UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

USING MACHINE LEARNING TO INFER THE PSYCHOMETRIC CHARACTERISTICS OF MULTIPLE-CHOICE QUESTIONS

TESIS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIA DE DATOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

DIEGO EDUARDO REYES TRONCOSO

PROFESOR GUÍA:
Sebastián Ríos Pérez

MIEMBROS DE LA COMISIÓN:
Abelino Jiménez Gajardo
Pablo Dartnell Roy
Severin Lions Maitre

Este proyecto ha sido parcialmente financiado por ANID/ PIA/ Fondo Basal para Centros de Excelencia FB0003 y la subvención FONDEF ID21I10343

SANTIAGO DE CHILE
2023

USO DEL APRENDIZAJE AUTOMÁTICO PARA INFERIR CARACTERÍSTICAS PSICOMÉTRICAS DE PREGUNTAS DE SELECCIÓN MULTIPLE MÚLTIPLE

En este trabajo se exploran dos tareas relacionadas a preguntas de selección múltiple (PSM): inferencia de asignatura y estimación de dificultad. Para cada una se proponen nuevas metodologías con el fin de resolver estas tareas utilizando algoritmos aprendizaje de máquinas, considerando la exactitud de las predicciones junto con la factibilidad de implementar estos algoritmos en ambientes productivos.

La primera tarea consiste en inferir la asignatura de la PSM, lo que se encuentra dentro del ámbito de clasificación de texto. Este último ha sido ampliamente explorado bajo el campo de Aprendizaje Automático, para diferentes categorías de documentos, sin embargo, hasta donde conoce el autor, no existen artículos de investigación de este tipo para el idioma español, que es el utilizado en las preguntas de este trabajo. Esta tarea es de alta importancia en el campo educacional y está asociado a oportunidades de mejora en las tecnologías utilizadas, tales como etiquetado en bancos de preguntas o guiar el diseño de preguntas dentro de una asignatura. Para esta tarea, un modelo de Regresión Logística se utiliza para clasificar las preguntas, utilizando solo el texto de los ítems, sin incluir las alternativas. Los resultados muestran que el modelo propuesto obtiene un buen desempeño para la tarea con los datos proporcionados. Adicionalmente, se evalúa el modelo para detectar posibles imperfecciones en sus predicciones, así como los límites de la interpretabilidad de sus resultados.

La segunda tarea es la estimación de dificultad. Las metodologías actuales tienen un gran espacio de mejora debido a los costos requeridos. En este trabajo se estudian tres soluciones basadas en aprendizaje de máquinas para resolver esta tarea.

En primer lugar, se compara un algoritmo tradicional: Máquina de vectores soportantes (SVM), junto con un algoritmo de redes neuronales: Red Neuronal Recurrente (RNN), con el fin de contrastar ambas arquitecturas. Los resultados muestran que el modelo basado en RNN es mejor que SVM en términos de exactitud en las predicciones, por muy poca diferencia. Luego se introduce un algoritmo de redes neuronales más complejo: Modelo de memoria a largo corto-plazo (LSTM), el cuál logro superar considerablemente a los modelos anteriores. Finalmente se prueban arquitecturas basadas mecanismo de auto-atención, BERT, que se considera estado del arte para numerosas tareas relacionadas a lenguajes. Después de probar numerosos modelos de este tipo, no se logra obtener mejores resultados que LSTM. Al final se discuten todas las metodologías utilizadas, con sus beneficios y desventajas, así como posibles mejoras para futuros trabajos.

Para Misael y Violeta,
que a pesar de que ya no estén
los siento tan cerca como siempre.

Agradecimientos

En primer lugar quiero agradecer a los principales colaboradores y guias de este trabajo. A Pablo Dartnell por hacerme parte de este proyecto, por ayudarme a perfeccionar mis ideas y ser muy claro y oportuno en decirme qué debía mejorar. A Sebastián Rios por ser mi profesor guía y por el apoyo que me entregó, desde ayudarme a definir mis ideas, hasta facilitar el hardware necesario para poder realizar parte de los experimentos más importantes. A Severin Lions, por conseguir los datos necesarios para el proyecto, junto con los permisos de las instituciones. Además por su interés y disposición en la corrección de este documento. En especial, quiero agradecer a Abelino Jimenez, que además de su enorme dedicación y apoyo durante la realización de este trabajo, se convirtió en un mentor para mí. Agradezco enormemente los valiosos consejos que me ha dado y por guiarme en mi desarrollo profesional.

En segundo lugar, agradecer a mis amigos de la universidad: Harsh, Cristobal y Mitchel, con quienes nos conocemos desde la primera semana de clases y seguimos teniendo una gran amistad a lo largo de todo estos años, aún cuando ya no nos vemos tanto. A Gabriel, que nos conocimos en la carrera pero formamos una hermosa amistad a través de la música. A Consuelo y Felipe, que los conocí en el DII y me alegran el día siempre que nos vemos.

Agradezco a Josefa por hacerme inmensamente feliz desde que nos conocemos, por todo lo que hemos vivido juntos durante estos años y por todo el amor que me entregas cada día.

Por último, agradecer a quienes han estado desde el principio. A mis tios, por ser un ejemplo importante en distintas etapas de mi vida. A mi abuelo Misael y a mi bisabuela Violeta por haberme críado y compartir conmigo momentos inolvidables. A mi abuela Angélica, por cuidarme, asegurarse que nunca me faltara nada, y por ser mi segunda madre. A mis hermanos, Juan y Violeta por darme alegrías y apoyarme siempre. Y finalmente a mi Mamá, Ángela, por darlo todo por mí a pesar de las dificultades, por ser mi mayor inspiración y por siempre empujarme a dar lo mejor de mí.

Table of Contents

Table Index

Figures Index

Chapter 1

Introduction

1.1. Psychometric attributes in Multiple-choice questions

Multiple Choice Questions (MCQ) are one of the most used tools to asses students [1]. The purpose of this type of question is to measure students' knowledge and skills in a specific subject, at a specific level. To obtain reliable results, MCQs (which in the context of MCQs are also referred to as *items*) must be designed to be valid. There are some guidelines to develop and test MCQs following good practices [2], [3]. In general, they should assess the knowledge and ability for which they were designed. They also need to discriminate between students based on their knowledge and ability in the subject. The question design field should aim to ensure that no other skill or knowledge may benefit some students and disadvantage others. In addition, questions must have an appropriate difficulty for the skill level that is intended to measure, so that the level to be assessed is reflected in the question.

The structure of a Multiple-choice Question2 is composed of two sections:

- Stem: the question itself, which contains the information required to answer it correctly.
- Options: possible answers for the question. There are two types of options: the correct option, known as *key*, and *distractors*, which represent wrong answers.

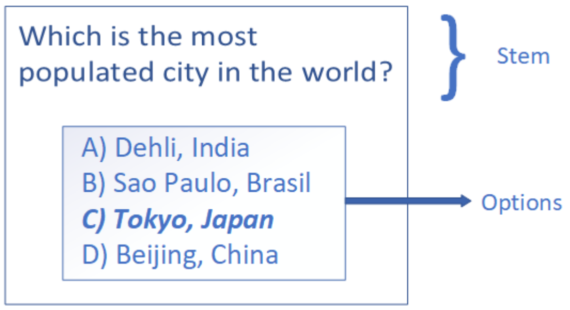

Figure 1.1: A Multiple-choice question item with 4 options. The question stem contains all the required information to choose the appropriate option. The key is the option *c*, *Tokyo, Japan*. Choices A, C, and D are known as distractors.

Multiple-choice questions are used in many standardized tests with a high impact on the future of students who answer them, known as high-stakes exams. To give an example, in Chile, the process of admission to universities begins with a test called PAES (previously known as PSU), which contains only multiple-choice questions. The score obtained in this test will allow the student to join the study programs of the country's universities, as long as their position is not occupied by another student with a better score. This is just one example of the application of multiple-choice questions in high-stakes tests. There are many more tests of this style in which this type of questions are used, such as driver's license tests, exams applied to nursing students, and even exams for admission to jobs or IT certifications. Therefore, the study of multiple choice questions is extremely important to design good knowledge measurement tools.

Although the questions included in these tests pass through long review processes, due to the complexity that a Multiple-choice question entails, many of them may still have imperfections when they are finally used in a real test [4]. To improve these processes, some of the research on multiple-choice question design aims to improve item quality by decreasing the number of *flaws* [5] and identifying the reasons why these flaws reduce the quality of the item. A similar scope is trying to understand the complexity of the question, to improve the item's design process. The complexity of a question can vary depending on the number of psychometric attributes included, which are the characteristics of a question that are related to its ability to measure the desired knowledge or skills and, hence, they allow to evaluate and measure their quality. Psychometric attributes can be divided into two categories. For one category, the attributes are fully defined at the time the question is designed, while in the other category, they can only be fully obtained empirically (once the question is tested or applied in a real test) and, hence, they can only be controlled by a priori estimates.

Psychometric attributes that are fully defined at the time the question is designed can be: the subject matter of the question, level (or grade), number of alternatives, and text length, just to mention a few. These attributes are fully controllable by the questions' designer and reflect the objectives and scopes considered in the success criteria defined for the designers. However, being able to predict these types of attributes can be very helpful for educational purposes, in particular, for the prediction of the subject of a question. For example, a study aid or practice test platform should be able to predict the subject matter of a question to help to recommend relevant study materials to users. It can also help to group questions by subject matter, which can help study or organize test materials. It could also be useful for educational researchers or assessment designers who are interested in understanding how well students understand the different subject matter or how well a particular assessment is measuring student learning.

Subject classification for questions is a particular type of text or topic classification. Text classification can be applied to any type of text document considered within a spectrum of similar documents such as news [6], e-mails [7] or even Twitter trending topics [8]. For multiple choice questions, there is not a large number of articles about subject classification. However, there is some research on similar types of topics, questions in internet forums [9], and types of categories of mathematics questions [10].

Additionally, to correctly infer empirical attributes in multiple-choice questions, such as difficulty and discrimination, it is often necessary to identify some design attributes, such as subject and level. Questions from different subjects don't necessarily use the same criteria to evaluate questions, hence, applications that estimate empirical attributes before being tested could use different approaches for each subject, to obtain more accurate estimates.

On the other hand, there are psychometric attributes that are harder to control, which can only be identified and measured accurately once the question has been applied in a test. Some of the empirical attributes are difficulty and discrimination. The question's difficulty is one of the most relevant empirical attributes of a question, having a good estimation of it is crucial when tests composed by MCQs are designed because they must apply to a set of students with a specific skill level, hence, the average difficulty of the test must be appropriate for the students. For an MCQ, the content itself, its readability, and the item's distractors can be a source of difficulty. However, the most popular method for calculating (or labeling) the item's difficulty is *Item Response Theory* (IRT), which does not consider the features mentioned above, instead, it leverages the skill level of the individuals who answered the question. IRT is considered as a *Logistic Model* due to the relationship of its components to a Logistic Function. This method is used to tag question difficulty, as it requires the test to be fully administered to have response data.

One way to estimate the item's difficulty before being used in a real test is by pre-testing questions and applying item response models [11] to the results. Another approach is to estimate the difficulty with subject matter experts. Both methods have advantages and disadvantages compared to each other. By pre-testing the question before it is applied in a real test, a smaller and more controlled test must be simulated with individuals who manage to mirror the target test takers. This process requires many resources to obtain accurate results. In addition, the test construction processes usually receive a large number of items, which makes the process even more difficult if they are planned to be fully evaluated. On the other hand, the use of experts in the field can considerably reduce the resources required, but the final result is usually less precise and inconsistent, due to the internal biases of each expert.

The main purpose of an assessment is to measure students' knowledge and ability on a specific topic, at a specific level. If the question is not well calibrated (i.e., the estimated difficulty is considerably inaccurate), the information retrieved from the students' responses can't be fully trusted, because the questions were not necessarily designed with the appropriate difficulty for the group of students that are intended to be evaluated. This situation becomes more relevant when it comes to tests having a similar difficulty for all students. Some tests that are applied massively (for example, those that are applied for university admissions in Chile) have different forms (that is, they have different questions), which may not have the same average difficulty, therefore, some students could have an unfair advantage over other students.

1.1.1. Objectives and Expected Results

Based on the information presented in the previous point, the need to seek improvements in the field of evaluation with multiple choice questions is notorious, due to the lack of research focused on the classification of subjects for MCQs and the failures and inefficiencies of the current forms. to estimate the difficulty. In this work, two types of tasks associated with the inference of psychometric attributes are distinguished: the subject of the question and the difficulty of the question. The main objective is to test the efficiency and effectiveness of the inference of these attributes with the use of machine learning algorithms. The general result that is expected is that the application of this type of technology delivers results that are close enough to the real values together with the typical benefits of its use, such as automation and resource savings.

To achieve this goal, we will take a tour of some of the main types of machine learning algorithms. Starting with traditional algorithms: those that do not have complex architectures based on neural networks, such as Support Vector Machines, Logistic Regression, and Random Forests. Then algorithms from the field of deep learning will be used, starting with classical architectures such as recurrent neural networks, Long Short-term Memory, and up to the most modern architectures such as Transformers. The objective is to study the efficiency of these algorithms and compare it together with their requirements for the hypothetical case in which this technology is used to evaluate questions. It is known that deep learning models tend to be very computationally expensive both in their training and in making the predictions for which they were trained. But the high cost of these models is often accompanied by considerably higher performance than traditional models. Part of this work is to verify that, at least for this problem, this type of behavior is fulfilled, and in the same way, to justify whether it is really necessary to introduce this type of technology.

1.2. Document's structure and contributions

In the previous section, two highly important tasks in the educational field have been introduced. On the one hand, the prediction of the subject in a multiple-choice question, with great possibilities of improving applications, services, and processes in the field of technology in education, and on the other hand, the estimation of the difficulty of a multiple-choice question, which It can allow test designers to anticipate the outcome of an empirical attribute and to be able to choose the best questions to go on a test with more information, thus supporting the design process.

In Chapter 2, the fundamental concepts of natural language processing that are shared in the following chapters are introduced along with the numerical representations and features used in this work.

In chapter 3, a model that can predict the subject matter of a Multiple Choice Question is proposed, only by using the items' stem as input with *TF-IDF* representation. It is composed of a Logistic Regression model, which proved to be a very good approach for text classification tasks. The Model is tested with *LIME* methodology to discuss if based on his results, their predictions are considering the appropriate features and if they can be suitable for different sources of data.

In chapter 4, the task of Difficulty Estimation with Machine Learning is explored. For this purpose, several machine learning models are proposed for items' difficulty estimation. First, traditional representations and algorithms (TF-IDF and Support Vector Regression), followed by models with representations and methods such as pre-trained word embeddings (GloVe) and Deep Neural Networks Architectures (Recurrent Neural Network and Long Short-term

Memory). Finally, two formulations of Transformers[1] models are also tested.

[1] Machine learning models that are considered state of the art for several NLP tasks

Chapter 2

From text to numbers

2.1. What is Natural Language Processing?

There is not a single definition for Natural Language Processing (NLP). It can be considered as the design of methods and algorithms that have natural language text as input [12]. NLP is focused on the design and analysis of computational algorithms and representations for processing natural human language [13] and is related to *Computational Linguistics* [14], which is considered as the computational processes involved in studying how to understand, produce and learn language. It refers to a set of methodologies, tools, and frameworks to extract information from the natural text and use them with machines to solve text-related tasks.

In Natural Language Processing, a collection of samples of text data is known as *Corpus*, and each sample are commonly referred to as *Document*, which is considered as a piece of text that can be processed and analyzed by a computer. They can be a single sentence, a paragraph, an article, or any other unit of text. For example, let D be a corpus, which is a multiset[2] and let $d_1,\ d_2\ \in D$ such that:

$$\begin{aligned} D &= \{d_1, d_2\} && (2.1) \\ &= \{\text{The fervent fox leapt over the fence}, \text{The fox jumps over the lazy dog}\} && (2.2) \end{aligned}$$

Documents are often represented as vectors of *tokens*, which are a single unit of meaning in a text or corpus. A token could be a word, a punctuation mark, or any other building block of a text. To transform text to tokens, a *tokenizer* must be used, which is an algorithm that breaks down text into a piece of tokens. When using a tokenizer, it must be defined for a specific type of *n-gram*. The type of *n-gram* refers to the number of items to be considered as a token. For example, for the following sequence:

$$\text{"The fervent fox leapt over the fence"} \qquad (2.3)$$

A *uni-gram* ($n = 1$), which is a type of n-gram that considers each token as a single element

[2] A multiset is a collection of elements in which each element may occur more than once

in the sentence, will return the following array:

$$\{\text{The}, \text{fervent}, \text{fox}, \text{leapt}, \text{over}, \text{the}, \text{fence}\} \tag{2.4}$$

While a *bi-gram* ($n = 2$), will return:

$$\{\text{The fervent}, \text{fervent fox}, \text{fox leapt}, \text{leapt over}, \text{over the}, \text{the fence}\} \tag{2.5}$$

In addition, for each corpus, a vocabulary set is attached to it, which contains all the distinct tokens that can appear in any document in the corpus. For example, for the corpus D his vocabulary set will be:

$$V = \{\text{The}, \text{fervent}, \text{fox}, \text{leapt}, \text{over}, \text{fence}, \text{jumps}, \text{lazy}, \text{dog}\} \tag{2.6}$$

These are some of the foundational concepts in NLP that apply to any type of task. The procedures required to extract information from natural text, which can be interpreted by algorithms on a machine, are introduced in the next section.

2.2. Numerical representations for natural text

Machines are unable to understand text in its natural form; it must first be transformed into a numerical representation to be processed. There are different ways to obtain meaningful numerical representations from the text. The goal is to capture as much information as possible, both syntactically and semantically[3].

Among the existing procedures to represent text, one of the most popular ones is one-hot encoding [15]. This methodology creates vectors of dimension equal to the cardinality of the vocabulary, which is equal to 1 in the cell associated with the word to be represented and equal to 0 for the rest of the cells. Then, the documents are represented as an array of one-hot encoded vectors, where each vector represents a word. Alternatively, there is the Bag-of-Words (BoW) representation [16], which provides only one vector per document, and where the value in each cell corresponds to the number of times the word associated with the cell appears in the document. The main difference between bag-of-words and one-hot encoding is that BoW counts the frequency of each word, while One-hot encoding represents each word as a unique categorical value.

There are also more complex ways of representing text numerically, which attempt to improve information extraction by capturing the relative importance of words within a Corpus or by interpreting the similarity between words. One-hot encoding and Bag-of-Words are

[3] Syntax is a branch of grammar that studies the principles of ordering and combining words and the sets of words they form in a sentence. Semantics is the study of the meaning attributable to syntactically well-formed expressions.

classical representations and the foundations of the ones used in this work: TF-IDF and Word embeddings, which are introduced in this section.

2.2.1. Term Frequency, Inverse Document Frequency

The Bag-of-Words (BoW) representation does not allow a precise interpretation of the relative importance of words in the corpus. This only considers the frequency of a vocabulary in a document. When trying to classify text, it is necessary to take into account that the frequency by itself can be a biased indicator of the type of text that you want to classify. For example, the word *government* will probably have a high frequency in texts of the type: *History*, *Politics*, and *Economics*. But, to distinguish between the mentioned categories, this word would not help, since it would be quite common throughout the corpus. In this context the idea of *Term Frequency* and *Inverse Document Frequency* is introduced [17].

First, let D be the set of Documents (or Corpus), where each document $d \in D$ is a multiset of words that contains the mapped version of text to tokens and let V be the vocabulary set, which contains all the distinct tokens that appear in the corpus.

Term Frequency is the number of times a token $w \in V$ appears in a document $d \in D$ divided by the total number of tokens in the document $d \in D$. The result represents the relative importance of a word in the document. Then for a token $w \in V$ its term frequency over the document $d \in D$ is calculated by:

$$TF(w,d) = \frac{f_{w,d}}{\sum_{w' \in d \cap V} f_{w',d}} = \frac{f_{w,d}}{|d|} \tag{2.7}$$

Notice that the divider in (2.7) is the sum of all of the frequencies of the distinct words in the document d over the document d, i.e. the cardinality of the document d. The numerator is the frequency of the word $w \in V$ over the multiset d.

Also, the following property is satisfied:

$$\sum_{w} TF(w,d) = 1 \quad \forall d \in D \tag{2.8}$$

Which means that it is normalized by a norm *l1*. It can also be normalized by *l2* norm, obtaining the following formula:

$$TF(w,d) = \frac{f_{w,d}}{\sqrt{\sum_{w' \in d \cap V} f^2_{w',d}}} \quad \forall w \in V,\ \forall d \in D \tag{2.9}$$

Where:

$$\sum_{w} (TF(w,d))^2 = 1 \quad \forall d \in D \tag{2.10}$$

It is important to notice that Term Frequency by itself is not that different from Bag-of-Words. It is just a version of Bag-of-Words normalized by the number of words in the document.

Inverse Document Frequency measures how important a word is within a corpus. It is based on the idea that words that are more common in a corpus are less important to discriminate between documents, hence, they should have a lower weight in the prediction. It is calculated by:

$$\text{IDF}(w) = \frac{|D|}{|d \in D : w \in d|} \quad \forall w \in V \tag{2.11}$$

Where the numerator refers to the cardinality of the corpus (a constant) and the divider is the cardinality of all the documents in which the term w appears. For instance, a very common word should have a bigger value in the divider and, hence, a lower IDF.

The metric used to compose the numerical representation is the product of the Term Frequency and the Inverse Document Frequency:

$$\text{TF-IDF}(w, d, D) = TF(w, d) \cdot IDF(w, D) \tag{2.12}$$

Since the TF values go from 0 to 1 and the IDF values can be much larger, instead of using just the IDF formula from (2.11), some changes are added: (i) an ln function is applied over the formula to not absorb the TF values, (ii) add 1 outside the logarithm so that tokens with IDF equal to 0 are not immediately 0 in their TF-IDF value (iii) add 1 to the divisor and denominator to avoid dividing by 0.

$$IDF(w, D) = \ln\left(\frac{1 + |D|}{|d \in D : w \in d| + 1}\right) + 1 \tag{2.13}$$

Hence, the final formula is:

$$\text{TF-IDF}(w, d, D) = \frac{f_{w,d}}{\sqrt{\sum_{w' \in d \cap V} f_{w',d}^2}} \cdot \left(\ln\left(\frac{1 + |D|}{|d \in D : w \in d| + 1}\right) + 1\right) \quad \forall d \in D,\ \forall w \in d \tag{2.14}$$

TF-IDF representation is considered a great improvement from Bag-of-Words. Their main value is by capturing the relative importance of words within a document and the corpus. However, there are also some limitations to using TF-IDF. It is not able to capture the context or meaning of words in a document, and it can be very expensive to compute for *big corpora* because larger vocabularies are used and many more documents are processed (which increases the number of columns of the TF-IDF matrix). Methods such as distributed computing or parallel processing can be used to calculate TF-IDF scores across multiple machines, allowing for more efficient processing. However, compared to TF-IDF, word embeddings can be a more efficient approach [18]. They don't necessarily require calculating the TF-IDF score for each word in a document or corpus. Instead, word embeddings are typically pre-trained on a large dataset and can be used directly as input to machine learning models. This can save time and computational resources, especially when working with large datasets.

2.2.2. Word Embeddings: GloVe

Word embeddings are a type of text representation used to transform words into vectors in a high-dimensional space, They are commonly used in NLP to capture the meaning of words and improve the performance of tasks by training a model to predict the surrounding words. During the process, the model learns to encode the meaning of words in vectors, which are called word embeddings. The main improvement, compared to the TF-IDF model, is that word embedding learns the semantic relationships between words. For example, with BoW and TF-IDF representation, "king" and "queen" would be treated as completely separate words, with no connection to each other. With a word embedding representation, they would be represented as vectors that are close to each other by being semantically similar.

A widely used representation is *GloVe*. It refers to *Global Vectors for Word Representation* [19], which is based on the framework proposed by Mikolov et al. [20], [21] known as *Word2Vec*. The key difference between GloVe and other word embedding techniques is the way it generates word vectors. While word2vec and other techniques generate vectors by predicting the surrounding words given a word, GloVe generates vectors by factorizing a global word-word co-occurrence matrix. This matrix is created by counting the number of times each word appears in the context of every other word in the dataset. Hence, for GloVe, the matrix considers the entire corpus instead of local context only (neighboring words), as Word2vec does.

Let's suppose that i and j are two words in a vocabulary. The co-occurrence matrix is defined by X_{ij} which is the times that i and j appear together in the same *context*, defined by a *window size*, which is the length of the sequence of tokens in which two words share the same context.

$$X_{m\times n} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix} \tag{2.15}$$

Then, the word vectors are obtained by factorizing them into two matrices W and C, where W_i is the word vector for word i and C_j is the context vector for word j. These matrices are obtained by minimizing the objective function:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f(X_{i,j})(W_i \cdot C_j - \log X_{i,j})^2 \tag{2.16}$$

Where $f(X_{i,j})$ is a weighting function that down-weights the importance of common words and up-weights the importance of rare words

$$f(X_{i,j}) = \frac{(X_{i,j}+1)^{\alpha}}{X_{i,j}+1} \tag{2.17}$$

Here, α is a parameter that controls the rate of decay. A larger value of α results in a stronger down-weighting of common words, while a smaller value results in a weaker down-weighting.

GloVe is a pre-trained model. It means that it has already been trained, on a large dataset, resulting in a model which can provide word embeddings without requiring training it again. As it has been trained with large amounts of text data, it has reached a high level of precision in the vectors it generates. For this work, the pre-trained model used is the one with 25 dimensions, trained with 2 billion of *tweets*, which contains approximately 27 Billions of tokens [4].

Overall word embeddings are a useful tool for various NLP tasks. They improve the text representation by capturing the meaning of words and the relationships with their context and also increase the efficiency of the processes involved by using pre-trained models.

2.3. Features

For some NLP tasks in the MCQs field, such as difficulty estimation, additional features are often used with the numerical representations to improve the accuracy of predictive models. These features aim to capture attributes that go beyond the natural text and their value depends on the task that is intended to be solved. In this section, two of these attributes are introduced: Readability y and Cosine Similarity, which were used to estimate the difficulty of MCQs.

2.3.1. Readability

Readability is an indicator of the level required to comprehend a piece of text [22]. For a Multiple-choice question, it may help to estimate the prior vocabulary domain or reading comprehension required to correctly answer the question. For example, if a question is written in plain language, it will be easier to identify and understand the main elements of the question needed to deliver a correct answer. On the other hand, if the language used is technical and complex, the question will require greater knowledge and skill to be correctly understood and, therefore, to provide a correct answer.

In this work, additional features are used in conjunction with text, to capture its readability. These features are highly influenced by the ones used in Garcia et al (2012) [23] and Benedetto et al. (2020a) [24] to estimate readability of Multiple-choice questions, primarily built by counting the items' tokens and sentences. In this case, only 9 of them were used (the most representative of the complexity of text as a hypothesis of the author), due to the number of features used in total. They are described in the following table:

[4] https://nlp.stanford.edu/projects/glove/

Table 2.1: Linguistic Features used in this project

Feature	**Definition**
Word Count Stem	Number of words in the stem
Word Count Key	Number of words in the key
Avg Word Count Distractor	Mean of the number of words of the distractors
Avg Word Length Stem	Mean of the lengths of every word in the stem
Avg Word Length Stem/Avg Word Lenght Key	Mean of the lengths of every word in the stem divided by the mean of the lengths of the words in the key
Avg Word Length Stem/Avg Word Lenght Distractors	Mean of the lengths of every word in the stem divided by the mean of the lengths of the words of the distractors
Sentences Count Stem	Number of sentences in the stem
Sentences Count Key	Number of sentences in the key
Avg Sentences Count Distractor	Average Number of sentences in the distractors

To estimate the readability of an item, there also exists *readability indexes*, which are calculated primarily by linguistic attributes and coefficients obtained by empirical research. In this section, three metrics to obtain readability are also introduced.

The first readability index is the **Flesch Reading Ease** (FRE) [25]. It is based on the idea that text that is easier to read should have a higher score. The score is calculated using two factors: the average number of syllables per word and the average number of words per sentence. The formula is given by:

$$\text{FRE} = 206.835 + 1.015 \cdot \frac{Number\ of\ words}{Number\ of\ sentences} - 84.6 \cdot \frac{Number\ of\ syllables}{Number\ of\ words} \tag{2.18}$$

The result is a value that when it is very high, is very easy to read, and as it decreases, the text is simpler. A score of 90-100 indicates text that is easily understandable by an average 11-year-old student, a score of 60-70 indicates text that is easily understandable by an average 13- to 15-year-old student, and a score below 30 indicates text that is best understood by university graduates.

Another readability index used in this project is the **Automated Readability Index** (ARI) [26], which approximates the US grade level needed to comprehend text. It is based on the idea that text that is easier to read should have a lower score. The score is calculated using two factors: the average number of characters per word and the average number of words per sentence.

$$\text{ARI} = 4.71 \cdot \frac{Number\ of\ characters}{Number\ of\ words} + 0.5 \cdot \frac{Numer\ of\ words}{Number\ of\ sentences} - 21.43 \tag{2.19}$$

The resultant score is a number between 0 and 100, with higher scores indicating text that is more difficult to read. According to the Automated Readability Index scale, a score of 0-30 indicates text that is easily understandable by an average 4th-grade student, a score of 31-50 indicates text that is easily understandable by an average 8th or 9th-grade student, and a score above 90 indicates text that is best understood by university graduates.

Both are still considered decent approaches for estimating readability. Even though the coefficients have remained constant since the method was proposed, it is still widely used for different tasks that require estimating readability. However, for text written in English, readability is not necessarily the same as its translation into Spanish. In the English language, short words could have a longer translation into Spanish and vice-versa. This immediately affects the perception of the text and the values of the features proposed. This is a complication because the data used in this work is written in Spanish. Therefore, there is no certainty that using the translation of the text will deliver an appropriate result for the tasks that are intended to be performed in this work. Even though the two mentioned indexes could be used as an approximation of the readability of Spanish text, they don't capture the required difficulty. In this context, another readability formula arises.

The **Spanish Readability Formula**, proposed by Seth Spaulding (1956) [27] uses the average sentence length as a feature for its calculation and introduces another concept: *density* (also called Index of Vocabulary Difficulty). This is a count-based measure from a specific vocabulary (which contains nearly 1500 words) which is computed as the number of words which doesn't appear in the vocabulary. The difficulty is calculated by:

$$\text{Spanish Difficulty} = 1.609 \cdot \text{Average Sentence Length} + 331.8 \cdot Density + 22 \tag{2.20}$$

The resultant score is a number between 0 and 10, with lower scores indicating text that is easier to read. According to the Spanish Readability Formula scale, a score of 0-2 indicates text that is easily understandable by an average 4th-grade student, a score of 3-5 indicates text that is easily understandable by an average 8th or 9th-grade student, and a score above 8 indicates text that is best understood by university graduates.

Although it may be counterintuitive to use indexes designed for a language other than the one in this work, FRE and ARI continue to be important benchmarks for reflecting reading difficulty. The simplicity and the good results they deliver have allowed them to remain current despite having been designed several years ago. In this sense, the fact that the index in Spanish is almost 70 years old does not prevent it from being a good point of reference on the difficulty of the text, but this time with the advantage of its original language.

2.3.2. Options' similarity

When the options of an item are very similar to each other it can be harder to decide which one is correct. Moreover, if the *key* is highly similar to a distractor, it may confuse the student about the correctness of his choice, even if the student is highly skilled in the subject matter of the question. Therefore, it is reasonable to use the similarity between choices as a factor for item difficulty. For this purpose, the Cosine Similarity metric is introduced.

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Suppose two vectors A and B that represents text, where $A, B \in \mathbf{R}^n$ and $A \neq B$. Their cosine similarity is calculated as follows:

$$\text{Cosine Similarity}(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.21}$$

The result will be a value between -1 and 1, with -1 indicating that the vectors are opposite, 0 indicating that they are orthogonal, and 1 indicating that they are identical.

A and B can be, for instance, a one-hot encoded version of two words, or even a TF-IDF representation of a sentence concatenated with the features described above. In this case, in all models the average similarity between the MCQ key and the distractors was calculated, using Bag-of-Words representations to build the vectors for each option. The cosine similarity function used is available from the ScikitLearn library[5].

Cosine similarity can be a useful tool for comparing the similarity of text sentences, but it is important to note that it only measures similarity based on the overlap of terms between the sentences. It does not take into account the context or meaning of the words, so it may not always accurately reflect the true similarity of the sentences.

Even though the difficulty of the text and the similarity of the options are not necessarily the only characteristics that can be correlated with difficulty, it was decided to occupy only these two categories to limit the scope of the research carried out, whose focus is mainly evaluation. of machine learning models. For this reason, in this work, we use features that have already been used in similar works (readability difficulty) and one that arises from a reasonable hypothesis raised during the work (option's similarity).

Throughout this chapter, the main methodologies required to train the predictive models of this work were introduced. It began with a brief introduction to what Natural Language Processing is and some of its foundational definitions. Then the two numerical representations used to vectorize text were introduced, illustrating the mathematics behind them and their main advantages and disadvantages. Both TF-IDF and GloVe are reasonable paths for any task that seeks to interpret text on a machine. Each one may be more convenient than the other depending on the problem. In particular, the NLP tasks explored in this work, and their performance will be compared in the following sections. In the same way, the performance of the characteristics presented at the end of this chapter will be studied in chapter 4, to see if they increase the predictive power of a difficulty estimation model.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine$_s$*imilarity.html*

Chapter 3

Subject Inference in Multiple-Choice Questions

3.1. Motivation and Data

As mentioned in Chapter 1, predicting the subject matter of a Multiple-choice question can bring a lot of benefits in the educational field, mainly because of the support it can provide for some types of applications or procedures as described in 1.1. For example, in the research project of which this paper is a part (in which other similar research papers have been developed), in some cases the data sources may not include the subject of a question, therefore, they are they must label to be able to analyze and study their composition, to find imperfections. A case in which being able to estimate the subject can be useful is the case in which it is intended to identify unusual words that can cause confusion or additional difficulty. Unusual words can have a different impact depending on the subject being tested, for example, technical words used in biology questions will be much less frequent in reading questions. Being able to accurately interpret the question's subject when it's not tagged can make a big difference when raising alerts for potential flaws like this.

In this section, a model for text classification is proposed to infer the subject matter of a Multiple-choice question. The algorithm used is Logistic Regression, which has been widely used for the text classification task and has proven to deliver good results [28], [29]. The motivation for doing this exercise comes from the fact that, although there is abundant research on text classification for various types of documents, this is not the case for questions from different subjects. In this sense, it is a natural exercise for the development of this work, testing the performance of the classification of subjects in the questions, with the same methods that are used for other types of documents. For this purpose, two different sets of MCQs are used to train and test the model. The first dataset has been provided by the *Educational Quality Agency*, an entity of the Chilean Ministry of Education. The organization creates a test called *Diagnóstico Integral de Aprendizaje* (Spanish translation for Integral Learning Diagnostic), to be applied to students from the second grade of primary school to the second grade of high school. The subjects of the available data are reading comprehension, mathematics, and social sciences. The second set of questions is the tests forms used in general university admissions in Chile from 2016 to 2022, provided by *DEMRE* which is the institution that designs the *PAES*, the test required to join higher education programs in Chile. This is a Multiple-choice questions test divided into four subjects (and hence, 4 tests): Mathematics, Reading Comprehension, Natural sciences, and Social sciences. Due to the difficulties in the parameterization process of the questions, only the questions of natural sciences and social sciences could be used. In future projects, it is intended to occupy all the questions.

With the first and second datasets, a third one is built by mixing all the questions from both. This allows us to make predictions for the 4 different subjects. The data was split with 75% of the samples designated for training (56.25% for pure training and 18.75% for validation) and the remaining 25% is used for testing. There is no general rule for the size of each set. Commonly the ratios vary from 65-35 to 80-20, for train-test sets.

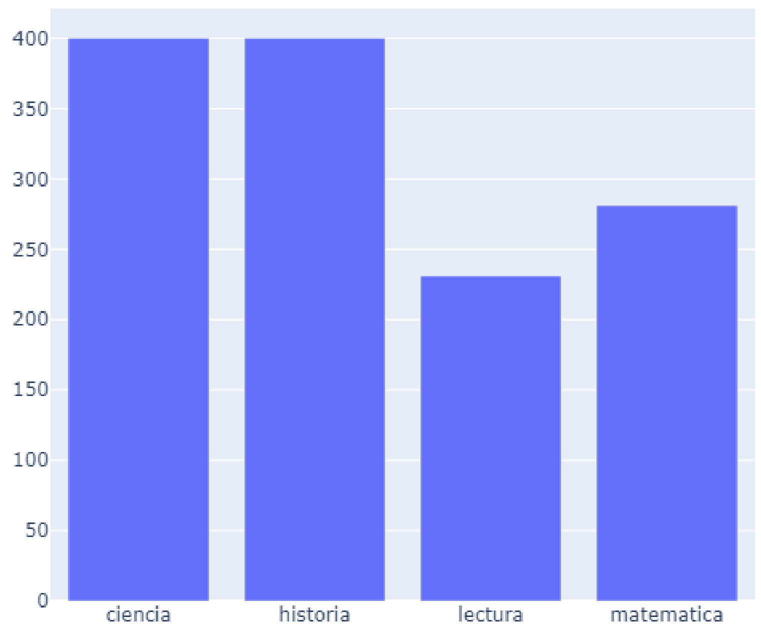

Figure 3.1: The 4 classes: ciencia (natural sciences), historia (social sciences), lectura (reading comprehension), and matematica (mathematics). Although the data set is not perfectly balanced, the minority class has more than half of the majority class.

3.2. Model Architecture and Results

The model proposed uses TF-IDF text representation with a Multi-label Logistic Regression model to infer four classes: Social sciences (labeled as *Historia*), Mathematics (labeled as *Matemática*), Reading Comprehension (labeled as *Lectura*) and Natural sciences (labeled as *Ciencia*). The TF-IDF matrix was built with unigrams and bi-grams, in addition, the

Spanish stopwords[6] were removed using the NLTK library [7]. To reduce dimensionality, only the top 1000 words (sorted by term frequency) from the vocabulary were included in the TF-IDF matrix.

The Logistic Regression model assumes the existence of K labels represented by y_i, with $i \in 1, \ldots, N$. There is also a matrix of coefficients W where each row vector W_k is associated with a class k. The main objective of Logistic Regression is to predict the probability of document X_i to be of class k:

$$P(y_i = k|X_i) = \frac{\exp(X_i W_k + W_{0,k})}{\sum_{t=1}^{K} \exp(X_i W_t + W_{0,t})} \tag{3.1}$$

where $W_{0,k}$ is the bias term associated to the k label. Notice that the numerator is equal to the dot product of X_i with W_k with the addition of $W_{0,k}$. The algorithm implementation used is the Logistic Regression class from ScikitLearn library [8]. To train the Logistic Model, the following objective function must be optimized:

$$\min_{W} -C \sum_{i=1}^{N} \sum_{0}^{K-1} \mathbb{1}_{y_i=k} \log(P(y_i = k|X_i)) + \frac{1}{2} \sum_{i}^{n} \sum_{j=1}^{K} W_{ij}^2 \tag{3.2}$$

The C value, which is the regularization term used to control the trade-off between the model's ability to fit the training data and the simplicity of the model is set to 10. The remaining arguments are the defaults from the library, such as the penalty function, which is of the type $l2$ and corresponds to the multiple sums in the right of (3.2).

The results of the recently described model gave an Accuracy of 93% in the test set. Considering that more than 9 out of 10 questions were correctly predicted in the test set, for a total of 4 possible classes, the result is considered, in general, good enough to be applied in some of the activities mentioned in chapter 1. When comparing the results with other text classification algorithms such as those proposed in [6], [7] and [8], it is clearer that the results are sufficient.

Table 3.1: Classification results in the test set.

Label	**Precission**	**Recall**	**F1-Score**	**Quantity**
Natural sciences	0.86	0.96	0.90	89
Social sciences	0.95	0.95	0.95	107
Reading comprehension	0.98	0.98	0.98	60
Mathematics	0.97	0.83	0.90	72

Almost all of the classification metrics are above 90%. The lowest precision observed is equal to 0.86 for the Natural Science class. This means that, for the total number

[6] words that are so common they are ignored by typical tokenizers
[7] https://www.nltk.org/
[8] https://scikit-learn.org/stable/index.html

of predictions made by the model for the Natural Sciences class, only 86% were correctly predicted. On the other hand, the lowest recall is 0.83 for Mathematics, meaning that only 83% of the questions in this class were correctly predicted. In this case, a possible reason is that, given the similarity between Natural Sciences[9] and Mathematics, the algorithm confuses the classes for some questions. However, for the subjects of Reading Comprehension and Social Sciences, this type of behavior does not occur.

Despite the different hypotheses that can be raised about the predictive behavior of the model, the factor of adjustment to the available data will always be present, especially in a situation with few available data. In this situation, a good strategy is to try to explain the model, based on the variables that are considered when the prediction is made, as will be seen in the next section.

3.3. Model Explanation with LIME

Explainability in machine learning refers to the ability of a model to provide explanations for its predictions or decisions. It allows us to understand how the model arrived at its conclusions and can help to identify potential biases or errors in the model. This is especially important in education, where the decisions made by a classification model can have significant consequences for students. For example, if the model is making predictions based on factors other than the relevant content of the questions, then this could result in a lack of accurate assessment of Multiple-choice questions. By providing explanations for the model's predictions, we can better understand how the model is making its decisions and identify any potential issues that need to be addressed.

In this work, the tool used to explain the predictions is *LIME* [30], which can provide detailed explanations for individual predictions made by a model. LIME refers to Model-agnostic Interpretable Local Explanations. It is a method for explaining the predictions of machine learning models, that is, identifying the most important variables in the model used to predict a value. It is agnostic to the model by applying to any machine learning model and local by using just a small piece of randomly generated data in the neighborhood of the input values.

LIME works by choosing a trained ML model and a single sample. Then, random numeric samples are generated *near* the numeric representation of the sample. Next, the generated samples are passed through the ML model to obtain their predictions. After that, an explainable model (such as Linear Regression) is trained with the random samples and their predictions from the trained ML model as labels. This results in a local interpretable model.

In this section, two samples of publicly available data [10] are tested with LIME. The first question is a Social sciences question:

"La Constitución de 1833 fue promulgada en los inicios de la República Conservadora y

[9] which, at least for the data from the PAES test, contains questions from the subtopics: Physics, Chemistry, and Biology

[10] https://demre.cl/publicaciones/modelos-resoluciones-pruebas

permitió consolidar un régimen de gobierno autoritario. Posteriormente, durante el periodo liberal, se mantuvo dicha constitución, pero se le hicieron reformas. Al respecto, ¿cuál fue uno de los objetivos de dichas reformas?"

In this case, the most important words are *gobierno* (translation for government), *periodo* (translation of period), *cual* (translation for which), *república* (translation of republic).

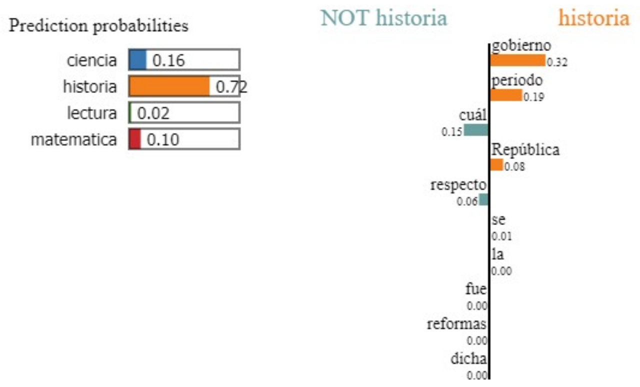

Figure 3.2: With 72% of probability, the model correctly predicts the question as *historia* (Social sciences). Just 5 words are enough to get almost 80% of the predictive power.

The predicted probability is 72% for the correct class. In this case, most of the relevant words are correctly identified. However, the model interprets the word "which" (which is clearly a common word and, therefore, shouldn't be representative of any subject) as an important word. One possible reason is that in the training set the occurrence of the word is not balanced (i.e. the word "which" is much more present in social science questions than in other subjects). In any case, this situation reflects an imperfection of the model that also occurs in more examples.

Another example is:

"Con la finalidad de conocer el estado de conservación de una especie de ave costera, se propone evaluar su densidad poblacional en el área que comprende su rango de distribución en Chile. Al respecto, ¿qué variables se requieren conocer para determinar la densidad poblacional de esta especie?"

In this case, the situation is similar. As can be seen (Figure 3.3), the word *especie* (translation for specie) is correctly labeled as a natural sciences word. However, other words also assigned to science do not add any relevance to the label at all. Moreover, the words *respecto* (translation for regard) and *conocer* (translation of know) are very common, regardless of the subject, but still, they are given considerable importance in predicting.

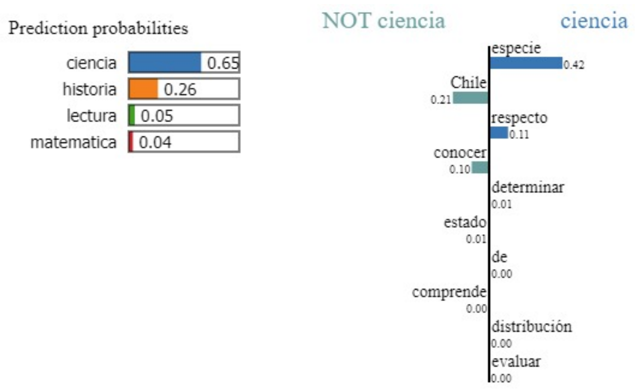

Figure 3.3: For this example, only 4 words are enough to get 84% of the importance in the prediction. Of the total words, 65% of the weight is assigned to the category of natural sciences. The rest is divided into the other 3 categories.

The main objective of using Lime in this project is to check if the model is using important data for the prediction (i.e. it took the words relevant to the document). For example, the word "Constitución" (translation for constitution), "reformas" (translation for reforms), and "liberal" (translation for liberal), are also expected to be relevant for the prediction in the first example, but that is not the case when using the analysis of LIME. The same occurs with words like "*densidad*" (translation for density) and *"poblacional"* (which means population in this context) for the second example, which is not relevant in the LIME analysis result. The most reasonable hypothesis for this behavior is that the train data fails to cover all the subjects with these frequent words, either because they are not included or considerably unbalanced between the subjects. Hence, some irrelevant words end up being important for classification.

Given the situation described above, it is not recommended to blindly trust this model. Although the performance is sufficient (based on the classification metrics obtained), the analysis made with LIME showed us that the model has flaws. It is suitable for the task under the scope of the source of data given, but problems could arise when using different sources for prediction. But if the model is trained with more sources and data, is expected that the prediction could fill the gaps that the current model has. The point here is not that the model must be fed with large volumes of data, but rather that the increase in data received must come from different sources or present significant variance in the contexts necessary to answer the questions. This would prevent the trained model from classifying certain types of words without a real value within a subject, simply because in the data that was trained, they coincidentally appeared in specific subjects with a higher proportion, or they only appeared once.

In this section, the multiple-choice question classification task was evaluated, mainly motivated by the possible applications described in chapter 1. A solution like this can be very valuable in the educational field, mainly for processing text in large volumes of questions. automatically. Despite the good results obtained with the logistic regression model on the data set, it was evidenced that it is not a sufficient reason to blindly believe in the model, thanks to the analysis made with LIME. For this reason, it is possible that by including other data sources the metrics used may be lower since all the available questions have similar contexts (all belong to tests applied in Chile, built by institutions dependent on the Chilean state, and massively applied). As future work, an improvement opportunity for the model is to include data sources with greater heterogeneity, which allow more complex scenarios to be simulated, to build a much more generalized model.

Chapter 4

Estimating the difficulty of multiple-choice questions

4.1. Motivation and Data

Difficulty estimation in Multiple-choice Questions is the process of determining how difficult it is to answer is a question. This can be useful for a variety of purposes, such as ensuring that a test or quiz has a balanced level of difficulty, or for providing feedback to students or test-takers about which questions they may have struggled with. It can be based on a combination of several factors, such as the number of correct answers provided by test-takers, the complexity of the question, and the amount of time it takes for test-takers to answer the question. Ultimately, the goal of difficulty estimation is to provide a measure of how challenging a given question is, so that test creators and educators can make informed decisions about how to use that question in a testing or learning context. In that sense, having a reliable prediction of the difficulty of the questions before they are applied in a real test can be of great help to test designers, since it allows them to adjust the mix of the questions and control the level of difficulty to be appropriate for students to measure

Predicting difficulty by traditional methods is an expensive task due to the need for experts in the subject matter, or by the time and resources involved in the pre-testing approach. To address these problems, some researchers have developed computational frameworks to estimate item difficulty. Some of them use feature extraction and then apply linear regression to estimate items' difficulty such as [23], which was a great advance in the matter and showed that there is an important effect between the way an item is written and its difficulty and discrimination. Others have used ontology-based models [31] to predict difficulty as category (*low*, *medium*, *high*), which could lead to correlations between the predicted and actual categories near to 80%. There are also custom Neural Network Architectures such as Document-enhanced Attention-based Neural Networks (DAN) proposed for item difficulty prediction in medical tests [32]. More recently, the estimation of items' difficulty with Transformers architectures [33] has shown the power of attention mechanisms.

In this work, three types of models are proposed and compared in the solving of this task, all of them based on Neural Network architectures: Recurrent Neural Networks (RNNs), Bidirectional Long Short-term Memory (BiLSTM), and Bidirectional Encoder Representations for Transformers (BERT). The work presented in this section not only seeks to evaluate the performance of neural networks to estimate difficulty but also to compare the different architectures used in other aspects apart from the mean square error of each model, such as the feasibility of its deployment in tools that can be used by multiple choice question builders. In that sense, it is not recommended to use the results of the mentioned works as the only point of reference to evaluate the models proposed here. Firstly, because they have different methodologies to process their data and to guide their experiments, and secondly, because the data sources used are different.

The data used in this chapter is the same provided by the *Educational Assessment, Measurement, and Registry Department* of Chile (DEMRE) used in the previous section: the natural and social sciences test forms for university admissions in Chile (from 2016 to 2022). In this case, the dataset provided by the *Educational Quality Agency* was not used because at the time it was developed it was not possible to label the difficulty in the format required by the implemented models. For this task, the difficulty of the items is not labeled using the Rasch model. In figures 4.1 and 4.2, we can see how they are distributed.

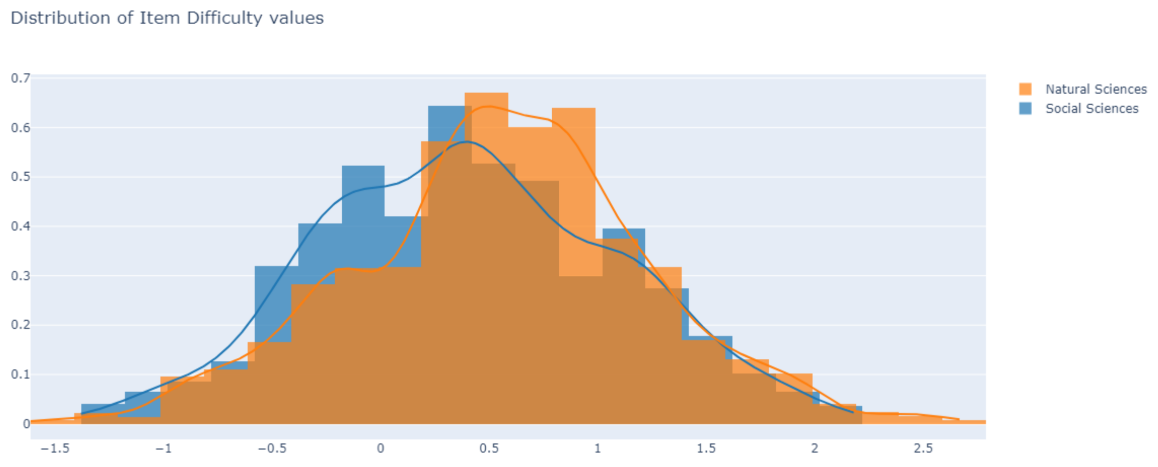

Figure 4.1: The distribution plot shows that the difficulty values go from −1.5 to 2.5. On average, the Natural sciences questions are more difficult than the Social sciences questions.

The dataset contains 4114 Multiple-choice Questions (Natural sciences: 3128|Social sciences: 986). The data was split by subject matter. The reason for this is that the goal is to make the estimate as accurate as possible. In principle, the traits that compose the difficulty of a question in a subject are not necessarily shared with other subjects.

For both sets, 75% of the samples are used for training (56.25% for pure training and 18.75% for validation) while the remaining 25% is used for testing.

4.2. Traditional Machine Learning and Recurrent Neural Networks

Before going completely to the use of neural networks, it is interesting to observe the behavior of at least one traditional machine learning algorithm and compare it with a dummy model (which predicts all values as the mean of the training data set), to check if is worth using Machine Learning models and to see if there is any substantial improvement with the use of neural networks.

The traditional model used in this section uses TF-IDF representation for text and the features described in 2.1. This model is trained to predict difficulty with a *Support Vector Regression* [34] (SVR). In general, Support Vector Machines are good when dealing with high-dimensional data. This is because Support Vector Machines find the hyperplane that maximizes the margin between the different classes in the training data. This can help reduce the complexity of the model and improve its generalization ability, which can in turn help prevent overfitting. In this case, the input data used has more columns than samples, so, it is a reasonable approach to go with SVRs. The parameters of the model were defined by a search[11] between the *linear* and *radial basis function* (RBF) kernels[12], and C[13] value between 1 to 10. The loss function used to find the estimators is the Mean Squared Error. The best parameters for both Social Sciences and Natural Sciences models were rbf kernel and C equals to 10.

The neural network model used in comparison to the model described above is a Recurrent Neural Network (See Appendix B). It is implemented by Tensorflow[14] and Keras[15] libraries, trained with Adaptative Moment Estimation algorithm (Adam) [35] as an optimizer and a *learning rate*[16] equals to 10^{-3}. It also uses Mean Squared Error as a loss function. The input is normalized by a Batch Normalization layer [17] and passed to a *Dense* layer which has 250 neurons and a *ReLu* activation function. Then, the flow is passed to a *Dropout*[18] layer which keeps only 50% of the data to prevent overfitting.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.model$_{s}$*election.GridSearchCV.html*

[12] A kernel is a function that transforms input data into a higher-dimensional space where it becomes linearly separable

[13] C is a regularization parameter that controls the trade-off between the complexity of the model and the amount of training data that is miss-classified. In general, a smaller C value will result in a more complex model, which can lead to overfitting, while a larger C value will result in a simpler model, which can lead to underfitting.

[14] https://www.tensorflow.org/

[15] https://keras.io/

[16] The parameter that determines the step size at which an optimizer minimizes a loss function

[17] *Applies a transformation that maintains the mean close to 0 and its deviation close to 1*

[18] *The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.*

Table 4.1: The results in the test set shows a minimal improvement from the dummy model by using SVR, while RNN performs 46% better in the Social sciences dataset and 31% in the Natural sciences dataset.

	Social sciences		Natural Sciences	
Model	**MSE**	**MAE**	**MSE**	**MAE**
Dummy model	0.50	0.56	0.48	0.54
SVR	0.48	0.55	0.45	0.52
RNN	0.27	0.40	0.33	0.42

In figure 4.1, it is observed that both proposed models perform better than the dummy model for the two metrics presented: *Mean Squared Error* (MSE) and *Mean Absolute Error* (MAE) (see Appendix A.), but, the improvement from SVR model is minimal: only 4% and 6.3% better than the dummy model for Social sciences and Natural sciences respectively when comparing MSE, but when comparing MAE they have only 1.8% and 3.7%. On the other hand, the RNN model has a 46% and 31% of MSE improvement and a 29% and 22% of MAE improvement over the dummy model. Both MSE and MAE represent the difference between the predicted and actual values. However, MSE is more sensible to outliers by using the squared values. At least for count-based representations, it is clear that RNNs considerably overcome SVR. That doesn't mean that RNNs are always a better choice, but, due to the results, is reasonable to think that is much more appropriate to use Neural Network Architectures, instead of traditional machine learning models (like SVR) for the difficulty estimation task.

4.3. Long Short-term Memory model

Neural networks have proven to be a satisfactory approach for most NLP tasks. Due to Backpropagation [36] algorithm, RNNs can perform much better than traditional machine learning models for several tasks. For example, in the next sentence prediction task, if the sentence or document is short, RNNs should perform well, but problems arise if the length of the document is larger. In practice, RNNs are unable to learn patterns from this type of data [37]. When using RNNs for natural text, the first words of the document lose relevance along the sequence. When the sequence of words given to a simple RNN is large enough, they start to forget words (vanishing gradient problem [38]), making them irrelevant to the prediction. In simple words, they have *short memory*. The *Long Short-term Memory* [39] model (LSTM), is a considerable improvement over simple RNNs (it is important to keep in mind that LSTMs are a special type of RNNs) in this type of task. LSTM models perform better when working with longer word sequences, due to the addition of gate mechanisms in their architecture, which allow the model to keep important information for longer periods, while irrelevant information is discarded. In this section, the LSTM model is reviewed and then, three models of it are proposed.

The core element in an LSTM network is the *cell state*. It reflects the state of the main flow of the network and spans the entire chain. The cell state interacts with 3 gates. These gates are sets of point-wise operations and Neural Network layers that allow information to pass into the cell. They also remove current information from the cell state. One of the gates is the *forget gate*. It is composed of a sigmoid function[19] and a point-wise multiplication operation. It receives as input the output from the previous Output gate (as can be seen in 4.2). Since a sigmoid is a function that goes from 0 to 1, the forget gate decides if all of the previous information received will be added to the cell state (sigmoid returns 1), none of the previous information (sigmoid returns 0) or just a piece of it.

Then, the input gate, decides which information is accepted to flow to the cell state, among all that is received. It takes as input, the input data itself, unlike forget gate which takes as input the previous output. First, it must pass through two layers: a sigmoid and a tanh[20] layer. The output from both layers is then multiplied and added to the output of the Forget gate, obtaining the new cell state as:

$$C_t = C_{t-1} \cdot f_t + i_t \cdot \tilde{C}_t \tag{4.1}$$

Where C_{t-1} is the previous Cell State, f_t is the output of the Forget Gate, i_t is the output of the sigmoid layer from the Input Gate, and $\tilde{C}_t$ is the output of the tanh layer (also from the Input Gate).
S

After passing the first two gates, the cell state is updated, removing the information that the forget gate decided to forget, and adding the information that the input gate decided to add. Now the cell needs to create an output. The output gate receives a filtered version of the updated cell state. First, the previous state, concatenated with the new input, passes through a sigmoid layer, which decides the content of the cell state that will be taken as output. Then, the cell state passes to a tanh function and multiplies it with the output of the sigmoid gate, obtaining only the information desired as output.

$$h_t = o_t \cdot tanh(C_t) \tag{4.2}$$

This is just a brief explanation of what an LSTM cell is (an LSTM network is a stack of those cells). However, the LSTM can improve their context understanding by being **bidirectional**, i.e. they can be trained from left to right and right to left in the same network[21].

[19] A function that is commonly used for modeling probabilities
[20] Hyperbolic tangent
[21] It is important to keep in mind that this is not a reference to the backpropagation algorithm, this only refers to the fact that the algorithm has trained both ways.

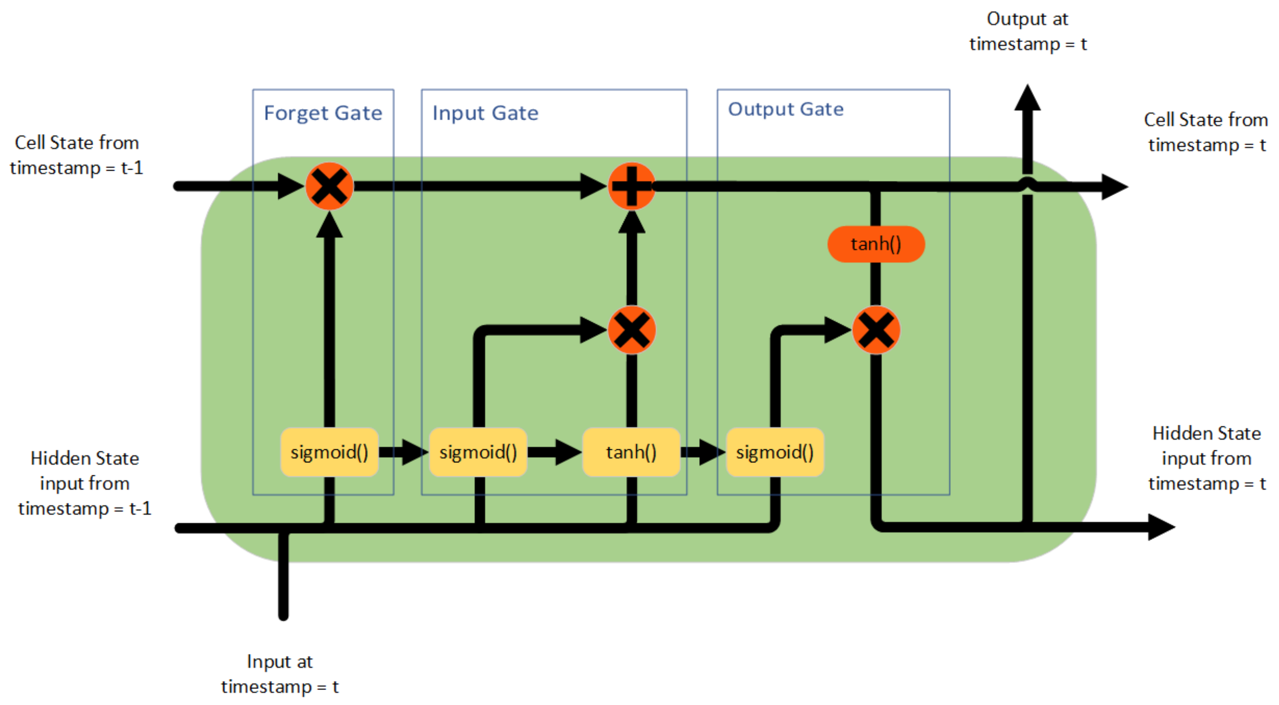
Output at timestamp = t
Forget Gate
Input Gate
Output Gate
Cell State from timestamp = t-1
Cell State from timestamp = t
tanh()
sigmoid()
sigmoid()
tanh()
sigmoid()
Hidden State input from timestamp = t-1
Hidden State input from timestamp = t
Input at timestamp = t

Figure 4.2: An LSTM cell. The yellow blocks are Neural Network Layers, the red circumferences are point-wise operations and each arrow is a vector transfer that can be copied (when an arrow splits in two) or concatenated (when two arrows meet). A cell has 3 *inputs*: the text itself, the previous hidden state, and the previous cell state. The hidden state data and input data are concatenated and pass through all the gates. The gates decide what information will be added, discarded, or forgotten in the Cell state and Output. The Cell State receives all the updates and then is given as input to the output gate (i.e. the next hidden state).

There are three models proposed in this section: (i) is a simple Bidirectional LSTM that only receives the item's stem as input, (ii) is the model (i) but includes all the features from section 2.3, and (iii), the model (ii) using also the text of the item's key in an isolated set of Bidirectional LSTM layers.

The root model architecture is shown in figure 4.3. Section one consists in an embedding layer that transforms text to a numeric representation with GloVe embeddings. It is followed by a Batch Normalization layer and passed to a Bidirectional LSTM layer. After that, 50% of the output is discarded by a Dropout layer. Finally, the information goes through a Flatten layer[22].

Section two receives input data as numbers, which are normalized before being passed to an RNN. Then, the output goes through a Dropout layer, to end up being flattened. The flattened output from both sections is then concatenated, before being passed to a linear activation which predicts the final values. In this case, the items' difficulty.

[22] It flattens the input. For example, if flatten is applied to a layer having an input equal to (batch size, 3,3), then the output shape of the layer will be (batch size, 6)

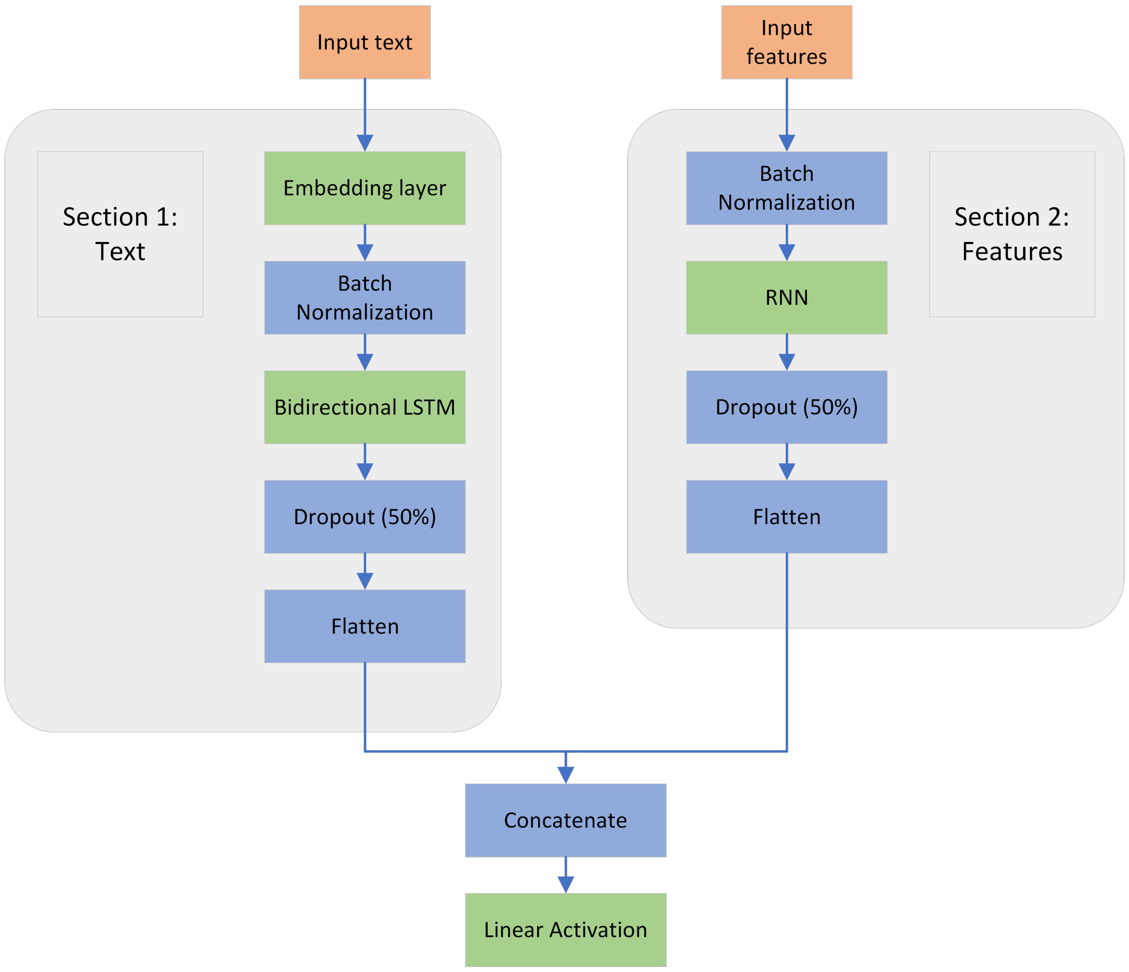

Figure 4.3: The Bidirectional LSTM root model for each of the proposed models. Model (i) has only section 1 and skips the concatenation step. Model (ii) is the one shown figure. Model (iii) has an additional section identical to section 1, but it receives the keys' text (instead of the stems' text) as input.

All models were trained using *Tensorflow* and *Keras*, with Adam as optimizer and learning rate equal to 10^{-3} and Mean Squared Error as loss function. Also, all models were trained using 150 Epochs with patience[23] of 15.

For all the models, the text representation used was the 25-dimensional pre-trained of GloVe[24], trained with 2 Billion tweets (and 27 Billion tokens).

25% of the data was intended for testing, and for the remaining training data, 25% of it (18.75% of the total) was intended for validation. The sampling was random.

[23] Refers to the limit of consecutive epochs allowed with no improvement
[24] https://nlp.stanford.edu/projects/glove/

Table 4.2: Model results in the test set. The RNN model from the previous section is also shown in the table to be compared with the Bidirectional LSTM models.

Model	Social sciences		Natural sciences	
	MSE	**MAE**	**MSE**	**MAE**
RNN	0.27	0.40	0.33	0.42
(i) BidLSTM (only stem)	0.27	0.29	0.12	0.16
(ii) BidLSTM + Features (only stem)	0.30	0.31	0.11	0.15
(iii) BidLSTM + Features	0.27	0.29	0.11	0.15

The results show that the Bidirectional LSTM model is, in general, an improvement over the RNN model. For the Social sciences questions, the Mean Squared Error (MSE) remains the same, but the Mean Absolute Error (MAE) decreases considerably. The models have an interesting behavior because adding more attributes and variables does not cause an improvement in the prediction. Instead, they could get even worse as shown in model (ii). On the other side, using this architecture for the Natural sciences questions gives better results overall. Model (i) is 64% better than the RNN model, and both models (ii) and (iii) are almost 67% better considering the MSE metric.

4.4. Pre-trained Transformers models

Transformers are deep learning models that rely on the *self-attention* mechanism [40]. Unlike RNNs, when processing sequences, they can use context for any position in the sequences due to their ability to process them in parallel by self-attention mechanism. In particular, pre-trained transformers models that are trained with *big corpora* like BERT [41] have become the current state of the art for multiple NLP tasks, such as masked language modeling, named entity recognition, and machine translation. In this section, we made a quick review of what Transformers and BERT are, and introduce a Deep Neural Network model that leverages two pre-trained models and the features obtained in section 2. Then we evaluate different model configurations in both history and sciences datasets.

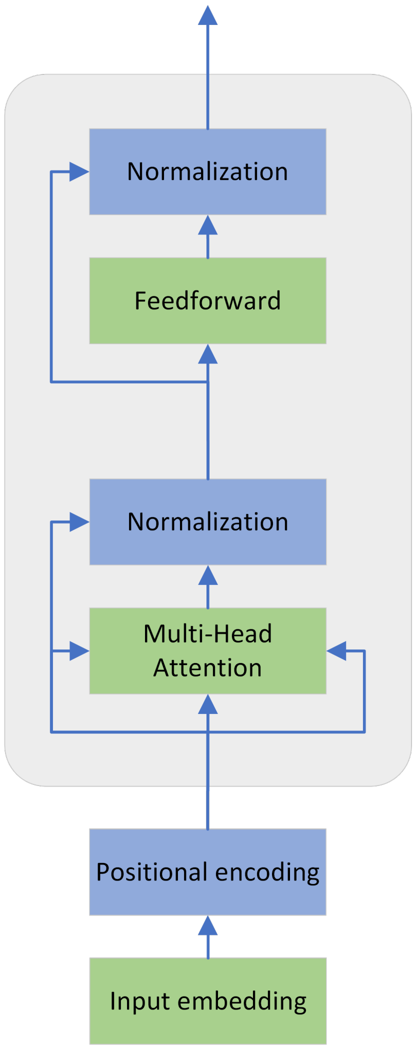

Figure 4.4: Encoder section of a Transformer model.

Transformers are an encoder-decoder architecture. They accept text input as word embeddings and add a positional encoding to each word in the sequence. This positional encoding is not an absolute position representation of it, instead, it captures which words tend to go together within a sequence. Then the input is fed to an encoder architecture whose job is to map all the sequences into a new representation.

The encoder section is represented by N_x in figure 4.2. The first element is a Multi-Head Attention layer. This part is the key element in transformer architecture. It is an extended version of a Self Attention layer that allows the Transformer to focus on different positions while encoding, for better outputs. Initially, *Self-attention* mechanism will be briefly explained, more precisely, the **Scaled Dot-product Attention** which is the one that Multi-head Attention layers use in Transformers implementation.

For each word three vectors are created by multiplying each token (with the embeddings dimensions defined as (1x512)) by three matrices: W^Q, W^K, and W^V (*Query*, *Key* and *Value*) each one with dimensions (512x64), so the resultant vectors will be Q, K and V of dimension (1x64).

In the Scaled Dot-product architecture (see figure 4.5), the input passed begins with Q and K to first compute the dot-products between them (first layer) and then divide it by $\sqrt{d_k}$ (second layer). A *Softmax* function is applied over the resultant scores (third layer) and to finally multiply with the V value vectors (fourth layer). For simplicity, the output of a Scaled Dot-product attention function is:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4.3}$$

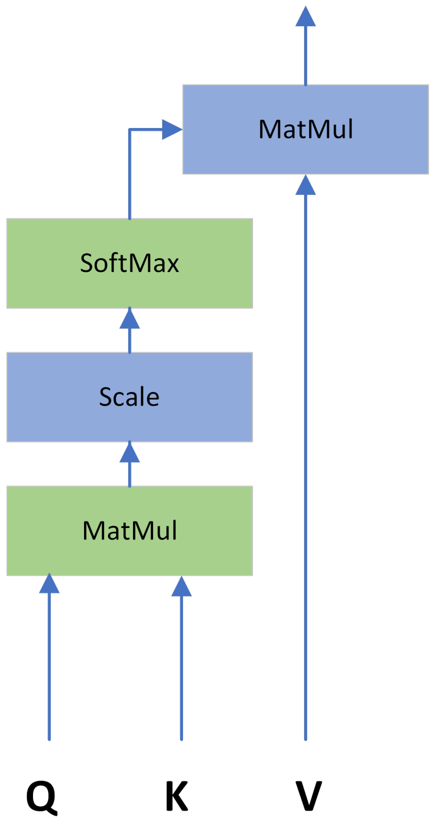

Figure 4.5: Scaled Dot-product Attention Architecture.

On the other hand, Multi-head Attention allows the performance of multiple Scaled Dot-product Attention for a finite number of different linear projections of Q, K, and V, i.e. it allows the model to jointly attend to information from different representation subspaces at different positions. These projections are then concatenated and projected to a reduced dimension.

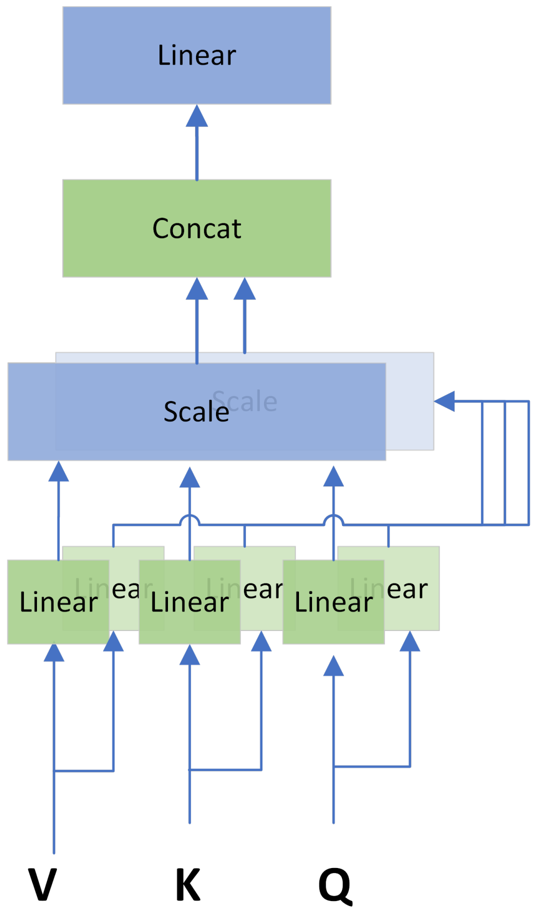

Figure 4.6: Multihead Attention Architecture.

To summarize, the output of the Multi-head Attention layer is computed as:

$$\text{Multihead Attention}(Q,\ K,\ V) = \text{Concat}(head_1, head_2, \ldots, head_h)W^O \tag{4.4}$$

Where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),\ i \in \{1, 2, \ldots, h\}$

The Multi-head Attention output is then normalized and passed to a point-wise feed-forward network that has identical parameters for each position. The paper refers to it as a separate, identical linear transformation with a ReLu activation in between. Afterward, the output is passed to an encoder-decoder multi-head attention layer inside the decoder section (the second attention layer of the decoder). For the context of this project, we are skipping the remaining part of the Transformers model because BERT only uses the encoder part.

BERT refers to Bidirectional Encoding Representation from Transformers. It is a way of learning representations of a language that uses the encoder part of the transformer. Therefore, it does not use recurrent connections and relies only on attention, normalization, and feedforward layers. However, it is not just a transformer's encoder. It has different hyperparameters and dimensions and is also intended to be used differently. Similarly to the previously proposed model, it is also bidirectional and, hence, gathers the context of previous and following words within a sequence. BERT models are trained in an unsupervised way. Unlike Transformers models, BERT models sentence embeddings, instead of token embeddings[25].

Input data in the BERT model uses 3 embedding layers:

1. Token Embedding: It tokenizes text as previous models but separates sentences by two new tokens: [CLS] and [SEP]. Let's illustrate with an example, If we have two sentences:

   a) "Tomorrow is my birthday"
   b) "you will be 23"

   Usually, the tokens are {Tomorrow, is, my, birthday, you, will, be, 23}. The token embedding layer adds new tokens. [CLS] to mark the beginning of the document (before the first sentence), and [SEP], the division between both sentences. So, the final tokens are:

$$\{[\text{CLS}],\ \text{Tomorrow},\ \text{is},\ \text{my},\ \text{birthday},\ [\text{SEP}],\ \text{you},\ \text{will},\ \text{be},\ 23\} \tag{4.5}$$

2. Segment Embedding: It adds an indicator for each token representing the sentences they are in.

3. Position Embedding: It adds information about the word order.

The final representation for the input embedding is:

Layer	[CLS]	Tomorrow	is	my	birthday	[SEP]	you	will	be	23
Token	$E_{[CLS}$	E_{Tomorrow}	E_{is}	E_{my}	E_{birthday}	$E_{\text{[SEP]}}$	E_{I}	E_{will}	E_{be}	E_{23}
Sentence	E_{A}	E_{A}	E_{A}	E_{A}	E_{A}	E_{A}	E_{B}	E_{B}	E_{B}	E_{B}
Position	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

BERT entails two processes: Pre-training and Fine-tuning. The pre-training step consists in training the model to solve two tasks: Masked Language Modelling[26] and Next Sentence Prediction[27] for any type of text input. Using both helps the model to fuse the left and right context, obtaining a bidirectional representation of text with text-pair representations. Then, it can be used as a feature representation of words with its pre-trained weights, or *fine-tuned* for a specific task. In the fine-tuning process, the model can be trained to solve any other task, using the pre-trained model as a starting point.

The models proposed in this project are highly influenced by the work presented by Benedetto, et al [42] because it leverages the output embedding by the "[CLS]" token and its

[25] embeddings for a single word.
[26] Consists in randomly masking words within sequences and training a model to be able to predict which is the masked word.
[27] Similarly to MLM, a model is trained to predict if a sequence A should follow a sequence B or not.

connection to a dense layer. In this work, the features presented in chapter 2, are used in conjunction with the pre-trained model. It is also compared how the models behave when BERT is used only as a pre-trained language model with non-trainable parameters in the embedding layer and with trainable parameters applying a fine-tuning procedure. There are also variations applied to each model by adding the same features used by the BiLSTM model proposed in 4.6.

The pre-trained models used in this project are *DistilBert* [43] (in its multilingual and cased[28] version) and BETO [44], a BERT model trained with a big Spanish corpus.

Table 4.3: Results of the models' prediction over the test set.

Model	Social sciences		Natural sciences	
	MSE	**MAE**	**MSE**	**MAE**
BETO (no fine tuning)	0.43	0.52	0.34	0.46
BETO (fine tuning)	0.50	0.57	0.48	0.54
Custom BETO (no fine tuning)	0.43	0.52	0.31	0.30
Custom BETO (fine-tuning)	0.60	0.63	0.48	0.54
DistilBERT (no fine-tuning)	0.47	0.55	0.43	0.53
DistilBERT (fine-tuning)	0.5	0.56	0.46	0.54
Custom DistilBERT (no fine-tuning)	0.43	0.53	0.48	0.54
Custom DistilBERT (fine-tuning)	0.65	0.66	0.44	0.51

Two important characteristics are observed in the results. First, none of the BERT models manage to outperform the models described above. The model based on RNNs with TF-IDF representation is performing better than all the models presented in the table, except *custom BETO* without the application of fine-tuning. This event is interesting, considering that algorithms based on self-service mechanisms tend to perform better for tasks that involve text.

Secondly, it is observed that the fine-tuning process does not improve the results of the pre-trained models for this specific task. This may be due to different reasons, some of them, attributable to the experimental configuration used, a possible overfit to the training data, also taking into account that the size of the available dataset might not be enough for training. Additionally, it is possible that the parameters used in the model aren't those that maximize its performance in the test set. Due to the capacity of the hardware used (since the fine-tuning of these models required training more than one hundred million parameters), it was not possible to validate the model during training nor to do parameter searches, therefore, there is room for improvement in the model's configuration. However, this also allows us to visualize the disadvantages of using BERT, compared to the deep learning methodologies described previously, which have a low computational cost and might be more appropriate for inferring difficulty within an application.

[28] It discriminate between words with lowercase and uppercase letters.

Chapter 5

Conclusion

Throughout this work, the relevance of multiple-choice questions has been introduced along with opportunities for improvement in estimating psychometric attributes. As presented here, having reliable estimations for both the subject and the difficulty can help to improve education in many ways. On the one hand, having a good estimation of the subject can help to improve the efficiency of educational platforms or systems, along with understanding better what is relevant for a question to its subject. On the other hand, being able to predict the difficulty of an MCQ item can help build tests that can correctly assess students. Furthermore, if this prediction is automatic and low-cost, its impact is further amplified by allowing rapid testing of the difficulty of these questions. This work has proposed alternatives for improvement in the estimation of these two attributes using machine learning algorithms. The main goal was to measure and interpret the models instead of comparing them just by loss and classification metrics. Sometimes, higher accuracy is less important than the feasibility and costs of implementing a predictive model, and that was the spirit of this work.

For the subjects' estimation, it was shown that traditional machine learning algorithms can predict the items' subjects successfully. Furthermore, the results in the test set have over 90% accuracy and similar results for each one of the other classification metrics: precision, recall, and F1-score. However, the analysis made with LIME showed that as some words relevant to the subject have a higher importance in the prediction, some others, that shouldn't be representative of a subject, also have a high predictive power. This shows that the model is largely based on the available data and this type of relationship between the words and the subjects is directly associated with the distribution of the data. Despite having good results, this situation makes the model hard to trust, due to the uncertainty of the prediction for a different data source. However, when predicting the items' subject is not a crucial part of a product or service, it could still be useful to have this estimation for a search service or grouping questions, just to give an example. For future research, having different sources of data could help to obtain more reliable estimations that focus on using representative words to predict the subject.

On the other hand, items' difficulty has taken a lot of attention from researchers in recent years. Having an accurate prediction of the items' difficulty is very important when tests are designed. Difficulty affects how an MCQ discriminates between test takers and how reliable is the interpretation of the results. In addition, different versions of a test must be balanced by their weighted difficulty to be fair. The traditional methods for estimating difficulty

aren't optimal in terms of costs and accuracy, and many researchers have found useful applications of machine learning algorithms to address the flaws that traditional methods for estimating difficulty have. In this project, it was shown that traditional machine learning isn't a successful method for difficulty estimation, but using count-based representations (as TF-IDF) with RNNs is a good place to start and can have surprisingly good results compared to most recent architectures. Pre-trained Transformers models (more precisely: BERT), despite being considered state-of-the-art for most NLP tasks today were surpassed in performance by previous architectures. The big winner here is the Long Short-time Memory model with bidirectional architecture and pre-trained word embeddings, not only for getting the best results but also by being considerably more efficient than Transformers architecture. The Bidirectional LSTM model can be used by educational organizations to have a proxy for a test's difficulty. However, the author's recommendation is to not rely only on the model prediction, but use it in conjunction with more precise methods instead, like choosing candidate items from a bank of items using the estimates of this model and then picking the best items using pre-testing.
| Finally, the main findings of this work, applicable to the estimation of both attributes, can be summarized in two. The first is that, although the results are satisfactory for both tasks compared to the reviewed bibliography, they have imperfections that may reduce the scope of their use. Certainly, machine learning algorithms can make numerous processes more efficient, however, it is debatable whether they can replace entirely other methodologies for estimating attributes, especially those involving human collaborators. The reason for this idea is not because their estimates are not close enough to the real values, but because the estimation of difficulty is a process that has a significant effect on the future of students and requires to be treated with great care. In this sense, as mentioned in the previous paragraph, the best path may be the collaboration between both methodologies, achieving more efficient and reliable processes.

Second, as observed in the comparison between transformers and Bidirectional LSTM, some models are not as modern or complex and are usually cheaper to train and use. The more complicated models tend to compensate for their high computational cost and complexity with surprisingly good results, however, at least for this task this was not the case. Regardless of the difference between these two architectures, it is important to highlight that the performance of the model with RNNs was very similar to that of the Bidirectional LSTM, despite having a much simpler architecture. The idea behind these points is that algorithms can respond differently to each problem and those that are appropriate for one task are not necessarily for others. Being open to testing new methodologies is just as valuable as being willing to try old ones, even when they haven't delivered good results on other tasks.

Bibliography

[1] Gierl, M., Bulut, O., Guo, Q., y Zhang, X., "Developing, analyzing, and using distractors for multiple-choice tests in education: A comprehensive review," Review of Educational Research, vol. 87, p. 0034654317726529, 2017, doi:10.3102/0034654317726529.

[2] Haladyna, T. y Rodriguez, M., Developing and Validating Test Items. Taylor & Francis, 2013, https://books.google.cl/books?id=DQNxvjHfJxAC.

[3] Jiménez, Hernández, Hernández, y González, "Evaluación de reactivos de opción múltiple en medicina. evidencia de validez de un instrumento," no. 21, 2017, doi:10.1016/j.riem.2016.04.005.

[4] Indrayani, M., Marhaeini, A., Paramartha, A., y Wahyuni, L., "The analysis of the teacher-made multiple-choice tests quality for english subject," Journal of Education Research and Evaluation, vol. 4, p. 272, 2020, doi:10.23887/jere.v4i3.25814.

[5] Haladyna, T. y Rodriguez, M., Developing and Validating Test Items. 2013, doi:10.4324/9780203850381.

[6] Sunagar, P., Kanavalli, A., Nayak, S. S., Mahan, S. R., Prasad, S., y Prasad, S., "News topic classification using machine learning techniques," en International Conference on Communication, Computing and Electronics Systems (Bindhu, V., Tavares, J. M. R. S., Boulogeorgos, A.-A. A., y Vuppalapati, C., eds.), (Singapore), pp. 461–474, Springer Singapore, 2021.

[7] Awad, W. A. y Elseuofi, S., "Machine learning methods for e-mail classification," International Journal of Computer Applications, vol. 16, pp. 39–45, 2011.

[8] Lee, K., Palsetia, D., Narayanan, R., Patwary, M. M. A., Agrawal, A., y Choudhary, A., "Twitter trending topic classification," en 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 251–258, 2011, doi:10.1109/ICDMW.2011.171.

[9] Qu, B., Cong, G., Li, C., Sun, A., y Chen, H., "An evaluation of classification models for question topic categorization," Journal of the American Society for Information Science and Technology, vol. 63, 2012, doi:10.1002/asi.22611.

[10] Schubotz, M., Scharpf, P., Teschke, O., Kühnemund, A., Breitinger, C., y Gipp, B., "Automsc: Automatic assignment of mathematics subject classification labels," en Intelligent Computer Mathematics (Benzmüller, C. y Miller, B., eds.), (Cham), pp. 237–250, Springer International Publishing, 2020.

[11] Rasch, G., "Probabilistic models for some intelligence and attainment tests," 1980, https://www.jstor.org/stable/2287805?origin=crossref.

[12] Goldberg, Y., "Neural network methods for natural language processing," Synthesis Lectures on Human Language Technologies, vol. 10, no. 1, pp. 1–309, 2017, doi:https:

//doi.org/10.2200/S00762ED1V01Y201703HLT037.

[13] Eisenstein, J., "Natural language processing," Jacob Eisenstein, 2018.

[14] Oettinger, A. G., "Computational linguistics," The American Mathematical Monthly, vol. 72, no. 2, pp. 147–150, 1965, http://www.jstor.org/stable/2313322 (visitado el 2022-12-11).

[15] Jie, L., Jiahao, C., Xueqin, Z., Yue, Z., y Jiajun, L., "One-hot encoding and convolutional neural network based anomaly detection," Journal of Tsinghua University (Science and Technology), vol. 59, no. 7, pp. 523–529, 2019.

[16] Zhang, Y., Jin, R., y Zhou, Z.-H., "Understanding bag-of-words model: a statistical framework," International Journal of Machine Learning and Cybernetics, vol. 1, no. 1-4, pp. 43–52, 2010, doi:10.1007/s13042-010-0001-0.

[17] Sammut, C. y Webb, G. I., eds., TF–IDF, pp. 986–987. Boston, MA: Springer US, 2010, doi:10.1007/978-0-387-30164-8_832.

[18] Bakarov, A., "A survey of word embeddings evaluation methods," CoRR, vol. abs/1801.09536, 2018, http://arxiv.org/abs/1801.09536.

[19] Pennington, J., Socher, R., y Manning, C. D., "Glove: Global vectors for word representation," en Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, 2014, http://www.aclweb.org/anthology/D14-1162.

[20] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., y Dean, J., "Distributed representations of words and phrases and their compositionality," Advances in neural information processing systems, vol. 26, 2013.

[21] Mikolov, T., Chen, K., Corrado, G., y Dean, J., "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[22] Zamanian, M. y Heydari, P., "Readability of texts: State of the art.," Theory & Practice in Language Studies, vol. 2, no. 1, 2012.

[23] García, C., Ponsoda, V., y Sierra, A., "Prediction of item psychometric indices from item characteristics automatically extracted from the stem and option text," International Journal of Continuing Engineering Education and Life-Long Learning, vol. 213, pp. 210–221, 2011, doi:10.1504/IJCEELL.2011.040199.

[24] Benedetto, L., "Introducing a framework to asses newly created questions," 2020, https://link.springer.com/chapter/10.1007/978-3-030-52237-7_4.

[25] Flesch, R., "A new readability yardstick.," 1948, doi:https://doi.org/10.1037/h0057532.

[26] Smith, E. A. y Senter, R., "Automated readability index.," AMRL-TR. Aerospace Medical Research Laboratories, pp. 1–14, 1967.

[27] Spaulding, S., "A spanish readability formula," The Modern Language Journal, vol. 40, pp. 433–441, 1956.

[28] Pranckevičius, T. y Marcinkevičius, V., "Application of logistic regression with part-of-the-speech tagging for multi-class text classification," en 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), pp. 1–5, 2016, doi:10.1109/AIEEE.2016.7821805.

[29] Shah, K., Patel, H., Sanghvi, D., y Shah, M., "A comparative analysis of logistic

regression, random forest and knn models for the text classification,” Augmented Human Research, vol. 5, 2020, doi:10.1007/s41133-020-00032-0.

[30] Ribeiro, M. T., Singh, S., y Guestrin, C., “"why should I trust you?": Explaining the predictions of any classifier,” en Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1135–1144, 2016.

[31] V, V. E., Alsubait, T., y Kumar, P. S., “Modeling of item-difficulty for ontology-based mcqs,” CoRR, vol. abs/1607.00869, 2016, http://arxiv.org/abs/1607.00869.

[32] Qiu, Z., Wu, X., y Fan, W., “Question difficulty prediction for multiple choice problems in medical exams,” en Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM ’19, (New York, NY, USA), p. 139–148, Association for Computing Machinery, 2019, doi:10.1145/3357384.3358013.

[33] Benedetto, L., Aradelli, G., Cremonesi, P., Cappelli, A., Giussani, A., y Turrin, R., “On the application of transformers for estimating the difficulty of multiple-choice questions from text,” en Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications, (Online), pp. 147–157, Association for Computational Linguistics, 2021, https://aclanthology.org/2021.bea-1.16.

[34] Drucker, H., Burges, C., Kaufman, L., Smola, A., y Vapnik, V., “Support vector regression machines,” Adv Neural Inform Process Syst, vol. 28, pp. 779–784, 1997.

[35] Kingma, D. P. y Ba, J., “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[36] Rumelhart, D., Hinton, G., y Williams, R., “Learning representations by back-propagating errors,” Nature, vol. 323, no. 1, p. 533–536, 1986.

[37] Hochreiter, S., “Untersuchungen zu dynamischen neuronalen netzen,” Diploma, Technische Universität München, vol. 91, no. 1, 1991.

[38] Basodi, S., Ji, C., Zhang, H., y Pan, Y., “Gradient amplification: An efficient way to train deep neural networks,” Big Data Mining and Analytics, vol. 3, no. 3, pp. 196–207, 2020, doi:10.26599/BDMA.2020.9020004.

[39] Hochreiter, S. y Schmidhuber, J., “Long short-term memory,” Neural computation, vol. 9, pp. 1735–80, 1997, doi:10.1162/neco.1997.9.8.1735.

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I., “Attention is all you need,” CoRR, vol. abs/1706.03762, 2017, http://arxiv.org/abs/1706.03762.

[41] Devlin, J., Chang, M., Lee, K., y Toutanova, K., “BERT: pre-training of deep bidirectional transformers for language understanding,” CoRR, vol. abs/1810.04805, 2018, http://arxiv.org/abs/1810.04805.

[42] Benedetto, L., “On the application of transformers for estimating the difficulty of multiple-choice questions from text,” 2021, https://aclanthology.org/2021.bea-1.16/.

[43] Sanh, V., Debut, L., Chaumond, J., y Wolf, T., “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” ArXiv, vol. abs/1910.01108, 2019.

[44] Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., y Pérez, J., “Spanish pre-trained bert model and evaluation data,” en PML4DC at ICLR 2020, 2020.

[45] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," 1957, https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf.

Annexes

Annexes A. Metrics

A.1. Classification

1. Precision: Percentage of correct positive predictions relative to total positive predictions.
2. Recall: Percentage of correct positive predictions relative to total actual positives.
3. F1 Score: A weighted harmonic mean of precision and recall. The closer to 1, the better the model.

$$\text{F1 Score} = 2 \cdot \frac{(\text{Precission} + \text{Recall})}{(\text{Precission} - \text{Recall})} \tag{A.1}$$

A.2. Regression

The metrics used are Mean Squared Error (MSE) and Mean Absolute Error (MAE).

1. MSE:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{\hat{y}^2 - y^2} \tag{A.2}$$

2. MAE:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{n} |\hat{y}^2 - y^2| \tag{A.3}$$

Annexes B. Neural Networks

B.1. The basics

Neural Networks are a type of Machine Learning architecture that is composed of a set of neurons interconnected with each other. These neurons represent a simple computational unit that is trained to receive an input and produce an output. The input received consists of the weighted sum of each previous neuron that is connected to it. The output is obtained by an activation function which receives the weighted sum and applies it. A common type of activation function is the Rectified Linear Unit:

$$\text{ReLU}(z = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + b) = \max(0, z) \tag{B.1}$$

Where z is the dot product between the outputs of the neurons in the previous layer $X = (x_1, x_2, \ldots, x_n)$ with the weights associated with the current neuron $W = (w_1, w_2, \ldots, w_n)$, plus a bias term b.

In a Neural Network, neurons are stacked together and organized into a layer that receives input data from the previous layer and produces an output that serves as input for the next layers. The input layer is the one that receives input data for the entire network, and the output layer is the output of the network. The layers in between are called hidden layers. This is where most of the parameters are trained to produce an output. The goal of this model is to find the best weights and biases for the task that is trained.

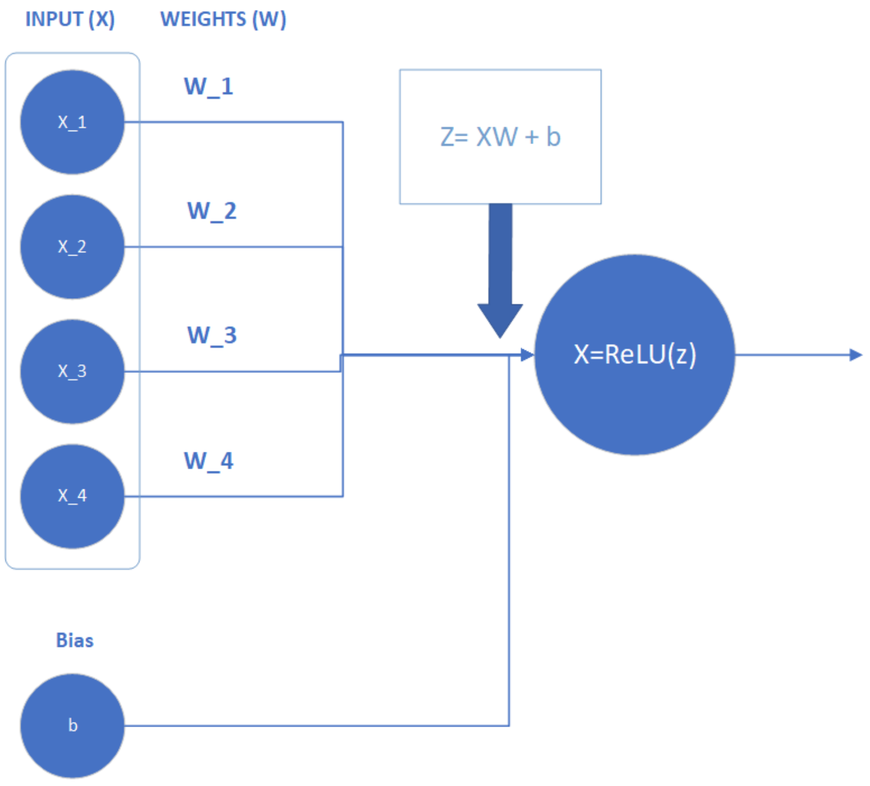

Figure B.1: A neuron which receives the output of a previous layer composed by 4 neurons. A ReLU activation function is applied over the dot product between the output of each neuron connected and the weight of the connection

Feed-Forward Networks are one of the simplest networks available. They are built by an input layer, a set of hidden layers (this set can contain just one layer), and an output layer. This model is based on feeding data forward to the input layer, passing through several hidden layers until reaching the output layer, all in one single direction. The most basic model of this type of architecture is the *Perceptron*[45], introduced in 1957. This model is a neural network with only one hidden layer that uses a set of neurons to predict an output

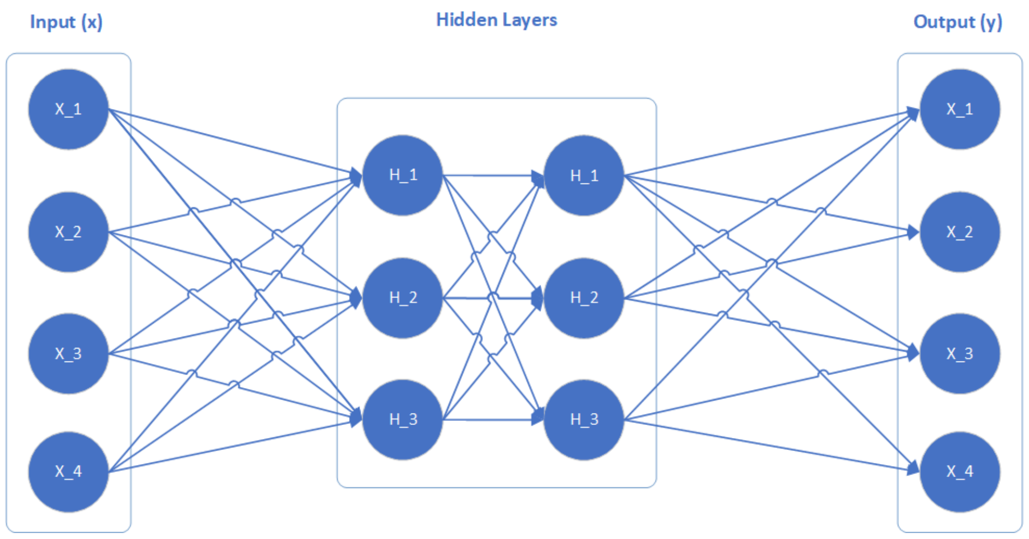

Figure B.2: A Feed-Forward Neural Network Architecture, with 11 neurons in total, 4 for the input layer, 3 for each hidden layer and 4 for the output layer.

B.1.1. Backpropagation Algorithm

Backpropagation is an algorithm that uses the gradients of the loss function to update the weights of a neural network in the direction of minimizing loss between the predicted and the actual output.

For a feedforward neural network with a loss function L, a set of inputs x, and a set of corresponding desired outputs y. The backpropagation algorithm consists mainly of three steps:

1. **Forward propagation**: Compute the predicted output ($\hat{y}$) given the input and current weights.

2. **Calculation of the error**: Calculate the error (loss) between the predicted output and the actual output.
$$L(\hat{y}, y)$$

3. **Backpropagation of the error**: Propagate the error back through the network by computing the gradient of the loss for each weight. This is done by applying the chain rule of differentiation:
$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial W}$$

   Where:
$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial L}{\partial \hat{y}}$$
$$\frac{\partial \hat{y}}{\partial z} = f'(z)$$

$$\frac{\partial z}{\partial W} = x$$

And:

$$z = Wx + b$$

$$\hat{y} = f(z)$$

Then, the new weight is computed as

$$W_{\text{new}} = W_{\text{old}} - lr\frac{\partial L}{\partial \hat{y}}$$

Where lr is the learning rate, which is a predefined parameter that represents the step size in updating the weight in the error-decreasing direction.

The weights are updated backward for each layer from the output layer to the input layer. The entire process is repeated until an error tolerance is achieved.

Backpropagation helps to find the optimal weights in a neural network. Without it, it would be a difficult and time-consuming task due to the complexity that a neural network entails. It makes it possible to train neural networks efficiently and effectively.

B.1.2. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of network architecture with a loop structure that allows predictions to be made by also considering information from the previous input. They are capable of retaining information during training which makes them useful for predicting sequential data. In their architecture, each layer also has connections to its previous layer creating a loop.

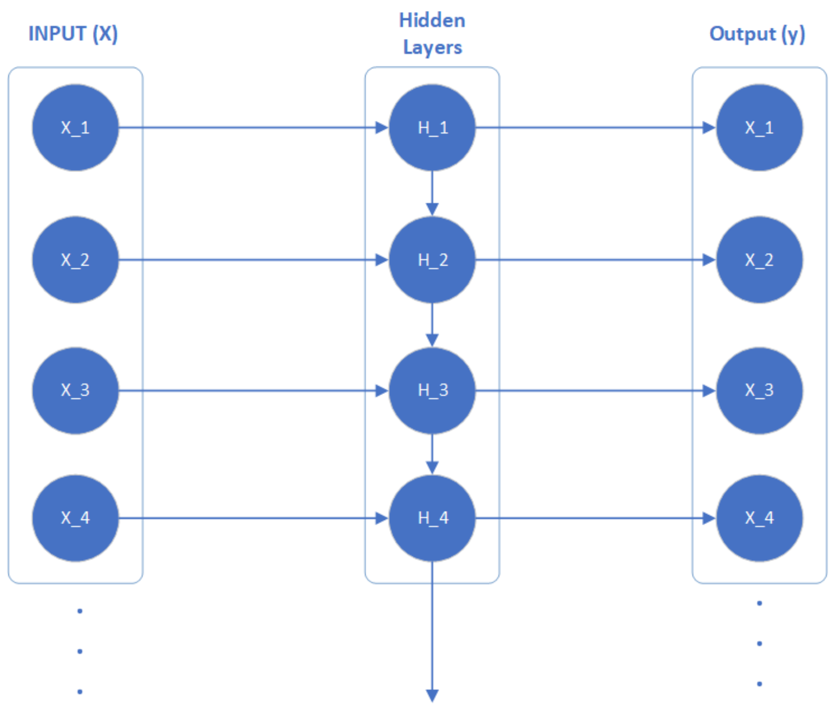

Figure B.3: An example of the architecture of a RNN. It has an input layer that sends the values to their respective hidden state in order the obtain the output values.

Suppose there is an input series: $x = (x_1, x_2, \ldots, x_T)$, where T is the number of the steps of the series and $y = (y_1, y_2, \ldots, y_T)$ the labels of the series. In an RNN, at each step t, the input x_t and the previous hidden state h_{t-1} are passed to the hidden state h_t as shown in B.3.

The value of the hidden state is calculated as:

$$h_t = f(W_h \cdot [h_{t-1}, x_t] + b_h)$$
$$y_t = g(W_y \cdot h_t + b_y)$$

Where W_h and b_h are the weights and biases associated with the hidden layer, while W_h and b_y are the weights and biases associated with the output layer. f and g are the activation functions of the hidden and output layer respectively.

Despite RNNs, being a large improvement from simpler architectures when dealing with sequential data, they can suffer from two problems called *vanishing gradient* or *gradient explosion*. They occur when the gradient values become too small or too large while being propagated across the network because they are multiplied at each step. For example in a time series with more than 200 steps, if the gradient values are less than 1, at step 200 the values from the first step will be very close to 0. On the other hand, values greater than 1 will have a large value, affecting the output. To solve these problems, Long Short-term Memory networks are a feasible approach.