikit-learn [Pedregosa et al., 2011], it contains functions to perform tasks for preprocessing data, training models and evaluating the results of said models. It was used extensively in all experiments that were made and it was the library chosen to train and evaluate the models based on traditional approaches. During the preprocessing of the data used later in the traditional models a list of stopwords was needed and the one used came from the NLTK [Loper and Bird, 2002] library.

Computation was carried out using a remote server with two Intel Xeon CPU E5-2660 @ 2.60GHz and 400GB of RAM. Because this machine was shared between several other users, that were also using it to perform data science tasks, the amount of time that a model took to train was very erratic and very dependent on the amount of work that the other users were doing. Therefore it was not possible to collect any relevant time statistics for the models trained.

## 4.2 Evaluation Metrics

When trying to compare models and their results its important to choose the evaluation metric very carefully because it might ruin the conclusions if a wrong metric is used. In this work, two metrics were used extensively:

- F1-score
- Area Under ROC Curve (AUC)

These were chosen because they are resistant when dealing with an imbalanced dataset and because when working with a binary target class, as it is done in this work, they do not favor some specific orientation, like predicting everything true or everything false.

### 4.2.1 F1-score

The F1-score metric is composed of the precision and recall, both of which are metrics also. The precision metric tries to answer the question "How many selected items are relevant?" and recall metric tries to answer the question "How many relevant items are selected?". They are complementary metrics and

usually increasing the performance in one reduces the performance in the other. The way precision and recall are calculated can be seen in equations (4.1) and (4.2) respectively.

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives} \tag{4.1}$$

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives} \tag{4.2}$$

The F1-score aims at levering these two metrics by calculating the harmonic mean of both, this means that to have a high value in the F1-score, both the precision and the recall need to be high too. The exact formula to the F1-score can be found in equation 4.3.

$$F1 = \frac{2\ *\ recall\ *\ precision}{recall\ +\ precision} \tag{4.3}$$

### 4.2.2 AUC

The AUC [Bradley, 1997] metric is the value of the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve represents the ability of a model at all classification thresholds, it is the graph of the true positive rate (a.k.a recall) against the false positive rate (equation 4.4).

$$False\ Positive\ Rate = \frac{False\ Positives}{False\ Positives\ +\ True\ Negatives} \tag{4.4}$$

Since the AUC is the entire two-dimensional area underneath the ROC curve it basically provides an aggregate of the performance of a model across all thresholds. It has one important feature, that it is scale-invariant, it means that it takes into account how the predictions are ranked in the order of probabilities instead of the absolute values of the predictions. This can be fundamental when there is a need to compare models while trying to predict a relatively rare event. The figure 4.1 show examples of a typical ROC curve and its corresponding AUC.

## 4.3 Traditional Models

The main objective of this work was to experiment with deep learning approaches but to be able to correctly evaluate their performance it was fundamental to find some baseline results to compare with. For that, two traditional approaches were tried, the first one used a simple Bag-of-Words (BoW) to generate features and use a logistic regression model to train and make predictions. The other approach, slightly

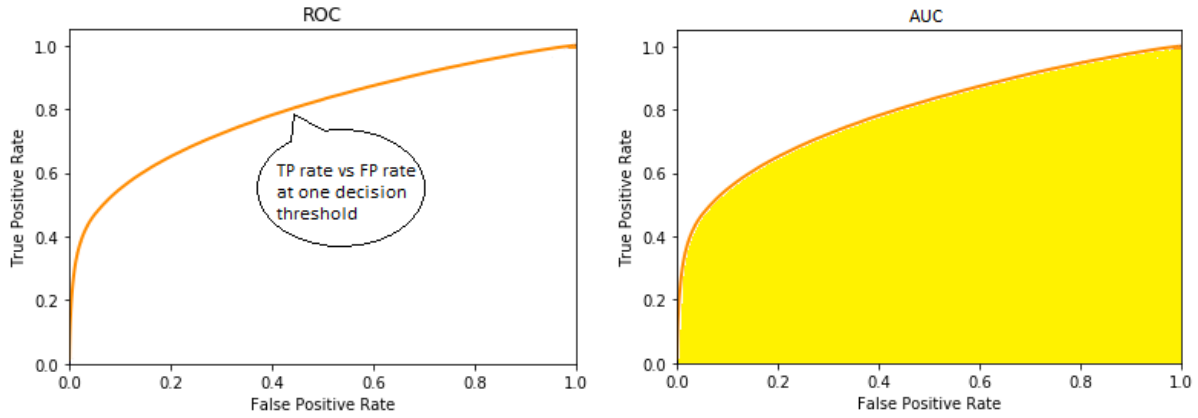



Figure 4.1: A ROC curve on the left and the area under the ROC curve in yellow on the right[1]

more advanced, was to use Term Frequency-Inverse Document Frequency (TF-IDF) to generate features and, again, use a logistic regression model to make predictions.

During these early experiments, two sets of texts were used, one set where the texts were anonymized and another where the texts were not. This had the purpose of seeing if the process of anonymization had a relevant impact on the utility/value of the texts, and also, to see how differently the models would behave.

When using traditional techniques the preprocessing task is very important, this is because they are more sensitive to certain particularities of the text that do not carry intrinsic value, like for example, the different frequencies of each word in the text, or spelling mistakes. Because of this, the texts used in the traditional models suffered three simple preprocessing tasks that were not done for the deep learning models:

1. Remove stopwords: All the stopwords were removed from the texts. The NLTK package for python was used to obtain the Portuguese stopwords list.
2. Remove high frequency words: The words that appeared in more than 85% of all the reports were removed. This was to remove words that although not being a stopword it behaved like one and most likely did not carried any value.
3. Remove rare words: The words that appeared in less than ten reports in total were removed. This was done because they were either a very rare word and therefore probably not significant in the text or, more likely, a spelling mistake. Given that there are more than five million reports, this step should not remove important and relevant words.

[1] Inspired in https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

### 4.3.1 Bag-of-Words

A traditional approach when working with texts is to apply the BoW method, described in section 2.1.1, where a set of features are generated using the words in the texts. When the BoW procedure was applied to the texts the number of features was limited to the thirty thousand words more frequent in the entire set of reports, doing this allowed to maintain the memory requirements lower while not being too restrictive to the point where the texts would lose significance and value. The features obtained from this were used in conjunction with a logistic regression model to obtain all the predictions presented here. From here onward, this arrangement will be referenced as a bag-of-words model.

The results are shown in table 4.1 and it can be seen that, curiously, the model trained on the anonymized texts managed to obtain better results both on the F1-score and the AUC metrics.

| | F1-score | AUC |
|---|---|---|
| Without Anonymization | 0.360 | 0.7244 |
| With Anonymization | 0.453 | 0.7665 |

Table 4.1: Results using Bag-of-Words with a Logistic Regression model

### 4.3.2 TF-IDF

TF-IDF is a technique for generating features when working with textual data and it's usually better and produces more value than the BoW method. By giving relevant weights to each word, it captures more information making the generated features more powerful. The way it works is explained with more detail in subsection 2.1.2. The features created using this approach consisted of n-grams of size one and two, this way they were more powerful and expressive in representing the real meaning of each word. The features obtained from this were used in conjunction with a logistic regression model to obtain all the predictions presented here. From here onward, this arrangement will be referenced as a TF-IDF model.

Similarly to the BoW approach, models were trained using texts both anonymized and not anonymized but for the TF-IDF two more models were built. These two models were trained while taking into consideration the fact that the dataset is somewhat imbalanced and that techniques could be used to make the dataset more balanced, so two ideas were tested:

- **Resampling**: To balance the dataset, the original percentage of true labels was increased from approximately 20% to around 40%. This was done by applying both oversampling and undersampling [Branco et al., 2016]. The oversampling was accomplished by simply duplicating all

true samples and the undersampling was applied in the form of randomly removing 25% of the negative samples.

- **Class weights**: When training the logistic regression model its possible to attribute different weights to each class, this means that during the training stage the model will give more importance to a class with a bigger weight. Although this aims to get the same result as the resampling idea since the imbalance of the dataset is kept and there is no duplication or removal of samples the model may focus/learn different things. The weights given to each class was inversely proportional to its frequency in the dataset.

The results are shown in table 4.2 and we can see that the model trained on the anonymized texts manage, again, to obtain better results both on the F1-score and the AUC than the one trained on texts not anonymized. The two models trained while dealing with the imbalance of the dataset got the best results on the F1-score for all the traditional models tested, although on the AUC metric they got marginally worst results than the best model.

| | F1-score | AUC |
|---|---|---|
| Without anonymization | 0.423 | 0.7450 |
| With anonymization | 0.522 | 0.7907 |
| With anonymization and class weights | 0.545 | 0.7891 |
| With anonymization and resampling | 0.563 | 0.7891 |

Table 4.2: Results using TF-IDF with a Logistic Regression model

## 4.4 Deep Learning Models

In this section, all deep learning architectures, explored in this work, are presented and their results showed while highlighting some of the most relevant comparisons between models of the same architecture but with different parameters.

The different architectures tested can be classified into three major types, those based in multilayer perceptrons, those based in convolutional layers and those based on recurrent neural networks. For each, different sizes for the networks were experimented, both in the number of neurons as in the number of layers. Various ideas and mechanisms associated with neural networks were also tested, things like dropout and batch normalization.

FastText, which is a model architecture released by Facebook, was another model type that was

experimented with. Both the model, pre-trained, available from its creators [Joulin et al., 2016], and a model trained from scratch using the dataset from the Telecom Company (Telco) were tested and evaluated.

A layer of embeddings was applied in all the networks tested, this is because when working with textual data the use of embeddings is a fantastic way to represent words. Different sizes for the embeddings were tested and experimented with. Most of the models trained the embeddings from scratch but some models were trained while using pre-trained embeddings, more details can be read in section 4.4.1.

The texts in the reports had a large variance in their length, as can be seen in section 3.1.2, with some texts having huge sizes. In order to limit the impact of the length of the texts in the performance of the models, the texts needed to suffer a step of preprocessing. The following three tasks were done:

1. Remove rare words: The very rare words were removed, this was accomplished by deleting the words that appeared in less than five reports in total. Most of those words were, in fact, spelling mistakes and those that were not, did not have a strong impact on the meaning of the text.
2. Vocabulary size: The texts were limited to a vocabulary of one hundred thousand words where only the most frequent ones were used. Given that these texts came from a very specific domain this was not a problem, nonetheless, different vocabulary sizes tested in order to see how much the number of different words impacts the model's performance.
3. Limit texts length: The length of the texts had a real impact on the model's training speed so in order to limit the impact on performance every text was truncated to a length of 120 words.

Similarly to what was done in the approach with TF-IDF which trained models that used resampling and class weights to manage the imbalance of the dataset, these mechanisms were also used here and for every major architecture type, different models were trained while applying either resampling or class weights.

The various parameterizations used in the deep learning experiments were chosen semi heuristically because there was not enough time to do a complete grid search on the parameters for each architecture, so only a few different parameterizations were tested, regardless of this, it was trained enough models with enough parameterizations to be able to draw some insights and some intuitions on what parameters worked best for every type of architecture used.

### 4.4.1 Word Embeddings

When working with Deep Neural Networks (DNN) in Natural Language Processing (NLP) a common technique is to use embeddings as a form of representing the words, this method is particularly effective

because they can be trained to capture the meaning of a word given a particular context. A full description of what are word embeddings can be read in section 2.2.

Various sizes for the embeddings layer were tried with the smallest being of size twenty and the biggest being of size fifty for those trained from scratch and three hundred for those that were pre-trained. Most models trained during the development of this work used embeddings that were built from scratch in parallel with the training of the model, although a few models, for each architecture type, were trained while using pre-trained embeddings. The pre-trained embeddings were obtained using three different methods, that are described next:

- **FastText base**: This method was to obtain the pre-trained embeddings from the FastText model that is distributed online and is available for the Portuguese language. The creators explain [Grave et al., 2018] that the embeddings were trained using Continuous-Bag-of-Words (CBoW) with position-weights, in dimension 300, on texts from Common Crawl and Wikipedia.
- **FastText custom**: Using also the FastText architecture, a FastText model was trained from scratch using the Telco reports and using embeddings of size 50, after training this model the embeddings produced were retrieved and used in the other models.
- **Custom**: The third method was to pre-train the embeddings using a custom approach. For each major architecture a new model, that contained an embeddings layer, was trained to predict a special label that described the type of report. After the training of this new model, the embeddings were retrieved and transferred to the real model that was going to be trained to predict the churn class.

All three methods of embeddings were applied to all three major architectures in order to verify how the different pre-trained embeddings would impact the results.

### 4.4.2 MLP

Multilayer perceptrons are ubiquitous in neural networks and therefore it was important to try an approach where the core layers were multilayer perceptrons. Various parameterizations were tried, including different sizes for each layer and different amounts of layers but all of them followed a basic structure. This structure was constituted by a layer of embeddings followed by a layer of max pooling where the dimensionality of the embeddings layer was reduced and after this, a series of dense layers were added, with and without dropout. The neurons on the dense layers used a Rectified Linear Unit (ReLU) as the activation function, with these layers it was also used regularization L2. A general scheme is presented in figure 4.2.

The table 4.3 presents some of the most interesting results for the models based on MLPs. The results obtained for all the models trained using MLPs as the core are presented in the appendix in table A.1.

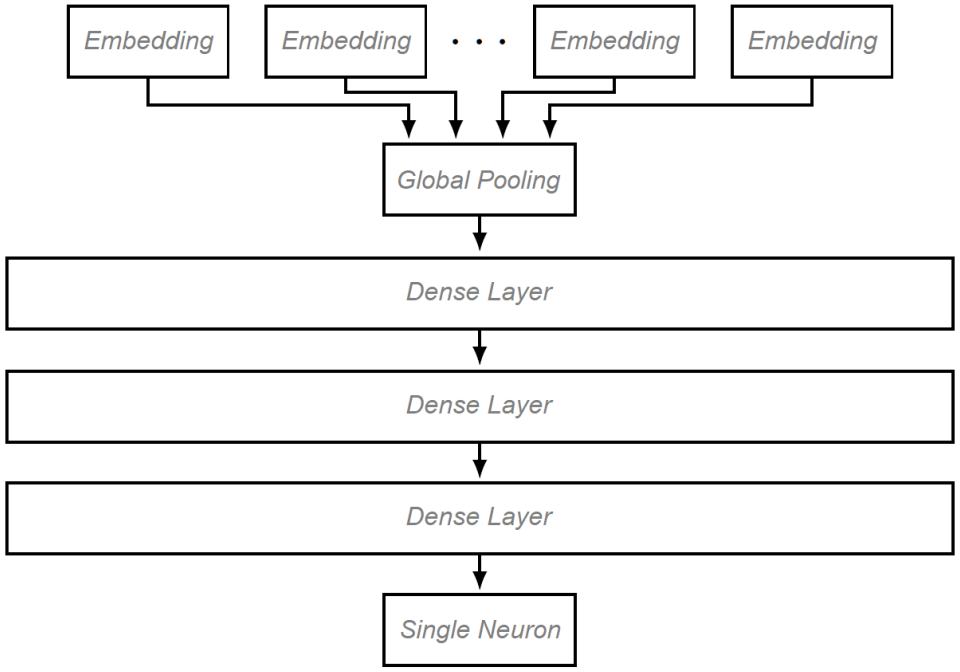



Figure 4.2: General scheme used for the MLP Architecture

| Name | Dropout | AUC |
|---|---|---|
| MLP-4 | Yes | 0.6498 |
| MLP-5 | No | 0.6170 |

| Name | Vocabulary size | AUC |
|---|---|---|
| MLP-3 | 50k | 0.6235 |
| MLP-7 | 75k | 0.6484 |
| MLP-8 | 100k | 0.6475 |

| Name | Embeddings [size] | AUC |
|---|---|---|
| MLP-10 | From scratch [50] | 0.6855 |
| MLP-11 | Custom [50] | 0.7082 |
| MLP-13 | FastText base [300] | 0.6083 |
| MLP-14 | FastText custom [50] | 0.6690 |

| Name | Imbalance | AUC |
|---|---|---|
| MLP-11 | - | 0.7082 |
| MLP-15 | weight | 0.7784 |
| MLP-16 | resampling | 0.7687 |

Table 4.3: Results using the MLP models

From the results of the MLP models trained and tested some values are worth mentioning. Experiments where the only difference between them was the dropout, the ones that used dropout performed better than the ones that did not, in some cases more than five percent in AUC. Also, the models trained with inputs which had a bigger vocabulary generally performed better. Another parameter that influences the success of the models was the size of the embeddings being trained with smaller sizes having lower scores of AUC. A technique used that had an important role in the performance of the models was pre-training the embeddings in which the three methods were used and are described in section 4.4.1.

The model that performed better was the last one and it achieved a AUC score of 0.7082, more than three percent better than the models where the embeddings were trained with the model from scratch.

Although some experiments performed better than others, almost all of them got an F1 score of zero, this is because these models were basically always predicting the label false. The only two model variants that managed to improve on this were the approaches where the problem of class imbalance was addressed. A model was trained while giving each class a weight where the weight was inversely proportional to its representation on the training set, this model achieved an F1 score of 0.5286 and a AUC 0.7784 making it one of the most successful. The other model that got an F1 score better than zero was the one trained using resampling in a similar fashion as it was when training the traditional model using TF-IDF 4.3.2. The results for this model was an F1 score of 0.5353 and an AUC of 0.7687.

### 4.4.3 CNN

Convolutional neural networks are primarily associated with images but recently they have been applied to textual data with great success, therefore, it made sense to make some experiments using architectures based on convolutional layers. Like with the other major architectures tested, various parameterizations were tried but all followed a similar structure. The models had a layer of embeddings that received the input and were followed by two convolutional layers, then a step of max-pooling was applied and after it, two more convolutional layers and in the end an average pooling was done. Unlike with the experiments based in MLP, it was not applied dropout between the convolutional layers, instead, steps of batch normalization were done in-between them. The activation function used in conjunction with the convolutional layers was a ReLU. To manage the weights in the convolutional layers is was used regularization L2. A general scheme of the convolutional architectures used is shown in figure 4.3.

The table 4.4 shows the more relevant models based on CNNs and their results. The results obtained for all the models trained using CNN layers as the model core are presented in the appendix in table A.2.

Various models were trained using CNNs as the core layers and generally, they obtained better results than the models based on MLPs. The differences between the various experiments when testing with different parameterizations maintained the same behavior as when using MLPs. Models that used dropout had better values of AUC, increasing the size of the embeddings and the number of units in the layers also appears to improve the results and give better values of AUC and F1, with the best among these with an AUC of 0.7626. Curiously the models that used pre-trained embeddings did not seem to improve on the best results obtained by the models where the embeddings were trained from scratch with the model, the best value of AUC gotten from pre-trained embeddings was 0.7565.

Similar to the models based on MLPs, the models that used CNNs got better results when the problem of class imbalance was addressed, both the models trained with class weights and with resampling outperformed all other models based on CNNs, the first one got an AUC of 0.7751 and an F1 of 0.549,

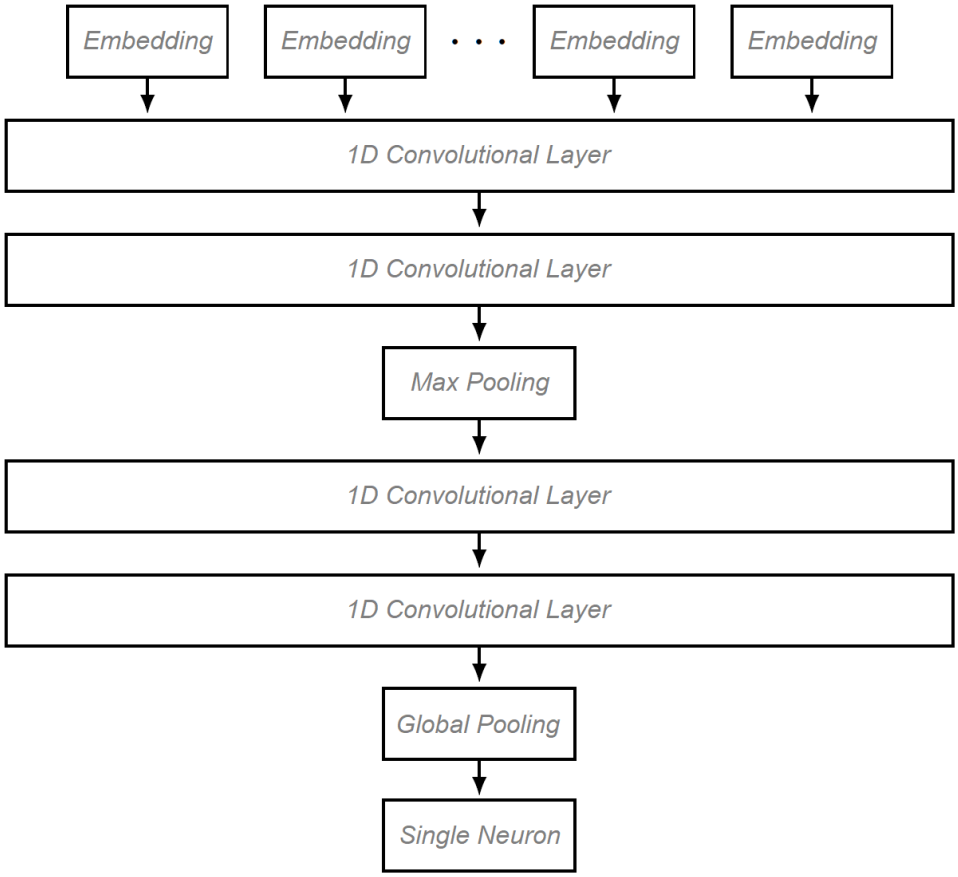



Figure 4.3: General scheme used for the CNN Architecture

the second got an AUC of 0.7764 and an F1 of 0.548.

### 4.4.4 RNN

Recurrent neural networks are at the forefront in the recent advancements when working with textual data. The tests made where the core layers of the models were RNNs followed a relatively simple structure. Like with the other architectures the models had a first layer of embeddings that received the input after that a bidirectional LSTM was applied followed by a layer of dropout and after it came a simple LSTM layer. Many parameterizations were tested including the amount of dropout applied, the size of the layers and the size of the embeddings. Although this type of model appears simple the LSTMs were very powerful and gave great results. A general scheme of the architecture based on RNNs is shown in figure 4.4.

The table 4.5 has some of the best results obtained while using models based on RNNs. The results obtained for all the models trained using RNN layers as the model core are presented in the appendix in table A.3.

Basically, all the models based on RNNs trained obtained better results than any other based on neural

| Name | Embeddings [size] | AUC |
|---|---|---|
| CNN-5 | From scratch [50] | 0.7626 |
| CNN-6 | Custom [50] | 0.7565 |
| CNN-7 | FastText base [300] | 0.6994 |
| CNN-8 | FastText custom [50] | 0.7380 |

| Name | Vocabulary size | AUC |
|---|---|---|
| CNN-2 | 50k | 0.7554 |
| CNN-3 | 75k | 0.7558 |
| CNN-5 | 100k | 0.7626 |

| Name | Imbalance | AUC |
|---|---|---|
| CNN-5 | - | 0.7626 |
| CNN-9 | weight | 0.7751 |
| CNN-10 | resampling | 0.7764 |

Table 4.4: Results using the CNNs models

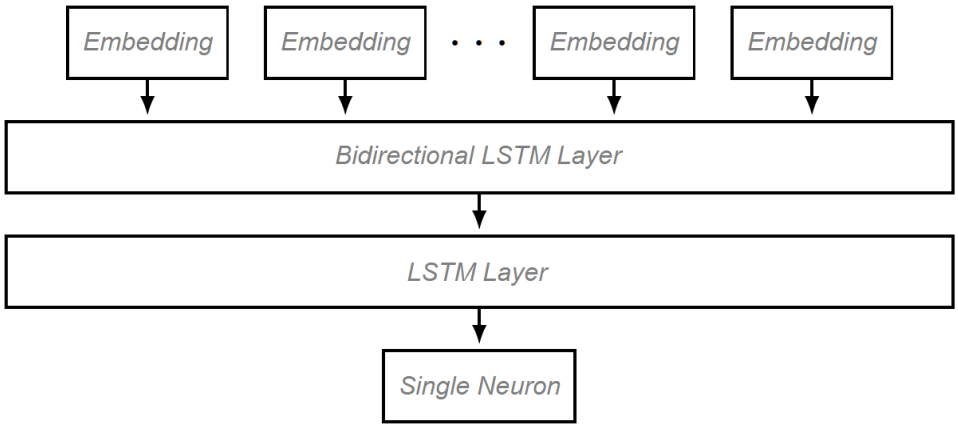



Figure 4.4: General scheme used for the RNN Architecture

networks. The various parameterizations, that included changing the size of the embeddings, changing the number of units in the LSTM layers, changing the size of the vocabulary and others, strangely did not play a significant role in the final performance of the models. With the exception of the model that was trained with the fewest LSTM units, smallest embeddings size and smallest vocabulary that got the worst result for this architecture with only 0.7587 of AUC pretty much all the others got approximately the same values for AUC, these values were in the range of 0.7834 to 0.7914. Similar to the models based on CNNs using pre-trained embeddings did not have and impact on the final results of the models.

Unlike the previous architectures, dealing with the problem of class imbalance did not improve the AUC scores already obtained by the other RNN models. The best AUC score obtained is 0.7908 when applying class weights during training. The F1-score did improve from 0.533, for the model 'RNN-10', to 0.564 for the model that used class weights (named 'RNN-12').

| Name | Embeddings [size] | AUC |
|---|---|---|
| RNN-7 | From scratch [50] | 0.7865 |
| RNN-8 | Custom [50] | 0.7885 |
| RNN-9 | FastText base [300] | 0.7893 |
| RNN-10 | FastText custom [50] | 0.7914 |

| Name | Vocabulary size | AUC |
|---|---|---|
| RNN-1 | 50k | 0.7587 |
| RNN-2 | 75k | 0.7870 |
| RNN-4 | 100k | 0.7867 |

| Name | Imbalance | AUC |
|---|---|---|
| RNN-10 | - | 0.7914 |
| RNN-12 | weight | 0.7908 |
| RNN-13 | resampling | 0.7817 |

Table 4.5: Results using the RNNs models

### 4.4.5 FastText

FastText is a model architecture distributed by the research team of Facebook and had some success in the text classification task. It also has a very fast training speed which makes it very interesting given that usually, deep learning models tend to be very slow while training. Two FastText models were trained, one with resampling and one without.

The table 4.6 has the results obtained.

| Name | Imbalance | F1-score | AUC |
|---|---|---|---|
| FastText-1 | - | 0.463 | 0.7703 |
| FastText-2 | resampling | 0.534 | 0.7716 |

Table 4.6: Results using the FastText model

There was no significant difference in the results of both models considering the AUC values, with one at 0.7703 and the other with 0.7716 but when looking at the F1 scores the model where resampling was applied performed much better with and F1 of 0.534 while the other only got 0.463 which means and improvement of fifteen percent.

## 4.5 Discussion

One of the objectives of this work was to evaluate how DNNs would perform with the data provided by the partner Telco. In particular, its performance when trying to predict *churn* using the Telco's call center reports. The way this problem was tackled was to compare the results from models based on traditional approaches against the results from models based on DNNs. After looking at the predictions and the results obtained from the models trained it appears that the deep learning models did not manage to outperform the traditional models despite being substantially more complex and significantly more time consuming to train. Both approaches ended up with results very similar.

Using the best models from both approaches, we can make a more detailed comparison of the results. The model with the best AUC value from the traditional side was the one that used TF-IDF with anonymization and the one from the deep learning side was the model 'RNN-10'. The results for both can be seen in table 4.7 and as it can be observed the AUC values are very similar.

| **Name** | **F1-score** | **AUC** |
|---|---|---|
| TF-IDF with anonymization | 0.522 | 0.7907 |
| RNN-10 | 0.533 | 0.7914 |

Table 4.7: Results from the models with best AUC

Another way of observing the closeness of both results is to visualize the ROC curves of the two models in figure 4.5 where it can be seen how strikingly similar they are.

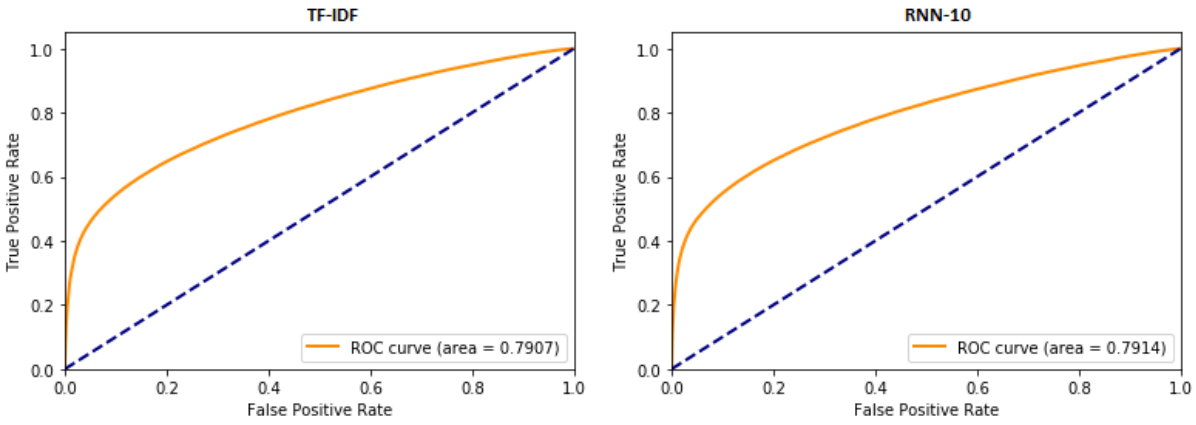



Figure 4.5: Comparison of ROC curves between models RNN-10 and TF-IDF

Yet another way to see that the two approaches manage to give predictions very similar to one another it to look at the confusion matrices of both in figure 4.6. The values in the confusion matrices are normalized for each class label.

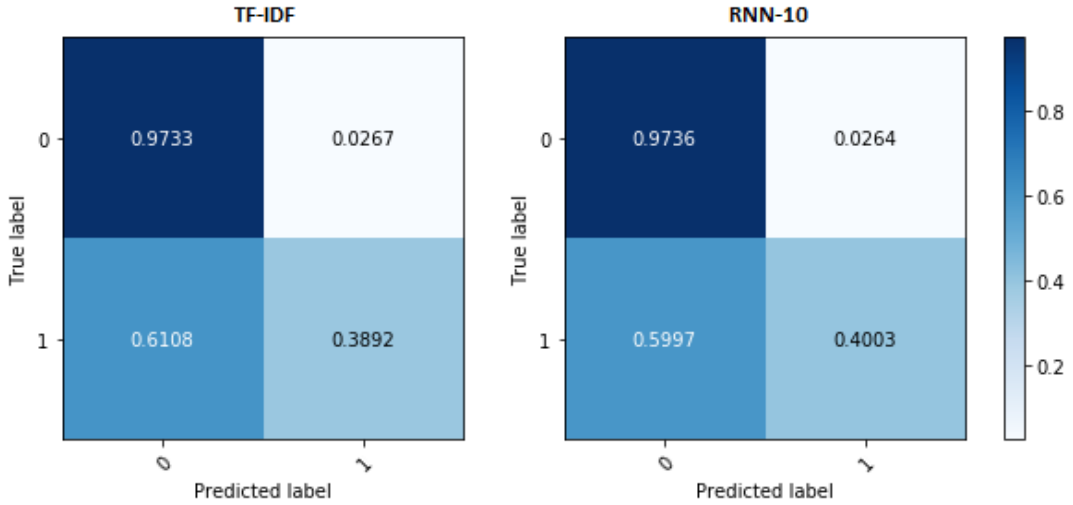



Figure 4.6: Comparison of confusion matrices between models RNN-10 and TF-IDF

It is also possible to verify how much the two models agree with each other. Making the intersection of the set of elements that are classified with the label True by the models its possible to see how many they give the same label. This can be extended to other sets of elements. Table 4.8 has the percentages on which both models agree on the same label for various sets.

| **Set** | **Percentage** |
|---|---|
| True | 0.9347 |
| False | 0.9860 |
| True Positives | 0.3621 |
| False Positives | 0.0195 |
| True Negatives | 0.9665 |
| False Negatives | 0.5726 |

Table 4.8: Sets percentages that the RNN-10 and TF-IDF give the same label

Although the results from both approaches are not equal, they are very similar and that fact raises the question of why that happened. Given that deep learning models are much more powerful than simple logistic regressions accompanied by simple generators of features it would be reasonable to expect

that they would outperform easily the traditional approaches tested in this work. Since DNNs routinely outperform other forms of machine learning one can make the assumption that part of the problem must rely on the dataset used. In order to understand if and why the dataset was to blame, three different analysis of the texts were done:

1. Frequency Analysis: The word frequencies were calculated to verify if there was any particular group of words that appeared more frequently for the set of True examples or the set of False examples. This would indicate that some words could be giving too much information and facilitating the problem for the traditional approaches. More details in section 4.5.1.
2. Weights Analysis: The weights that the traditional model attributed to each word were analyzed to see what the model considered the most important words. These words could then be used to measure how much the model was depended on them. More details in section 4.5.2.
3. Entropy Analysis: A more generic way to understand how much information the words carried about the target class is to calculate the entropy of each word for the entire dataset. This could reveal that a small group of words carried too much information, trivializing part of the problem. More details in section 4.5.3.

### 4.5.1 Frequency Analysis

In order to verify if the distributions of the different words between the sets of True examples and False examples were skewed, all the texts from the test set were divided into two groups, the one with the True examples and one with the False examples. After this, it was counted how many times each different unigram and bigram appeared in each of the two groups. Bigrams were used instead of only single words because the traditional model TF-IDF used both unigrams and bigrams for its features and it would be inaccurate to consider the impact of only individual words.

The figure 4.7 shows how much more each n-gram (considering only unigrams and bigrams) appears in a set in relation to the other set.

It can be seen that approximately 30% of all n-grams from the False set appears at least five times more in the False set than in the True set. In the True set, it is not as dramatic but still, around 8% of all n-gram from the True set appear five times or more in the True set than in the False set. Additionally, it can be seen that there is a significant percentage of words, in both sets, that appear more than twenty times in that particular set.

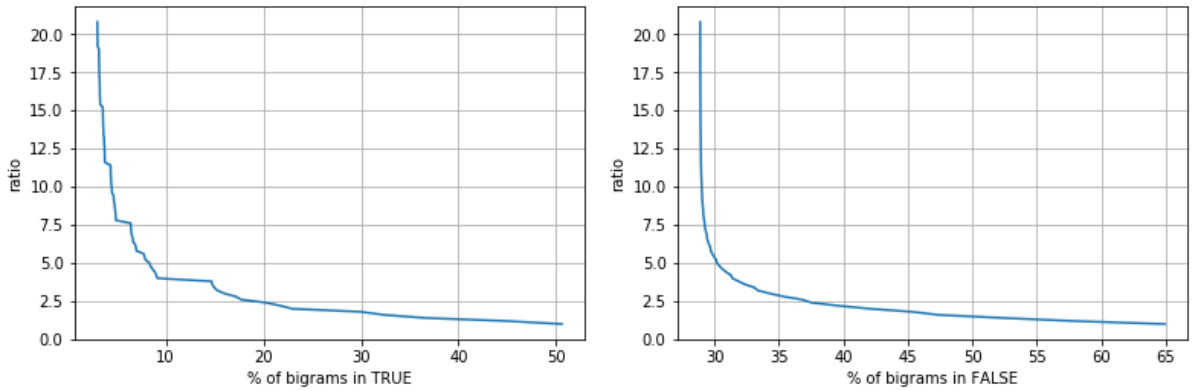



Figure 4.7: Distributions os unigrams and bigrams between the True and False sets

### 4.5.2 Weights Analysis

The traditional approach model trained using TF-IDF in conjunction with a logistic regression managed to achieve performance on par to the best deep learning models tested in this work. In order to try to understand how this happened, an effort was made to determine what n-grams (considering only unigrams and bigrams) the traditional model was focusing on. The weights attributed to the n-grams by the TF-IDF method and the weights from the logistic regression model were used to create a rank of the most important n-grams.

By multiplying the TF-IDF weights for each report by the logistic regression weights it is possible to leverage the relevance that each part is giving to each n-gram. After that, by taking the average of the weights of each n-gram for all the reports, the impact of the particular frequency of each n-gram is reduced. Equation 4.5 shows roughly the calculations done. Doing this gives a value for each n-gram that can be related to its importance for the traditional model.

$$N-grams\ Values = Average(TF\_IDF\ weights * Logistic\ Regression\ weights) \tag{4.5}$$

The table 4.9 contains the first eight n-grams with the highest values, and therefore more important.

It is curious to see that for some reason the model is giving particular attention to the tag '#DATE' that represents a date that was removed during the anonymization step. Intuitively the n-grams 'rescisao', 'nao retido' and 'desligamento' make perfect sense to be important because the model is trying to predict churn.

To verify how much these n-grams are actually important, they were removed from the reports texts and new predictions were made with each model. Table 4.10 presents the values of metric F1 and how much they changed.

| Rank | N-gram |
|---|---|
| 1 | #DATE |
| 2 | rescisao |
| 3 | sr |
| 4 | servico |
| 5 | nao retido |
| 6 | desligamento |
| 7 | tm |
| 8 | #NOME oferta |

Table 4.9: Top 8 n-grams according to TF-IDF and logistic regression weights

| Name | New F1-score | F1-score change |
|---|---|---|
| TF-IDF with anonymization | 0.408 | -21.86% |
| RNN-10 | 0.490 | -8.24% |

Table 4.10: Results change after removing TF-IDF important n-grams

Has it can be seen the deep learning model was much more robust to the removal of the n-grams showing that although the traditional model got comparable results to the deep learning model it was too reliant on a particular small set of words.

The figure 4.8 has the confusion matrices of the new results of each model.

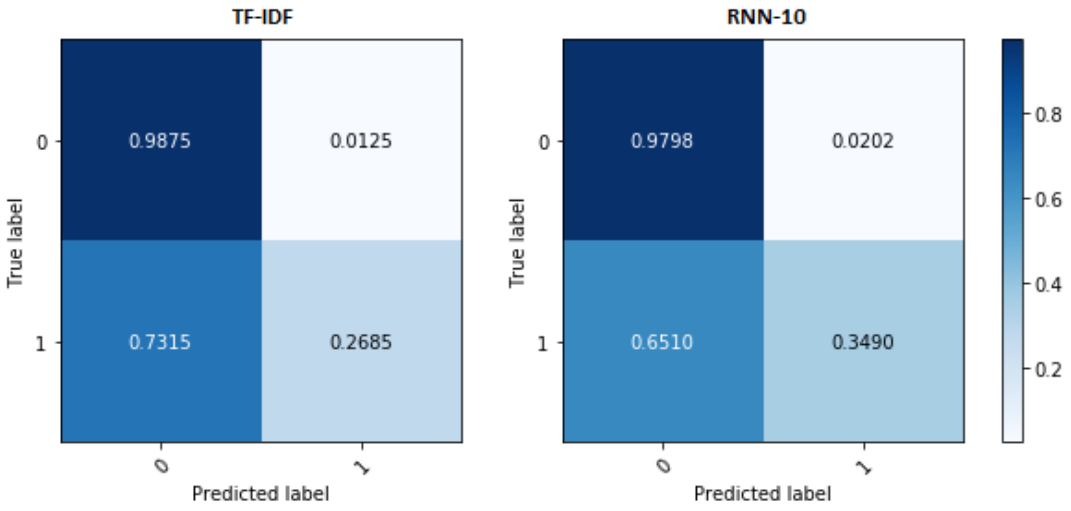



Figure 4.8: Confusion matrices after removing TF-IDF important n-grams

The set of True positives, arguably the most important predictions, got smaller for both models but the RNN-10 model suffered much less than the TF-ÎDF model, losing only 12.82% against the 31.01% lost by the traditional model, again showing its resilience.

### 4.5.3 Entropy Analysis

Entropy is a generic technique that can be used to understand the amount of information that exists within some piece of data [Goodfellow et al., 2016]. In this work, it was used to try to extract the words that were more descriptive and valuable when predicting churn. Following on the idea of the Shannon entropy [Shannon, 1948] the formula 4.6 was to calculate the entropy of each n-gram (considering only unigrams and bigrams).

$$Entropy = -\sum p(x) * \log_2 p(x) \tag{4.6}$$

Table 4.11 shows the top four n-grams with the lowest entropy, which means that they carried the most information about churn than any other n-gram in the texts.

| Rank | N-Gram |
|---|---|
| 1 | rescisao |
| 2 | nao retido |
| 3 | nao recuperado |
| 4 | pedido rescisao |

Table 4.11: Top 4 n-grams with the least entropy values

In similarity to what was done the the previous section (4.5.2), these four n-grams were removed from the texts and new predictions were made using these new texts, this way it was possible to see it the models were focusing too much on a few words or if they were catching the general value from the texts. The new results are presented in table 4.12.

| Name | New F1-score | F1-score change |
|---|---|---|
| TF-IDF with anonymization | 0.425 | -18.63% |
| RNN-10 | 0.515 | -3.54% |

Table 4.12: Results change after removing the n-grams with least entropy

Like it was seen before the traditional model lost significantly more performance than the model based on neural networks, and this was with a setup where only four n-grams were removed. These results raise serious concerns about the generality and robustness of the models tested in this work based on traditional approaches.

The confusion matrices for these new results can be seen in the figure 4.9 and pretty much like in the experience done in the section 4.5.2 the model RNN-10 managed to suffer much less consequences from the removal of these important n-grams. Considering only the set of True positives, the RNN-10 model lost 5.95% while the TF-IDF models lost 26.03% nearly five times more.

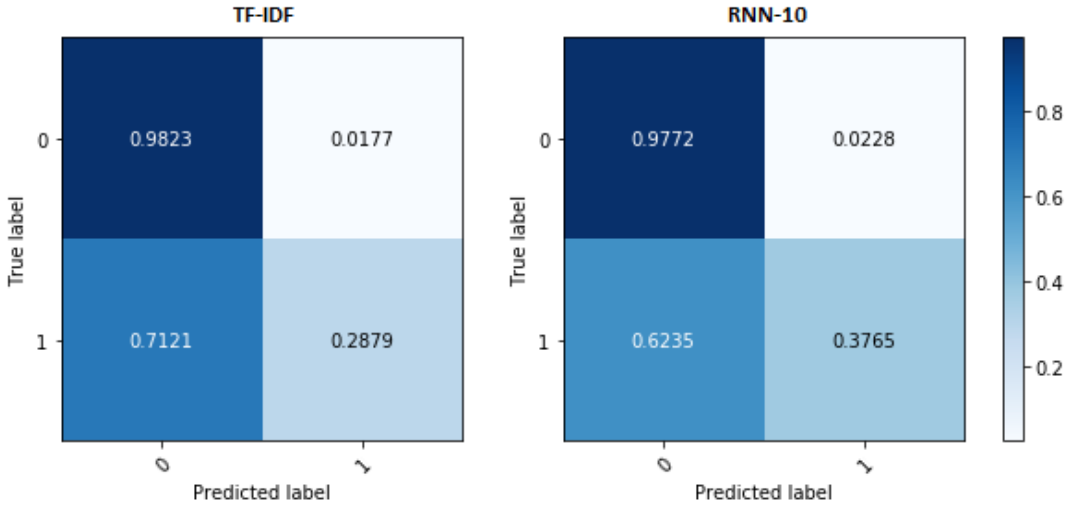



Figure 4.9: Confusion matrices after removing the n-grams with least entropy

### 4.5.4 Imbalance Considerations

The main topic of discussion is centered on the fact that deep learning approaches had similar results to more traditional approaches, but another discussion that can be made is to look at how the imbalance of the dataset affected the overall results and how each model architecture was impacted by the dataset imbalance.

Looking at the best results from each approach it can be seen that the impact of the imbalance on the results is dependent on the metric used and the model architecture. The various model architectures suffered differently with the imbalance. The TF-IDF model shows (table 4.2) an improvement when using the F1 score but no improvement when using the metric AUC. The same can be seen with the results from the FastText models (table 4.6) and from the RNN models (table A.3). The other models, those based on the MLP and CNN architectures got better results when training with resampling or class weights for the F1 score and had a slight improvement on the metric AUC as shown in tables A.1 and A.2 respectively.

Overall, it can be said that training with resampling and class weights helped the models to increase the confidence of its predictions. This is seen in more examples being classified with the *True* label which made the F1 scores increase. The relative ranking of its predictions, on the other hand, remained pretty much the same which is shown by the AUC score stability. This means that applying these techniques brought and artificial sense of improvement. The rise in the F1 score was not due to the model gaining more information from the dataset but because it simply skewed all predictions towards the *True* label.

Using resampling and applying class weights during training, aims to the same objective which is to combat the dataset imbalance by trying to make the less represented class more important and relevant to the model. Although these two methods function differently, by looking at the results from the experiments conducted in this work it is not possible to consider one better than the other. There were cases where using resampling originated in better results like when training the TF-IDF models (table 4.2) and cases that were the other way around like when training the RNN models (table A.3).

## 4.6 Summary

Throughout chapter 4, results concerning predictions of churn, obtained from several models, were presented. Although each model has its strengths and weaknesses, most of them performed very similarly, especially when considering the dramatic difference between traditional approaches and deep learning approaches. We find that models trained using TF-IDF features, which is a simple and fast model to train, obtained very good results when comparing with recurrent neural networks which are complex and very slow to train.

This leads to the immediate conclusion that using deep learning approaches with this dataset is not worth the effort. However, this conclusion requires additional explanation and study. One possible reason for this behavior, and probably the most convincing argument, comes from analyzing the dataset and understanding that, despite being very large, it fails to contain more than superficial and obvious information. When just a few words, like 'rescisao', have such a strong impact on the results, it shows that the information that predicts churn is easily obtainable and even simple approaches manage to capture it. The fact that the deep neural networks fail to retrieve more information than what is on the surface is probably a reflection of the origin of the reports. The operators in the call center have the task to summarize the phone call in a few lines of text, thus removing many of the subtext and intrinsic value from the phone call interaction. This process removes important information that would aid in pushing the DNN models above the traditional models.

# Chapter 5

# Conclusion

## Contents



This chapter presents the conclusions of this work and how the result compares to the objectives proposed in the beginning. It is also shown what future work could be done to further improve what was explored in this thesis.

## 5.1 Conclusions

In conclusion, we can infer that the main objective of this thesis was accomplished. An extensive exploration of the value of the reports from the Telecom Company (Telco) was done and multiple models were trained using various approaches and technologies.

It can be considered an advantage to use deep neural networks to mine value from this particular set of reports, although more simple methods appear to give similar results they are too brittle are prone to slight changes of the words on the texts. It should also be said that the human operators that wrote the reports most likely had already filtered and condensed the information in the reports, making it easier for the simple models but much harder for the more complex approaches to utilize their full capabilities.

One can easily conclude that although there is value to be extracted from these reports it is not a simple job. Trying to predict what a customer will do based on a single report from a phone call is an ingrate proposition and it would benefit greatly from other sources of information. Regardless of all the difficulties, the results obtained with the predictions are reasonably good, considering the difficulty of the problem.

The most important conclusion that can probably be made is to make the supposition that the reports are possibly being handicapped by the fact that they are written by humans. The human operator that makes the phone call, writes the report based on his opinion of what happened and that does not necessarily reflect the opinion or the interaction that actually happened with the customer. This layer of indirection between the reports and the customers makes it especially hard to retrieve value from them.

## 5.2 Future Work

There are many avenues that can be taken to improve and proceed with the work done in this thesis:

- A simple and obvious way would be to explore further with the different target classes that were created but were not selected for further work.
- Another rather simple proposition to continue on the path of exploration of this work would be to experiment with more powerful neural network architectures, like for example the new Bert model [Devlin et al., 2018] developed recently by Google research team.
- One task that needed to be done in this work and that definitely could be improved is the step of anonymization. Deep neural networks could be used to perform the anonymization step and very likely obtain a better dataset, possibly improving the final results.
- The results could probably be improved if extra information from the customers was used together with the reports from the call center. Adding this new information to the dataset and making new tests would be another way of progressing with this work.
- Of all the possibilities to continue with the work of finding value on the reports the best one is probably not to use these reports at all. Instead, use the soundtrack from the phone call to generate automatic text and afterward use it in the prediction models. This would have the advantage of truly containing all the information from the phone call and bypassing the judgment of the human operator.

# Bibliography


Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN: 1935-8237. doi:10.1561/2200000006.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997. ISSN: 0031-3203. doi:10.1016/S0031-3203(96)00142-2.

Paula Branco, Luís Torgo, and Rita P. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Comput. Surv.*, 49(2):31:1–31:50, August 2016. ISSN: 0360-0300. doi:10.1145/2907070.

François Chollet et al. Keras. https://keras.io, 2015.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc., 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 493–499, Cambridge, MA, USA, 1995. MIT Press.

Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Alex Graves and Jürgen Schmidhuber. 2005 special issue: Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Netw.*, 18(5-6):602–610, June 2005. ISSN: 0893-6080. doi:10.1016/j.neunet.2005.06.042.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954. doi:10.1080/00437956.1954.11659520.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN: 0899-7667. doi:10.1162/neco.1997.9.8.1735.

Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1080.

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN: 0131873210.

Cannannore Nidhi Kamath, Syed Saqib Bukhari, and Andreas Dengel. Comparative study between traditional machine learning and deep learning approaches for text classification. In *Proceedings of the ACM Symposium on Document Engineering 2018*, DocEng '18, pages 14:1–14:11, New York, NY, USA, 2018. ACM. ISBN: 978-1-4503-5769-2. doi:10.1145/3209280.3209526.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1181.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN: 0018-9219. doi:10.1109/5.726791.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1(4):309–317, October 1957. ISSN: 0018-8646. doi:10.1147/rd.14.0309.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943. ISSN: 1522-9602. doi:10.1007/BF02478259.

Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Interface: Talking to Smart Devices*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN: 3319329650, 9783319329659.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 1st edition, 2017. ISBN: 1491914254, 9781491914250.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and

E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Know.-Based Syst.*, 108(C):42–49, September 2016. ISSN: 0950-7051. doi:10.1016/j.knosys.2016.06.009.

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, 51(5):92:1–92:36, September 2018. ISSN: 0360-0300. doi:10.1145/3234150.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Guido Rossum. Python reference manual. Technical report, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands, The Netherlands, 1995.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN: 0-262-68053-X.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN: 0070544840.

Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Comput.*, 4(2):234–242, March 1992. ISSN: 0899-7667. doi:10.1162/neco.1992.4.2.234.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN: 1053-587X. doi:10.1109/78.650093.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, July 1948. ISSN: 0005-8580. doi:10.1002/j.1538-7305.1948.tb01338.x.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883, 2018.

# Appendix A

# Full Tables

The following tags are used in the tables:

- **Scratch #**: Embeddings trained from scratch with size #
- **Custom #**: Embeddings pre-trained while predicting the 'symptom' with size #
- **FastText base #**: Embeddings obtained from Portuguese model of FastText with size #
- **FastText custom #**: Embeddings obtained from FastText model after training it on Telco data with size #
- **Dr#**: Dropout layer with probability percentage #
- **De#**: Dense layer with # units
- **Emb**: Embeddings layer
- **GAP**: Global Average Pooling layer
- **GMP**: Global Max Pooling layer
- **C#1|#2**: 1 dimensional convolucional layer with #1 feature maps and kernel of size #2
- **MP#**: Max Pooling layer of size #
- **BN**: Batch Normalization layer
- **BiLSTM#1|R#2|#3**: Bidirectional LSTM layer with #1 units and Recurrent Dropout of #2 using operation #3 to join both outputs.
- **LSTM#1|R#2**: LSTM layer with #1 units and Recurrent Dropout of #2

| Name | Layers | Imbalance | Embeddings | Inputs | F1 | AUC |
|---|---|---|---|---|---|---|
| MLP-1 | Emb - Dr20 - GAP - De32 - Dr20 - De16 - Dr20 | - | Scratch 20 | 100k | 0.000 | 0.5702 |
| MLP-2 | Emb - Dr20 - GMP - De32 - Dr20 - De16 - Dr20 | - | Scratch 30 | 100k | 0.000 | 0.6395 |
| MLP-3 | Emb - Dr20 - GMP - De32 - Dr20 - De16 - Dr20 - De8 - Dr20 | - | Scratch 30 | 50k | 0.000 | 0.6235 |
| MLP-4 | Emb - Dr20 - GMP - De32 - Dr20 - De16 - Dr20 - De8 - Dr20 | - | Scratch 30 | 50k | 0.000 | 0.6498 |
| MLP-5 | Emb - GMP - De32 - De16 - De8 | - | Scratch 30 | 50k | 0.000 | 0.6170 |
| MLP-6 | Emb - GMP - De64 - De32 - De16 | - | Scratch 40 | 75k | 0.000 | 0.6288 |
| MLP-7 | Emb - Dr20 - GMP - De64 - Dr20 - De32 - Dr20 - De16 - Dr20 | - | Scratch 40 | 75k | 0.000 | 0.6484 |
| MLP-8 | Emb - Dr20 - GMP - De64 - Dr20 - De32 - Dr20 - De16 - Dr20 | - | Scratch 40 | 100k | 0.000 | 0.6475 |
| MLP-9 | Emb - Dr20 - GMP - De64 - Dr40 - De32 - Dr40 - De16 | - | Scratch 40 | 75k | 0.000 | 0.6394 |
| MLP-10 | Emb - Dr30 - GMP - De64 - Dr30 - De32 - Dr30 - De16 - Dr30 | - | Scratch 50 | 100k | 0.000 | 0.6855 |
| MLP-11 | Emb - Dr30 - GMP - De64 - Dr30 - De32 - Dr30 - De16 - Dr30 | - | Custom 50 | 100k | 0.000 | 0.7082 |
| MLP-12 | Emb - Dr30 - GAP - De64 - Dr30 - De32 - Dr30 - De16 - Dr30 | - | Scratch 50 | 100k | 0.000 | 0.5697 |
| MLP-13 | Emb - Dr30 - GMP - De128 - Dr30 - De64 - Dr30 - De32 - Dr30 | - | FastText base 300 | 100k | 0.000 | 0.6083 |
| MLP-14 | Emb - Dr30 - GMP - De128 - Dr30 - De64 - Dr30 - De32 - Dr30 | - | FastText custom 50 | 100k | 0.000 | 0.6690 |
| MLP-15 | Emb - Dr30 - GMP - De64 - Dr30 - De32 - Dr30 - De16 - Dr30 | weight | Custom 50 | 100k | 0.529 | 0.7784 |
| MLP-16 | Emb - Dr30 - GMP - De64 - Dr30 - De32 - Dr30 - De16 - Dr30 | resampling | Scratch 50 | 100k | 0.535 | 0.7687 |

Table A.1: Full MLP results

| Name | Layers | Imbalance | Embeddings | Inputs | F1 | AUC |
|---|---|---|---|---|---|---|
| CNN-1 | Emb - Dr20 - C16\|3 - BN - C16\|3 - BN - MP2 - C32\|3 - BN - C32\|3 - BN - GAP | - | Scratch 20 | 100k | 0.469 | 0.7651 |
| CNN-2 | Emb - Dr20 - C16\|3 - BN - C32\|3 - BN - MP2 - C64\|3 - BN - C128\|3 - BN - GAP | - | Scratch 30 | 50k | 0.440 | 0.7554 |
| CNN-3 | Emb - Dr20 - C16\|3 - BN - C32\|3 - BN - MP2 - C64\|3 - BN - C128\|3 - BN - GAP | - | Scratch 30 | 75k | 0.437 | 0.7558 |
| CNN-4 | Emb - Dr20 - C32\|3 - BN - C64\|3 - BN - MP2 - C128\|3 - BN - C256\|3 - BN - GAP | - | Scratch 50 | 100k | 0.393 | 0.7383 |
| CNN-5 | Emb - Dr20 - C32\|3 - BN - C64\|3 - BN - MP2 - C128\|3 - BN - C256\|3 - BN - GAP | - | Scratch 50 | 100k | 0.495 | 0.7626 |
| CNN-6 | Emb - Dr20 - C32\|3 - BN - C64\|3 - MP2 - C128\|3 - BN - C256\|3 - GAP | - | Custom 50 | 100k | 0.434 | 0.7565 |
| CNN-7 | Emb - Dr20 - C32\|3 - BN - C64\|3 - MP2 - C128\|3 - BN - C256\|3 - GAP | - | FastText base 300 | 100k | 0.409 | 0.6994 |
| CNN-8 | Emb - Dr20 - C32\|3 - BN - C64\|3 - MP2 - C128\|3 - BN - C256\|3 - GAP | - | FastText custom 50 | 100k | 0.410 | 0.7380 |
| CNN-9 | Emb - Dr20 - C32\|3 - BN - C64\|3 - BN - MP2 - C128\|3 - BN - C256\|3 - BN - GAP | weight | Scratch 50 | 100k | 0.549 | 0.7751 |
| CNN-10 | Emb - Dr20 - C32\|3 - BN - C64\|3 - BN - MP2 - C128\|3 - BN - C256\|3 - BN - GAP | resampling | Scratch 50 | 100k | 0.548 | 0.7764 |

Table A.2: Full CNN results

| Name | Layers | Imbalance | Embeddings | Inputs | F1 | AUC |
|---|---|---|---|---|---|---|
| RNN-1 | Emb - Dr20 - BiLSTM32\|R20\|concat - Dr40 - LSTM32\|R20 - Dr40 | - | Scratch 30 | 50k | 0.475 | 0.7587 |
| RNN-2 | Emb - Dr20 - BiLSTM32\|R10\|concat - Dr20 - LSTM16\|R10 - Dr20 | - | Scratch 40 | 75k | 0.528 | 0.7870 |
| RNN-3 | Emb - Dr30 - BiLSTM32\|concat - Dr40 - LSTM16 - Dr40 | - | Scratch 50 | 100k | 0.525 | 0.7834 |
| RNN-4 | Emb - Dr30 - BiLSTM32\|R10\|concat - Dr40 - LSTM16\|R10 - Dr40 | - | Scratch 50 | 100k | 0.530 | 0.7867 |
| RNN-5 | Emb - Dr30 - BiLSTM32\|R10\|sum - Dr40 - LSTM16\|R10 - Dr40 | - | Scratch 50 | 100k | 0.520 | 0.7854 |
| RNN-6 | Emb - Dr30 - BiLSTM32\|R10\|mul - Dr40 - LSTM16\|R10 - Dr40 | - | Scratch 50 | 100k | 0.530 | 0.7868 |
| RNN-7 | Emb - Dr30 - BiLSTM64\|R10\|concat - Dr40 - LSTM32\|R10 - Dr40 | - | Scratch 50 | 100k | 0.532 | 0.7865 |
| RNN-8 | Emb - Dr30 - BiLSTM64\|R10\|concat- Dr40 - LSTM32\|R10 - Dr40 | - | Custom 50 | 100k | 0.514 | 0.7885 |
| RNN-9 | Emb - Dr30 - BiLSTM64\|R10\|concat - Dr40 - LSTM32\|R10 - Dr40 | - | FastText base 300 | 100k | 0.529 | 0.7893 |
| RNN-10 | Emb - Dr30 - BiLSTM64\|R10\|concat - Dr40 - LSTM32\|R10 - Dr40 | - | FastText custom 50 | 100k | 0.533 | 0.7914 |
| RNN-11 | Emb - Dr20 - BiLSTM64\|concat - AM | - | Scratch 50 | 100k | 0.542 | 0.7853 |
| RNN-12 | Emb - Dr30 - BiLSTM64\|R10\|concat - Dr40 - LSTM32\|R10 - Dr40 | weight | FastText custom 50 | 100k | 0.564 | 0.7908 |
| RNN-13 | Emb - Dr30 - BiLSTM64\|R10\|concat - Dr40 - LSTM32\|R10 - Dr40 | resampling | Scratch 50 | 100k | 0.559 | 0.7817 |

Table A.3: Full RNN results