TÉCNICO LISBOA

Open-domain Conversational Agent based on Pre-Trained Transformers for Human-Robot Interaction

Mariana Fidalgo Fernandes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Plínio Moreno Lopez
Prof. Jose Alberto Rosado dos Santos Vitor

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Plínio Moreno Lopez
Members of the Committee: Ricardo Daniel Santos Faro Marques Ribeiro
Fabio Nataneal Kepler

November 2021

Declaração/Declaration

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Throughout the development of this project I received a great deal of support and assistance.

I would like to thank Professor Plínio Lopez, my primary supervisor, for his guidance during this project. His insightful remarks encouraged me to improve my thoughts and elevate my work.

I would like to thank my parents, Francisco and Benilde for their sage and attentive advice. Everything I have and am, I owe it to you. I want to thank my brother, Miguel, for constantly pushing me to explore new things and strive to be the greatest version of myself. In addition, I would like to thank my uncles, aunts, cousins and grandparents for always believing in me, encouraging me, and acting as role models for me in numerous ways.

Furthermore, I could not have completed this dissertation without the support of my friends, who provided both interesting conversations and enjoyable distractions to take my mind away from my work. I want to thank you Helena, Barata, Adelaide, Calamar, Matilde, Bia, Jonas, Miguel, Proença and Xico. A special kudos to Carlos and Ricardo for helping me develop a tool to aid the evaluation. And lastly, thank you Ricardo, for your never ending support and for always being there for me.

Abstract

Over the past years, many breakthroughs occurred in the field of Machine Learning (ML) and Natural Language Processing (NLP), such as generative pre-trained transformers (GPTs), and attention mechanisms that learn contextual relationships between words in a text. These breakthroughs came with several new possibilities regarding Human-Robot Interactions (*e.g.* the creation of an open-domain chatbot). However, a substantial amount of research and available data are in English, causing low-resourced languages to be overlooked. This thesis explored this problem with two options: (i) Translation of the sentences before and after using the model fine-tuned on an English-based dataset, (ii) Translation of the English-based dataset to Portuguese and then fine-tune this model on it. When in presence of adequate training data and a good choice of generation method, it was demonstrated that DialoGPT (dialogue generative pre-trained transformer), a tunable neural conversational answer generation model, could learn the basic skills to conduct a dialogue. For the language models as well as the baseline methods, two sources of evaluation were used: (i) Metrics for text generation based on uncertainty (*i.e.* perplexity), and similarity between sentences (*i.e.* BLEU, METEOR and ROUGE) and (ii) Human-based evaluation of the sentences. Finally, it was shown that it is possible to resort to MT to have a fluent speaking chatbot, in portuguese. The translation of sentences before and after of the modified DialoGPT model, using the Daily Dialogue dataset led to the best results.

Keywords

Elderly companion, Natural Language Processing, Deep Learning, Machine Translation, Transformer, Attention Mechanisms

Resumo

Nos últimos anos muitos avanços ocorreram no campo de Aprendizagem Automática e do Processamento de Língua Natural, como Transformers generativos pré-treinados (GPT), e mecanismos de atenção que aprendem relações entre palavras num texto. Estas descobertas tornaram possíveis diversas abordagens para a Interação Homem-Robô (*e.g.* a criação de um chatbot de domínio aberto). No entanto, como a maioria desta pesquisa e conjuntos de dados estão em Inglês, as línguas com menos recursos são negligenciadas. Esta tese explora este problema com duas opções: (i) Tradução das frases antes e depois da sua utilização no modelo treinado num conjunto de dados em inglês, (ii) Tradução do conjunto de dados de inglês para português e depois fazer o treino do modelo no mesmo. Tendo dados de treino adequado e após uma inspeção sobre os métodos de geração, demonstra-se que o DialoGPT, um modelo neural de geração diálogo, consegue adquirir as capacidades básicas para conduzir um diálogo com nexo. Para os modelos linguísticos, bem como para os métodos de base, foram utilizadas dois métodos de avaliação: (i) Métricas para avaliar a geração de texto baseadas na incerteza (*i.e.* perplexidade), e semelhança entre frases (*i.e.* BLEU, METEOR e ROUGE) e (ii) Avaliação das frases com base humana. Finalmente, mostra-se que é possível recorrer à Tradução por Máquina para ter um chatbot fluente, em português. A tradução das frases antes e depois da sua utilização no modelo DialoGPT treinado, utilizando o conjunto de dados do Daily Dialogue, conduziu aos melhores resultados.

Palavras Chave

Companhia para Idosos, Processamento de Língua Natural, Aprendizagem Profunda, Tradução por Máquina, Transformer, Mecanismos de Atenção

Contents

List of Figures

List of Tables

Acronyms

NLP	Natural Language Processing
NLG	Natural Language Generation
NLU	Natural Language Understanding
HRI	Human-Robot Interaction
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NMT	Neural Machine Translation
NN	Neural Networks
LSA	Latent Semantic Analysis
NBC	Naïve Bayes Classifier
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
Seq2Seq	Sequence to Sequence
LM	Language Model
GPT-2	Generative Pre-trained Transformer 2
MT	Machine Translation
NMT	Neural Machine Translation

DialoGPT Large-scale Pretrained Response Generation Model

DialogRPT Dialog Ranking Pretrained Transformers

1

Introduction

Contents

"We human beings are social beings. We come into the world as the result of others' actions. We survive here in dependence on others. Whether we like it or not, there is hardly a moment of our lives when we do not benefit from others' activities. For this reason, it is hardly surprising that most of our happiness arises in the context of our relationships with others." - Dalai Lama XIV.

1.1 Motivation

The quote shows the importance of interpersonal relationships and connecting to others to add value to people's lives. These social interactions are fundamental for a functioning community and are necessary in order to allow humans to live fulfilling lives.

Loneliness leads to unhealthy lifestyles, with poorer physical and mental health, with age as a relevant factor [5]. As a consequence of the aging problem, specially in the most developed countries [6], some elderly people face this problem almost on a daily basis, with the prevalence of loneliness in elders being estimated to be 40% [7]. Some care homes already consider loneliness as an independent risk factor but, unlike other factors such as age, disability or memory problems, this one may be preventable [8]. The motivation for this thesis had this problem in consideration. The goal was to prove that it was possible to create an application that would behave as a listener, giving an appropriate response, causing an increase in affective interactions and a reduction of the sense of loneliness, similar to the one seen in [6] and [9].

Human-Robot Interaction (HRI) is a "field of study dedicated to understanding, designing and evaluating robotic systems for use by or with humans" [10]. It is currently a very extensive and diverse research field, with hundreds of publications each year and with activity by many different professionals. One area of HRI is social interaction, which includes robot devices to provide entertainment, teaching, comfort, and assistance for children and elderly, autistic, and handicapped persons [11].

Natural Language Processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence that employs computational techniques for the purpose of learning, understanding, and producing human language content. Natural Language Generation (NLG) is the subfield concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information [12]. Today's researchers refine and make use of these computational tools, creating spoken dialogue systems [13]. With the advances in Artificial Intelligence (AI), Machine Learning (ML) methods, the availability of very large amounts of linguistic data and with a much richer understanding of the structure of human language, more efficient algorithms were developed. Even though the results are not perfect yet, they are getting more and more accurate [13]. For example, ConveRT, a deep learning chatbot model, is able to dialog about a big variety of topics. It was trained using 727M Reddit utterances (input and response), and

shows an accuracy score of 71.8% [14].

1.2 The challenge

The main obstacle of creating a good textual generation system, is having a suitable quantity of labelled data. Nonetheless, with a large dataset, a dialogue system somewhat trained for every situation, could be built. Since the majority of the research in these areas is in English Language [15], it is already possible to obtain a good performance robot, to talk about various topics, like the example discussed above [14]. But what about the low resourced languages? If it was possible to obtain a big labelled dataset, like the ones existing in English, and convert it to the language in study (in a viable way), a robot with a good performance could be created.

During the research for this report, we concluded that machine translation always had some issues that resulted in poor translations. However, with the evolution of deep learning techniques that replace the ML statistical translations, the results significantly improved [16]. Neural Machine Translation (NMT), is a ”deep learning method that essentially involves the use of a broad scope of linguistic sources while looking at whole sentences instead of just words when translating.” [17]. This is the method used by OPUS MT [18]. They provide pre-trained neural translation models trained on OPUS data translation service. Despite being fallible, it gave accurate results for the great majority of translated sentences.

1.3 Problem Formulation & Objectives

The main goal of this project is to develop a chatbot, capable of pursuing conversations regarding the daily life. The topics would vary from: chit-chats about life and family, holidays and tourism, work and health, among others. This speech-based dialogue was supposed to be conducted in European Portuguese and should work in a reactive manner.

Considering the scarcity of Portuguese dialogues, to attend to the main goal, the approach was built on the grounds of text translation. Where, in a first approach a large, English, database was translated to the language in focus. And in the other approach only the input and output utterances were translated.

The objectives of this thesis are:

- Build a dialogue system that engages in a conversation about several matters, focusing on daily life topics;
- Prove the advantages of resorting to deep learning and transfer learning instead of using traditional statistical learning;
- Take advantage of Machine Translation (MT) in order to reach the first goal;

- Have human evalution to test the robot and the MT performance.

Due to the Covid pandemic that took place in 2020 and 2021, the performance of the robot in a nursery home could not be tested. The datasets used to train the models also could be more focused on the elderly, however this type of data is lacking, therefore more general datasets were used.

1.4 Thesis Document Outline

This thesis report is divided in 6 chapters. Chapter 1 gives an introduction to the problem and its importance, the main goal is presented and the organization of the document is shown. Chapter 2 gives an overview of the state-of-the-art, where past work is analysed along with some theoretical background, crucial to a better understanding. Chapter 3 comprises an explanation to most of the tools and technologies to build the solution. Chapter 4 provides an insight of what was developed for the thesis. Chapter 5 shows the obtained results together with the corresponding analysis and discussion. Lastly, Chapter 6 has some conclusions regarding all the developed work for this thesis as well as the possible future work.

2

State of the Art

Contents

In this chapter an overview of the evolution of Natural Language Processing will be conducted. A survey to the several models used to address the Dialogue Generation task will also be done. Finally, their advantages and disadvantages will be discussed.

2.1 Natural Language Processing

NLP is one of AI's sub-areas and it dates back to 1950, when Alan Turing publishes an article [19], proposing a method comprising the automated interpretation and generation of natural language. Alan tested the machine's ability to exhibit intelligent behaviour. This was later called the Turing test.

Prior to the introduction of ML techniques, the majority of language processing systems were built on hand-written rules (logical rules) based on expert knowledge [20]. Later on, research started to focus on statistical models that would make probabilistic decisions, using ML methods. This approach showed many advantages over the hand-produced rules system [21]. ML statistical methods have a major setback since they require feature engineering, therefore in the early 2010s the research shifted to the now popular Deep Learning (DL) methods [22]. This evolution can be better observed in Figure 2.1.

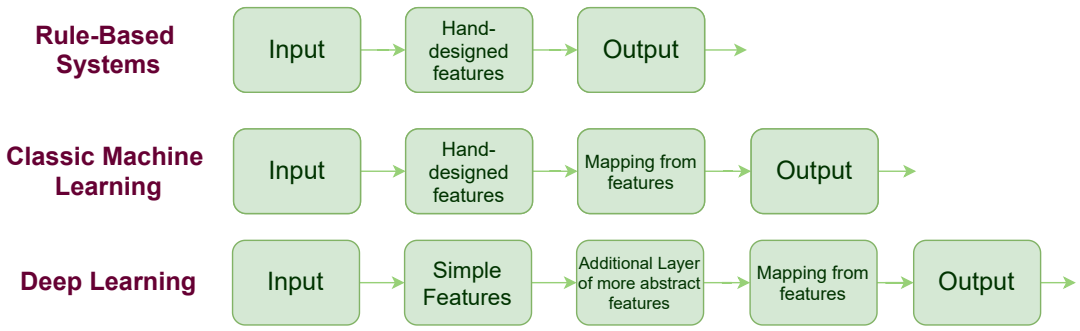

Figure 2.1: Natural Language Processing Evolution.

ML uses statistical learning methods, that work with a labelled dataset, which is described by a set of features or attributes. DL uses, likewise, a statistical learning method that instead, extracts the features or attributes from raw data. DL resorts to Neural Networks (NN), a considerable amount of hidden layers, a large amount of data and powerful computational resources. DL is a branch of ML that integrates algorithms to form a "artificial NN" capable of learning and making intelligent decisions on its own. Both ML and DL are able to handle huge dataset sizes. However, ML methods make more sense with small datasets. For example, if there are only 100 data points, decision trees, k-nearest neighbors, and other ML models will be much more valuable than trying to fit a deep NN on the data.

2.2 NLP subfields and applications

NLP encompasses the entire system, from data interpretation to decision-making. These decisions comprise of reading the information, breaking it down, understanding it and making judgments on how to respond. NLP, can be divided into two parts: the Natural Language Understanding (NLU) and the Natural Language Generation (NLG).

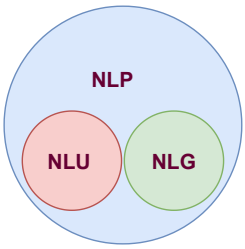

Figure 2.2: NLP, NLG and NLU.

NLU helps the machine to understand the data and its meaning so it can be processed accordingly. To find the objective behind the text it resorts to three linguistic levels [23]:

- **Syntax**: Refers to the grammatical structure of a sentence.
- **Semantic**: Understands the meaning of the text.
- **Pragmatic**: Understands the context to achieve the goal intent.

NLU will understand the flawed text and convert it into a machine-readable format. This subfield is used for semantics phrasing, semantic analysis, dialogues agents, among others.

NLG is known as the technique of creating natural language outputs from non-linguistic data inputs [24]. NLU focuses on computer reading comprehension, whereas NLG allows computers to write. Initially, for NLG, the systems worked on a ”fill the black space” methodology. However, with the application of Markov Chains, Recurrent Neural Networks, Long short-term memory and Transformers, computers evolved. This enabled a more dynamic text generation in real time. NLG is composed by three main stages [12]:

- **Text planning**: Where the text intent and content are formulated in a logical manner.
- **Sentence planning**: This stage takes care of punctuation and text flow. It divides the content among sentences and adds the appropriate pronouns and conjunctions.
- **Realization**: This stage considers grammatical accuracy.

Some of the most common NLP applications are: Dialogue Systems, Machine Translation, Summarization, Question Answering, Information-Extraction and Information Retrieval [25].

2.3 An High-Level NLP application: Conversational Robots

A chatbot is a software also referred to as virtual assistant. It is a form of artificial intelligence that is able to mimic human conversation [26]. The use of chatbots has become greatly popular in a large scale of applications. Tailored (*i.e.* customized) chatbots have been developed for multiple applications. However, these customized chatbots are limited to a specific type of conversations [27]. There are many ongoing studies, regarding the personification of these robotic systems, that show that humans tend to appreciate agents whom are enriched with conversational and social intelligence [28], [29]. Chatbots can be used in various fields such as Education (QuizBot [30]: a bot that helps students learn factual knowledge in science, safety, and English vocabulary), Virtual assistants (the well known *Siri* or *Alexa*), Health (*Sensely*, to whom patients can report their symptoms and receive service or self-care advice), among others.

The two major categories of conversational AI are task-oriented and open-domain. The goal of task-oriented bots is to help humans accomplish a specific task, through multi-turn dialogue. Knowledge regarding only one specific task facilitates the development of efficient and accurate chatbots. In this case, simple models can achieve good performance. Whereas open-domain bots aim to serve as a social companion to humans, with whom humans can have engaging and natural conversations. The latter is harder to create, because it needs to naturally engage in a conversation, and also to master several skills that are natural to humans, such as: comprehension, world knowledge, conversation history and constructing valid responses. Socialbots also need to be able to have adequate topical knowledge and perform smooth topic transitions [31].

The problem with assembling a chatbot with all these characteristics relies on the database used to train it, since the models already had a big evolution. Therefore, by knowing where the chatbot would be inserted, the database can be built to fit this environment, and as mentioned in Chapter 1, a suitable quantity of data is required to have a free-flowing dialogue.

2.4 Past work

The past iteration of this thesis project [32], that also had the Dialogue Systems application in focus, did not approach the problem in a DL manner. It made use of smaller databases and it was based on a NLP technique with newly collected data, intertwined with a ML classifier. To perform feature selection and extraction it resorted to Latent Semantic Analysis (LSA), since it is quite efficient in text categorization. The ML algorithm used was Naïve Bayes Classifier (NBC), since it has a simple implementation and works very well with the LSA method.

The results obtained were not faultless and the robot was only able to answer to the last human phrase uttered, completely missing the conversation context. Another setback was that the robot could

not generalize beyond the small training dataset.

Notwithstanding, with the advancement of the already existing methods, NBC has been surpassed [33]. Keeping in mind that LSA made the classifier a more feasible algorithm, it is still a technique that works better under small datasets. However, as mentioned in Chapter 1, in order to create a good performance chatbot a large dataset is needed. Therefore, NBC will not be the chosen approach to solve this project's problem.

2.5 Word Embeddings

It is important to understand how words can be represented as real-valued vectors in a predefined vector space, even before seeing how ML and DL are being used in NLP applications.

The simplest method for representing words with vectors is to count the occurrence of each word in the document. This method was called countvectorizer or one-hot encoding [34]. The encoded words become binary vectors where the position corresponding to the word is set to 1, and the others are set to 0. This creates high dimensional sparse vectors and does not take into account semantic and syntactic relationships.

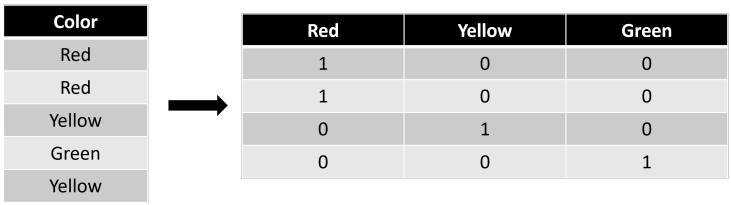

Figure 2.3: One hot encoding representation.

Word embedding tries to tackle these issues. This Language Modelling technique can map words to number vectors, where words are stored in a low dimensional vector space. This also uses distributional hypothesis which states that words that appear in similar contexts tend to have similar meanings [35]. Word embeddings can also be pre-trained on large amounts of data, enabling the model to only be fine-tuned on a smaller, more specific, dataset.

The most common model in word embeddings is Word2Vec [36]. It is a two layer Neural Network based on the distributional hypothesis. The word embedding vectors are learnt by an unsupervised model when fed with large amounts of text. The semantic similarity of words, along with other properties, are embedded in the representation. The vectors are created in a way that the distance between vectors, for words with close meanings, is smaller (e.g. "Portugal" and "portuguese" will be closer than "Portugal" and "banana"). The Figure bellow shows an example of how words with similar meaning can be mapped in a 3-D vector space. [35].

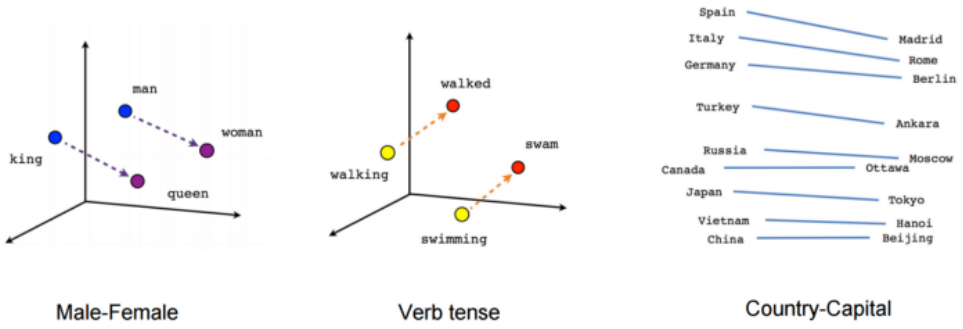

Figure 2.4: Source: Introduction to Word Embeddings and its Applications

While this model is able to capture semantic and syntatic relationships between words, they do not take the context into account. Meaning that the same word, even having different meanings, will be assigned the same vector. To address this, ELMo approach was proposed.

ELMo [37] is a deep contextualized word representation that cares for the syntactic and semantics of words along with how they are used in several linguistic contexts. It works through a bidirectional Long Short Term Memory (LSTM), explained in Section 2.6.4, pre-trained on a large corpus. ELMo's performance, shows a relative improvement of 6 - 20% in some NLP tasks when compared with others baselines. Its architecture can be seen in Figure 2.5.

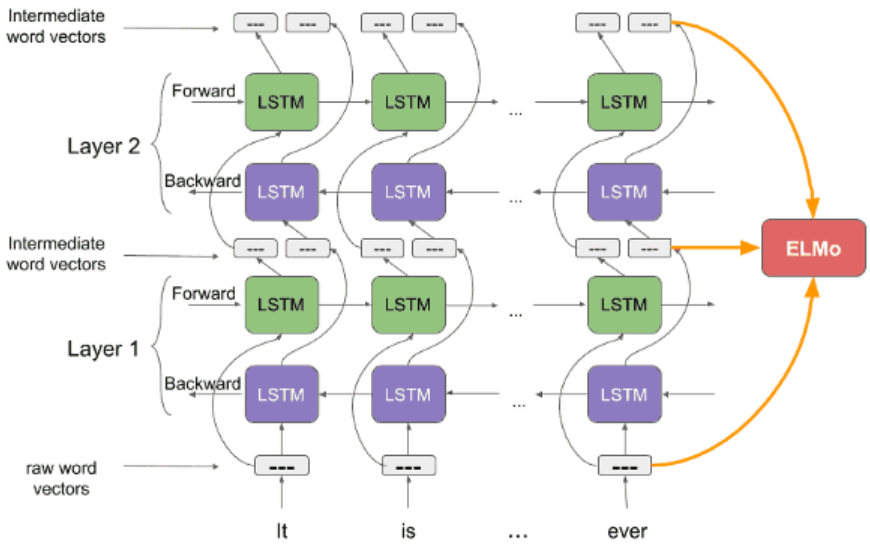

Figure 2.5: Source: NLP: Everything about Embeddings.

Finally, for the current state-of-the-art Transformer models (that will be explained later in this section), they do not use LSTMs like ELMo. Instead of processing words in a sequential manner, all words are processed in parallel, speeding the process and solving vanishing gradient problem [38]. This problem

arises when using gradient-based learning methods and backpropagation to train artificial NNs. In these methods, each of the NN's weights receives an update proportional to the partial derivative of the error function, with respect to the current weight in each iteration of training. The issue is that in some situations, the gradient will be so small that the weight will not be able to change. In the worst-case scenario, the NN's ability to learn might not be possible.

Transformers use attention mechanisms [39] to describe the connections and dependencies of each specific word, using all the other words in the sentence. For Transformers, the method responsible for preparing the inputs for a model is also know as tokenizer. The model text input, when encoded, goes through the following pipeline [40]:

- **Normalization:** It is the transformation applied to a raw string to make it less random. These operations involve stripping white space, removing accented characters or lower-casing all text.
- **Pre-Tokenization:** It is the act of splitting a text into smaller parts. An intuitive way to think is that this step will divide the text into words.
- **The Model:** Once the input is normalized and pre-tokenized, the model has the role of splitting the words into tokens, using the rules it has learned (since it is a pre-trained model). The tokens will be segments of those words (e.g. ”work”+”ing”). It will also map those tokens into their corresponding IDs in the model vocabulary.
- **Post-Processing:** This last step performs an additional transformation, making inputs suitable for the model in use, like adding potential special tokens.

2.6 The evolution of Natural Language Generation

Although we might be still far from robots that can think for themselves, ML and NLP have had a massive evolution over the past decades due to the usage of DL, Artificial and Recurrent Neural Networks, Sequence to Sequence (Seq2Seq) models and Transformers. The way we interact with technology and live our daily lives is being revolutionized by robots, where chatbots also play a part.

2.6.1 Markov Chains

Markov chains were one of the first algorithms to be used in language generation tasks. They were used to create suggestions for the next word in a phrase in older versions of smartphone keyboards. This model predicts the next word in the sentence by resorting to the current word. It considers the relationship between each unique word, in order to calculate the likelihood of the following word in the phrase [41].

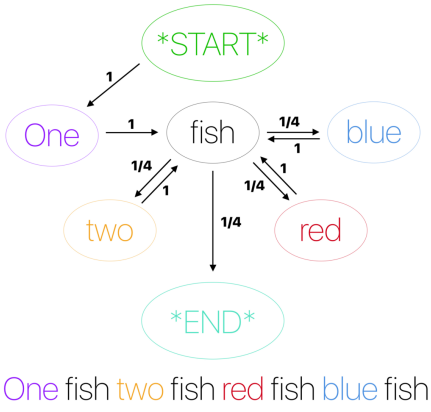

Figure 2.6: Source: From "What is a Markov Model" to "Here is how Markov Models Work".

Markov chains, since they only focus on the current word, loses all the context and structure of the preceding words. This leads to incorrect predictions making it an undesirable approach to solve our problem.

2.6.2 Introduction to Deep Learning

Deep Learning algorithms are one of the most popular types of ML algorithms, which learn from a layered model of inputs, called NNs. The "deep" in deep learning is referring to the depth of layers in a NN. In DL, the learning of a current layer depends on the previous layer input. One advantage of this type of approach is that, even without understanding semantic or syntactic structure of the language in use, the models have a better performance on very large datasets, when compared to more traditional approaches [42]. As mentioned in Chapter 1, making use of MT, it could be possible to obtain a database large enough to use these algorithms to achieve the desired goal.

2.6.3 Artificial Neural Networks

Identically to the human brain, which has the neuron as its elementary unit, neural networks are composed of perceptrons. These were originally proposed by Frank Rosenblatt in 1957 [43].

Biological neurons communicate through synapse impulses which have both chemical and electrical reactions. This impulse travels along the entire neuron and are transmitted to other connected cells. The perceptron, artificial neuron, receives a vector of inputs, also known as features. The dot product between this input and a weight vector, along with a bias value, that allows a shift on the activation function (allowing a better fit to the given data) are processed in a non-linear activation function. The activation function enables this model to deal with more complex data and produces a binary output, 1

if the dot product is above a certain threshold or 0 otherwise. The aforementioned perceptron [44] is described in Figure 2.7,

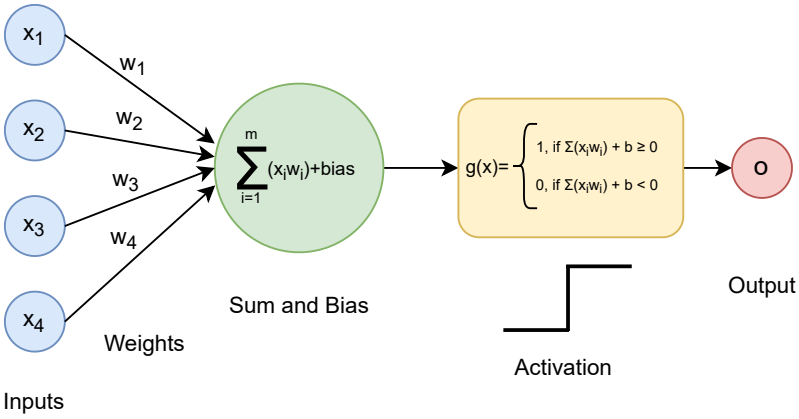

Figure 2.7: Perceptron.

In the Figure above, *o* is the output, *x* and *w* are the input data and the weight vectors respectively, *b* is the bias value and *g* is the activation function. For the activation function, there are some variants that can be use, e.g. ReLU linear function, that outputs the input directly if it is positive, otherwise, it will output 0. The weight vector, *w*, in the perceptron, starts with random values and it is updated during the training process, by receiving an observation and comparing it with the target value.

$$w_i^{new} = w_i^{old} + \eta(t - o)x_i, \tag{2.1}$$

The update can be formally described in the Equation 2.1, where *t* is the target value, *o* is the output computed by the perceptron and η is the learning rate, that quantifies how much the weights update in the course of the training process. This process is repeated several times, each time known as an epoch, for the whole dataset. It finishes when one of two possibilities is encountered: either all the observations were correctly predicted or a predefined number of iterations has been reached. The perceptron will converge only if the dataset is linearly separable by a hyperplane.

In order to solve this limitation, and to enhance the artificial brain model, a new model was proposed, called Multi-Layer Perceptron (MLP). MLPs are fully-connected feed-forward networks, whose key idea is to add intermediate layers of artificial neurons before the final output layer. Each hidden layer computes some representation of the input and propagates it forward.

In the case of one hidden layer, the model can be described by the Equation 2.2, where g_o and g_h are the activation functions that correspond to the output and hidden layers, respectively. W_h, W_o and b_h, b_o represent the weight matrix and the bias vector for each mentioned layer.

$$f(x) = o = g_o(g_h(x \cdot W_h + b_h) \cdot W_o + b_o), \tag{2.2}$$

In Figure 2.8, a Perceptron and MLP comparison can be observed.

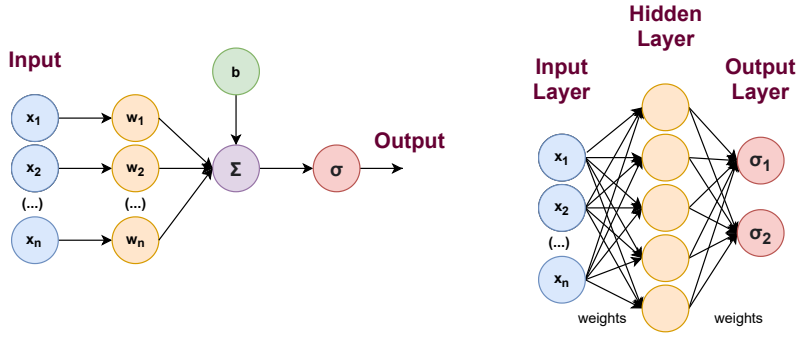

Figure 2.8: Perceptron vs MLP.

For the MLP model, the training phase begins with the forward propagation (feed-forward), where neurons compute the output predictions. Then the model attempts to minimize the error of a loss function (L), chosen accordingly to the problem and, in order to find the parameters and the weights that produce the minimum error, the backpropagation algorithm [45] is used. When the algorithm is performing the forward feed, an output is obtain, which will be compared do the true labels in order to get the error. The error is propagated through the network, with the goal of updating the weights of each node. This computation provides the gradient and describes the importance that each neuron has on the initial input. Model training comprises the gradient descent method, which is an optimization algorithm that has the goal of finding the function's minimum. This optimization algorithm is represented by Equation 2.3.

$$W^* = arg \quad min(L(w)), \tag{2.3}$$

$$w_{new} = w_{old} - \eta \times \Delta L(w), \tag{2.4}$$

The algorithm has a weight's update formula, that is represented in Equation 2.4, with η as the learning rate and *L* as the loss function.

2.6.4 Recurrent Neural Network

A Recurrent Neural Network (RNN) is an artificial NN where the output of a layer is fed into another artificial NN, as an input. In other words, a RNN has the ability to remember the previous iterations

and use its information in current calculations. It differs from the previously mentioned Markov Chains because it considers more than just the previous word. This characteristic makes this method more suitable for chatbots, since understanding the context in a conversation is crucial in order to deduce the intent and build an appropriate answer. [46]. Furthermore, this sequential network shares parameters along the NN, allowing the model to have a generalization capability, which is useful for words that occur many times in a sentence. In Figure 2.9 the model representation can be observed.

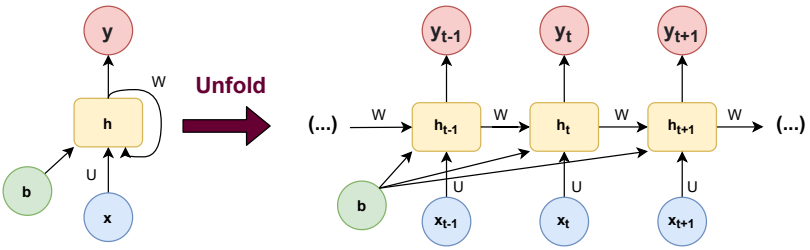

Figure 2.9: RNN and its unfolded structure.

Formally [47], in the first step, the RNN receives an input sequence vector x_t, and a hidden state vector h_{t-1} to produce a new state vector h_t.

$$h_t = g(U \cdot x_t + W \cdot h_{t-1} + b), \tag{2.5}$$

At each step *t*, the hidden state is updated, according to Equation 2.5, where *W* and *U* are a weight matrices, *g* is a non-linear activation function and *b* the bias vector.

In the second step, the output y_t is obtained by applying an activation function, *o*, to the hidden state, as can be seen in Equation 2.6

$$y_t = o(h_t) \tag{2.6}$$

The input and output vectors can be of different sizes and RNNs can have different architectures. It can be inferred, by looking at the expressions above, that h_t and y_t are dependent of all previous inputs. Resorting to backpropagation through time, the error corresponding to a certain loss function is backpropagated, like in the feed-forward network case. However this method applied to long sentences, is computationally expensive. They cannot store words encountered far back in the sentence, and will only make predictions based on recent words. It can also lead to vanishing/exploding gradients. Recalling, the latter occurs when the gradient that carries the information decreases significantly, making it impossible to keep training the NN.

Several approaches have been proposed, and LSTM was one approach that succeeded in tackling this issue. LSTMs comprise a four interacting layer NN instead of a single layer network for RNNs. They introduce a memory cell and three gate units to control the flow of information in the RNN. The memory

cell preserves the error gradients and allows the carry out of the information, whereas the gate units are responsible for the addition and removal of information from the cell and what is the final output of the LSTM unit. In Figure 2.10 the LSTM cell architecture can be observed.

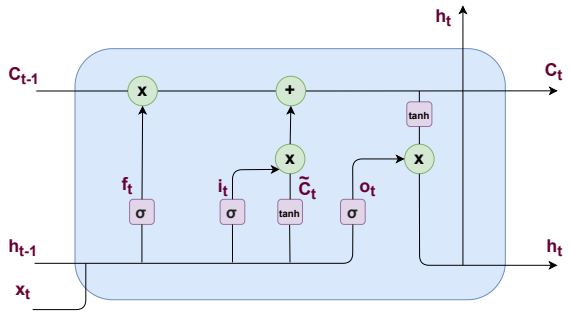

Figure 2.10: LSTM cell architecture.

Considering the sentence "I am from Portugal. I am fluent in _____". To predict the next word, "portuguese", the model will focus on the word "Portugal" from the past sentence, that was "saved" in the memory cell. The cell saves this information while processing the sequence and uses it to forecast the following word. When the forget gate hits a full stop, it understands that the meaning of the phrase may have changed, allowing the current cell state information to be discarded. This allows the network to keep up with relevant information and minimizes the vanishing gradients. The model is able to handle longer sentences when compared to the previous mentioned models.

A single LSTM [48] module comprises a memory cell *C* and three gates (*input gate, output gate, forget gate*), formally described in Equations 2.7, 2.8, 2.9, respectively.

$$f_t = \sigma(W^f \cdot x_t + U^f \cdot h_{t-1} + b^f) \tag{2.7}$$

$$i_t = \sigma(W^i \cdot x_t + U^i \cdot h_{t-1} + b^i) \tag{2.8}$$

$$o_t = \sigma(W^o \cdot x_t + U^o \cdot h_{t-1} + b^o) \tag{2.9}$$

Looking again at the Figure 2.10, initially, a cell receives as input the hidden layer vector h_{t-1} and the current input vector x_t and making use of the *forget gate* (which is a sigmoid layer), it returns a number between 0 (values to be removed) and 1 (values to be kept), scaling the impact on the cell's state. A candidate input vector $\widetilde{C_t}$ is also generated with the above mentioned vectors, it holds possible values to add to the cell state. The candidate input vector can be formally described by Equation 2.10

$$\widetilde{C_t} = \tanh\left(W^c \cdot x_t + U^c \cdot h_{t-1} + b^c\right) \tag{2.10}$$

Afterwards, the former state C_{t-1} is multiplied with the forget gate output value f_t, and summed with the previous values, originating the new cell state value C_t, which has the purpose of lowering the risk of vanishing/exploding gradient. This computation is represented in Equation 2.11 .

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t} \tag{2.11}$$

Finally, as shown in Equation 2.12 the cell state is normalized by a hyperbolic tangent function, and then multiplied by the *output gate*, computing the h_t which is passed onto the next cell.

$$h_t = o_t \cdot \tanh C_t \tag{2.12}$$

LSTM were proven to be very useful when it comes to chatbots, since they have the ability of refer to a piece of distant information in time [49]. Even though LSTMs and its variations seemed to be the answer, they have still some limitations. There is still a complex path from the past cells to the current one, limiting the length of sequences. An additional ”pain point” is that LSTMs are very difficult to train due to the high computational demand. They are also difficult to parallelize on account of their sequential structure, inhibiting the ability to take advantage of computing devices like GPUs or even TPUs.

2.6.5 Sequence to sequence

Sequence to Sequence (Seq2Seq) model [50] is based on the RNN architecture, consisting of two RNN. One is an encoder that takes a sequence (sentence) as input and processes one symbol (word) at each time step. Its goal is to convert a sequence of symbols into a vector with a fixed size that will encode only the important information, losing the unnecessary parts. Each hidden state will influence the next hidden state, being the final one the summary of the sequence. The last hidden state vector can also be called context. From this vector, the decoder generates another sequence, one symbol (word) at a time. In each time step the decoder is influenced by the context and by the previous generated symbols [47]. The encoder and decoder allow the model to be fed with input sentences of different lengths. This model was the industry best practice for response generation, in 2019, and it is still widely used [46]. Both encoder and decoder resort to LSTM units (represented by the blue and red rectangles). This architecture can be seen in Figure 2.11.

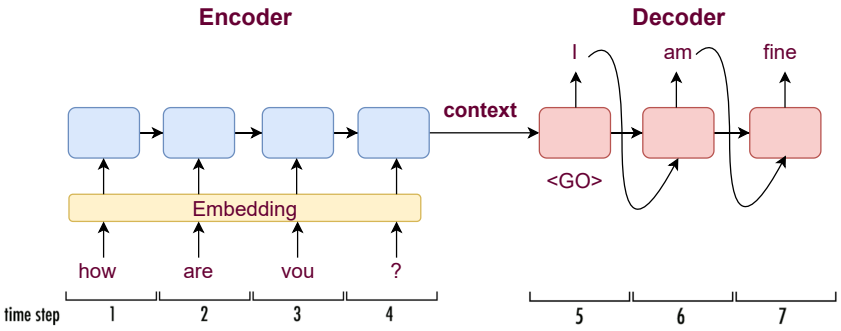

Figure 2.11: Sequence to sequence basic structure.

The goal of this model, is to map an input, $x = x_1, x_2, ...x_n$, with a fixed length sentence, along with a context vector, $h = h_1, h_2, ...h_n$, created by the encoder, into an output, $y = y_1, y_2, ...y_n$ generated by the decoder. The output is also a fixed length sentence (these sentences can have different lengths between themselves). This is an auto-regressive model, since the decoder uses the previous steps' output as additional input, when computing the current output. The main limitation of this model is that all the information in the input sentence should be encoded into a fixed length vector, the context. However, as the length of the sequence increases, the amount of information lost gets bigger too, leading to a poor performance when handling long sentences.

Attention mechanisms [39] came to solve this issue. When they are used, the encoder provides the decoder with all its hidden states, rather that just the last one with the intent prediction, allowing the decoder to notice all the important parts of the input at each timestep. The decoder, has therefore an additional input at each timestep, which is computed by taking a weighted sum over the encoder hidden states, creating the context vector for that timestep. The decoder, as mentioned, has as inputs the start of a sentence and an initial hidden state vector and it generates an output and a new hidden state. The attention process is performed, from where it results the context vector, which is concatenated with the decoder hidden state vector. This concatenation is inputed in a feedforward network and a word is generated, that will serve as input in the next time step, along with the decoder hidden state [51]. Attention mechanisms provided big improvements in ML [52].

2.6.6 The Transformer

Even though the encoder-decoder architecture, using LSTMs with attention mechanisms, was able to achieve satisfactory results, it is still necessary the state h_{t-1} to compute h_t, therefore it is not possible to use parallel computing techniques speeding up the training process. To solve this limitation, the Transformer architecture was proposed in 2017 [1]. It is based on an encoder-decoder model, to address tasks like automatic translation, conversational chatbots, among others. It leaves recurrence aside and relies on self-attention mechanisms, where the model uses one sequence of symbols enabling it to focus

on different words of the sentence and understand its structure. Transformers have been outperforming the previous described models in the mentioned tasks and its architecture can be analysed in Figure 2.12.

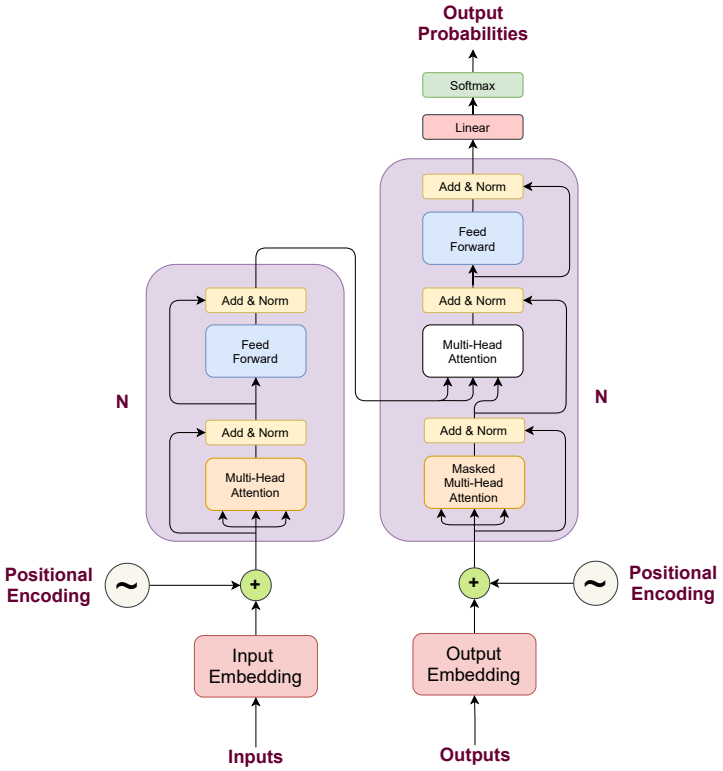

Figure 2.12: The Transformer architecture [1].

The model is constituted by two sub-networks [51]. The encoder, on the left side of Figure 2.12, with N identical layers, takes the word embedding of the source as input and computes a representation of it. Afterwards, the decoder network, on the right, with also N identical layers, takes as input this computation and the embedding of the past generated tokens from the output sentence (if it is not the beginning of the sentence). With this information, the decoder creates a new output sentence one word at a time.

The role of the encoder is to generate the context vector from the input sentence. The context vector, has the same purpose as in the Sequence to Sequence model. It is of the same size as the input vector and it will have seen all the tokens on every position of the input sentence, differing from standard RNN.

Initially the inputs and outputs are embedded in an n-dimensional space and are added to a positional encoding layer, since strings cannot be use directly. Due to not having any convolution or recurrence, information about the positions must be added. The positional encoding vectors are not learned, they are created using the sine and cosine functions. These functions are specified in Equations 2.13, 2.14, where *pos* is the position, *i* is the dimension of the input embedding and d_{model} is the size of the model's

encoding.

$$PE_{pos,2i} = \sin \frac{pos}{10000^{2i/d_{model}}} \tag{2.13}$$

$$PE_{pos,2i+1} = \cos \frac{pos}{10000^{2i/d_{model}}} \tag{2.14}$$

Each dimension is a sinusoid where the wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. These computations can be understood like binary encodings for numbers, where the least significant bit changes every number and the second least changes every two numbers, allowing the generations of unique encodings for every time-step. Distances between two words are independent of the sentence length and the model is able to handle longer sentences than those trained. This computation is then passed through the encoder layers, that will map all the input sequence into a abstract continuous representation. The encoder's first layer is the Multi-Head Attention which is represented in Figure 2.13, and it will be analysed further. It calculates the similarity score between each word and the rest of the words present in the input sentence.

The decoder's role is to take the encoded representations of the source sentence and convert it into predicted tokens. It then compares them with the target sentence and calculates the loss that will be used for further training. Firstly, similarly to the first step of the encoder, the target sentence tokens are passed through an embedding layer and summed with a positional encoding layer. This step's output is passed through a Masked Multi-Head Attention, whose goal is to avoid the current token to see the future words compromising the attention scores. In the last step, the softmax function normalizes the scores and the highest scoring word is chosen for that timestep.

An in-depth analysis will be done to the Multi-Head Attention Layer, whose architecture is shown in Figure 2.13.

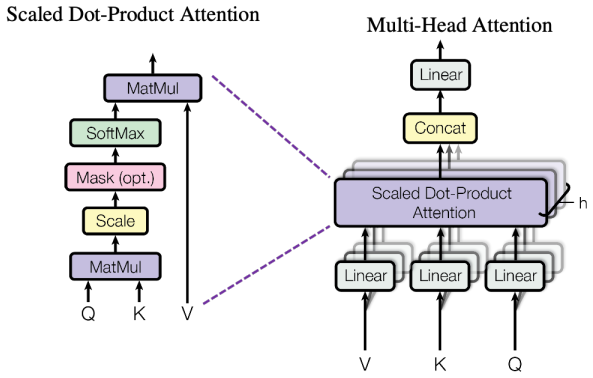

Figure 2.13: Multi-Head Attention Layer architecture [1].

Focusing on the left side of the Figure, the self-attention algorithm takes as input Q, which is a matrix that contains the query. This query is a vector of one word of the sequence. Takes K that are all the

keys, meaning they are vectors of all the words in the sequence and V that are the values, which are again the vectors of all words in the sequence. V is the same word sequence as Q for the encoder and decoder, multi-head attention modules (represented in orange in Figure 2.12). V differs from Q in the attention module, that considers both the encoder and decoder sequences. The output computation is given by Equation 2.15.

$$y = Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.15}$$

This computation ensures that the words we want to focus are kept and the irrelevant words are dropped. The weights, $\frac{QK^T}{\sqrt{d_k}}$, are determined by how each word in the sequence (represented by Q) is impacted by the other words (represented by K). This function is also scaled by the square root of the depth, avoiding exploding and vanishing gradients. The SoftMax function is applied to the weights, allowing the values to have a distribution between 0 and 1. The result generated by the SoftMax function is applied to all the words in the sequence that are introduced in V.

The right-side of Figure 2.13 shows that the attention mechanism can be parallelized. It is repeated multiple times with the linear projections of Q, K, V, enabling the learning of a variety of linear representations (various contexts at the same time), which are valuable to the model. These linear representations are achieved by multiplying Q, K, and V by weight matrices W learnt during training.

As already stated, Q, K and V matrices' values differ if the attention model is in the Encoder, Decoder or between both. This variation can be explained since the focus of the encoder is the entire input, the decoder focuses only on a portion of the input and the multi-head attention module that joins the encoder and decoder, needs to ensure that the encoder input is considered alongside the decoder input up to a certain position.

Following this last multi-head attention a normalization layer is employed, generating the Feed Forward layer's input. The Feed Forward layer is a fully connected network that increases the non-linearity and changes the data representation allowing it to have a better generalization and dropout. It consists of two linear transformations with a ReLU activation in between, shown in Equation 2.16.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2, \tag{2.16}$$

Transformers for Language Model (LM)

LMs are models that can predict the next word based on a portion of an utterance. For conversational tasks, the model Large-scale Pretrained Response Generation Model (DialoGPT) [2] was built by Microsoft researchers. This model was pre-trained using multi-turn dialogues extracted from Reddit discussion threads. It is based on the *Open AI*'s Generative Pre-trained Transformer 2 (GPT-2) [53] architecture, adopting the generic transformer LM [1] and inheriting from GPT-2 a 12-48 layer transformer, with normalization layers, an initialization scheme, and a byte pair encoder for the tokenizer.

The original Transformer [1] was made up of encoder and decoder blocks also known as transformer blocks. This type of architecture made sense since the model addressed a MT problem, where encoder-decoder architectures achieved good results in the past. In later studies, the architecture was stripped of either the encoder or decoder blocks [54] to adapt to new tasks.

A new method to perform Language Modelling was proposed, where the Transformer-Encoder blocks were thrown away [54], keeping just the Transformer-Decoder blocks . These latter, that can be seen in Figure 2.14, were identical to the earlier mentioned decoder blocks, apart from the fact that they did not have the second layer of self-attention.

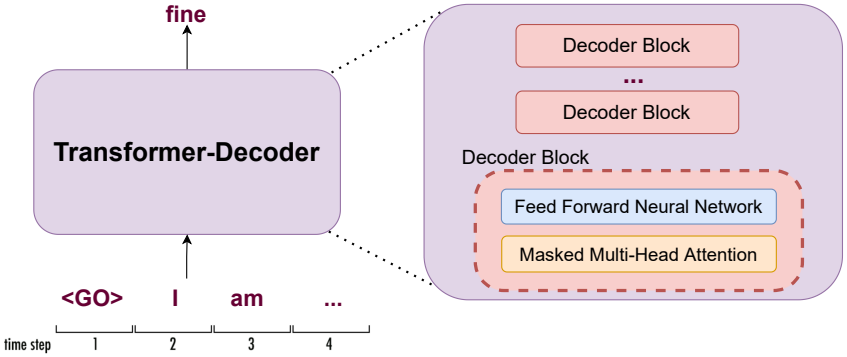

Figure 2.14: Transformer-Decoder architecture.

The GPT-2 model [53], similar to traditional LMs, outputs one token at a time. The model is built using only the mentioned decoder blocks. It is auto-regressive, since that after a token is predicted it is added to the input which will be fed again to the model. In a typical self-attention block, at one position, the model can peak at tokens to its right. However, this is avoided by masked self-attention layers which are used by the GPT-2 model and the DialoGPT.

The fact that it is pre-trained, can provide the capability of generalization, since it has seen a big amount of data. The fact that it is an auto-regressive model, can enable it to handle dependencies and relations between words. The big setback for this model is that it is only possible to use it pre-trained in English language. Even though it is possible to use it in other languages, the training needs to be done from scratch by the user, which is slow and computationally expensive.

As mentioned before, transformers have been outperforming the other architectures. However they have some drawbacks like the large amount of data needed to train and the large amount of memory needed. Despite the drawbacks, this approach will be used to tackle the desired goals of this thesis project.

An illustrative comparison was going to be conducted. However, the first drawback encountered was that most of the comparisons performed to the various models are done in the field of MT. Another drawback for the construction of this table was that it was not found a paper or article that compared

quantitatively these models, with the same dialogue based dataset. As it can be seen in the table bellow, even though the model DialoGPT is far superior to the RNN model, it has a worse BLEU-2 [55] score. Therefore, it is not relevant to take conclusions with tables like these.

Model	Size	BLEU-2
DialoGPT-small	117M	10.36%
Seq2seq	15K	26.8%

Table 2.1: Dialogue Generation model comparison: DialoGPT-small pre-trained [2] and Seq2seq [3] trained on Daily Dialogue dataset [4].

As seen, unlike previous models, transformers can use representations of all words in context without the need to compress all the information into a single representation. This architecture allows them to retain information across longer sentences without significantly increasing the computation requirements.

2.7 Machine Translation

MT is one of the most active subfields in NLP, with NMT achieving incredible results when a large amount of data is available. However, for low-resource language-pairs, it still remains sub-optimal. This is due to the unavailability of large parallel data, meaning it lacks sentences placed alongside its translations. The advances in ML along with the implementation of NMT techniques impacted significantly the automated translation field [56].

OPUS-MT models [18] are trained on state-of-the-art transformer-based NMT. Marian-NMT, which is a stable toolbox with efficient training and decoding capabilities [57], is applied to the framework. The models are trained on available open source parallel data. Similar to the LMs mentioned before, this system also uses an encoder-decoder architecture with attention mechanisms. They tested this model's accuracy with common benchmarks and test sets, allowing the monitoring of the quality of the NMT models.

For some languages, it incorporated back-translation, which consists in translating the target text back to the original language. It allowed the comparison between translations with the original text for quality and accuracy [58]. This way, more abundant monolingual text data could be used on the web for training OPUS-MT's models. This technique, generally also allows the model output to be more fluent, which is something where low-resourced translation models under-perform [59].

Taking in consideration this and the fact that for every translated sentence, its message is communicated correctly in almost all cases, as seen in Chapter 1, we can use the OPUS-MT Models translation tool to aid the development of this project.

3

The Approach

Contents

In this chapter some important concepts, like Transfer Learning, will be presented, along with an overview to the model and libraries that was used to address our problem, and finally the sets of data that were the key to fine-tune the model.

3.1 Transfer Learning

Humans have an intrinsic ability to transfer knowledge across tasks. The knowledge acquired while learning one task can be used to solve other related task. The more related, the simpler it is to re-utilize knowledge. Traditionally, ML and DL algorithms were trained after random parameters or weights initialization. They were design to work isolated, since they were trained to solve specific problems. Once the feature space changed, the models would have to be developed from scratch. Transfer Learning came to fight this dogma and prove that is possible for an already pre-trained model to be applied to a different but related problem [60]. This process avoids having to create a network's architecture from scratch and train it during a significant amount of time. In this technique, a NN is fine-tuned to a specific problem after being trained on a general problem. It allows DL models to converge faster and with less requirements [61].

Transfer Learning was one of the foundations for this thesis project. Pre-trained language models were fine-tuned to address the chatbot problem. The LMs will be described in the upcoming section.

3.2 The Model

There is no doubt that using Transfer Learning brings many advantages, therefore, pre-trained models were the foundation to attend to this thesis problem.

As seen in chapter 2, GPT-2 works well across a big variety of tasks, mainly due to another innovation known as fine-tuning. This is one of the most used approaches for Transfer Learning, when working with DL models [62]. It consists in starting with a pre-trained model on the source task and train it further on the desired task. When compared to the approach of training a model from scratch, it undoubtedly improved the performance. When no fine-tuning is used, the idea is to try to optimize all parameters of the DL model, using the target training data (after initializing the parameters with the pre-trained weights). However, since the target dataset is small (compared to the one that was used to train the model) and since the number of different parameters is big, if the whole network is fine-tuned to that small dataset, overfitting will occur [63]. The solution to this setback can be to only fine-tune the deep network's final layers, whilst the parameters of the remaining early layers are frozen at their pre-trained values. This solution can be supported due to a combination of both the insufficient training data for the target task and the credible evidence that early layers learn low-level characteristics [64].

The pre-trained model that was used is fine-tuned from Open AI's GPT-2 [53] and is called DialoGPT [2]. GPT-2 is a unidirectional transformer trained on a 40GB corpus, using language modeling. It had around 1.5 billions parameters, and it was trained with the goal of predicting the next word given all the precedent words in a text. Since it was trained on dataset with a lot of topic diversity and across many domains it is able to generalize [53]. The bittersweet part is that it has no humans labelling the data in any way. However, this enables it it to use lots of publicly available data with an automatic process to generate inputs and labels from those texts.

DialoGPT is a "tunable neural conversational response generation model" [2]. Consists on a GPT-2 model trained on 147M conversation-like data extracted from Reddit comment chains for a 12 year period (2005-2017). Three main models were released with a different number of parameters: small (117M), medium (345M) and large (762M). There were also numerous pre-processing steps done to the input and output samples, before feeding them to the model, as a precaution. Such as:

- Remove sentences where either the source or the target has an URL;
- Remove phrases in which the target comprises repetitions of at least 3 words;
- Remove sentences where there is a trace of toxic language;
- Remove sentences where the response did not contain any word from the top-50 most common word list in the English Language;
- Remove sentences where response contains special markups like "[" or "]";
- Remove sentences where the length of source and target is above 200 words.

The name DialoGPT is under the assumption that, with these rules, this model would capture the joint distribution of P(Target, Source) in a dialogue. Its purpose is to mimic the human performance in a single-turn conversation. This model is suitable for creating a virtual companion for an engaging conversation. The small and medium pre-trained implementations were tested. As it will be seen further down, due to having limited resources, the smaller model is preferable and it was able to maintain a coherent dialogue.

3.3 HuggingFace Transformers

As seen in the previous chapters, research in model architecture as well as in model pre-training have fueled advancements in NLP, leading to better results in a wide variaty of tasks.

The *HuggingFace Transformers* [65] is an open-source library, maintained by a team of engineers and researchers, whose objective is to make these breakthroughs available to the rest of the machine-learning community. The library is composed by meticulously built state-of-the-art Transformer architectures that are all controlled by a single API which is available at this url.

This library takes care of the distribution and usage of a wide-variety of pre-trained models in a centralized model hub, allowing users to compare various models using the same API. *HuggingFace Transformers* include tools to ease the training and development of models. Every model in the collection can be defined by three main parts:

- **A tokenizer**: Responsible for converting raw text into encodings, as aforementioned in Chapter 2. This class can be instantiated from a corresponding pre-tained model or can be configured manually. It comprises the vocabulary token-to-index for the corresponding model and handles the encoding and decoding of inputs based on the model's tokenization method. These classes are accessible at this url.
- **A transformer**: Responsible for taking the encodings and turning them into contextual embeddings, assigning to each word a representation based on its context; With the same API, different model architectures can be used, empowering users to easily transition between those models. This is also thanks to the use of Auto classes, which are instantiated automatically using the parameters given by the pre-trained model, picked by the user.
- **A head**: Responsible for making prediction, for a specific task, based on the contextual embeddings. On top of the Transformer's contextual embeddings, they add an output layer and an optional loss function. (They can be used for fine-tuning or pre-training.)

As mentioned, *HuggingFace Transformers* has the purpose to facilitate the use and distribution of pre-trained models. It now contains thousands of pre-trained and fine-tuned models. Every model has a model card describing its properties, architecture and use cases. Models are both available in PyTorch and Tensorflow and this library makes it extremely easy to use them. Recurring only to a couple of lines of code, the DialoGPT-small model, can be uploaded and used, as it can be seen in Appendix A. The Language Modeling use-case (more specifically text generation) was addressed, using *HuggingFace Transformers* open-source library.

3.3.1 Generation and decoding methods

The objective of text generation is to produce a comprehensible segment of text that follows from the provided context. As can be seen in the code listing, present in Appendix A, the pipeline (model) object invokes the function *generate()* from the *HuggingFace Transformers*, that was created to generate text. DialoGPT generates predictions based on the whole conversation history, which is concatenated before it is fed into the model. This is then utilized to generate a response, but only using the preset number of most recent tokens in the input sequence.

Besides having a good dataset and a good model architecture there is also a need of having good decoding methods. A brief overview to some of the most prominent methods will be conducted.

Starting with greedy search, it simply selects the word with the highest probability as its next word, at each step [66]. This method is illustrated in Figure 3.1. Even though the generated words tend to follow the context, the model will quickly start to repeat itself. Another disadvantage is that it misses words with a high probability that are hidden by low probability words.

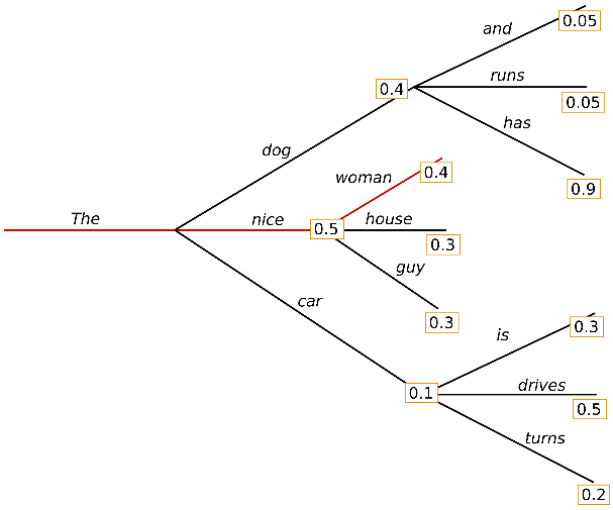

Figure 3.1: Greedy Search. Source: How to generate text

To tackle the latter mention disadvantage, Beam Search comes to the picture. It avoids missing high probability word sequences by storing a number of beams or hypotheses at each time step, ending up choosing the hypothesis carrying the highest probability. Figure 3.2 shows this method with 2 beams. The result can be more fluent, but it will have one of Greedy Search's problems: still being quite repetitive, which is a frequent issue in language generation [67]. Penalties for n-grams (sequences of n words) can be a straightforward solution [68]. The n-grams penalty ensures that no n-gram occurs twice by setting the likelihood of following words that could generate an already observed n-gram to 0. Nonetheless, the number of n-gram penalties must be chosen with caution. A 2-gram penalty should never be applied to text talking about the district of "Castelo Branco", for example, otherwise, the name of the district would only appear once in the entire corpus. In open-domain text generation, beam search might not be the best possible option since it works well in tasks where the length of the desired generation is predictable [69], which is not the case for dialogue (where the desired output length can vary). It also suffers from repetitive generation and since human language does not follow a distribution of high probability next words [70].

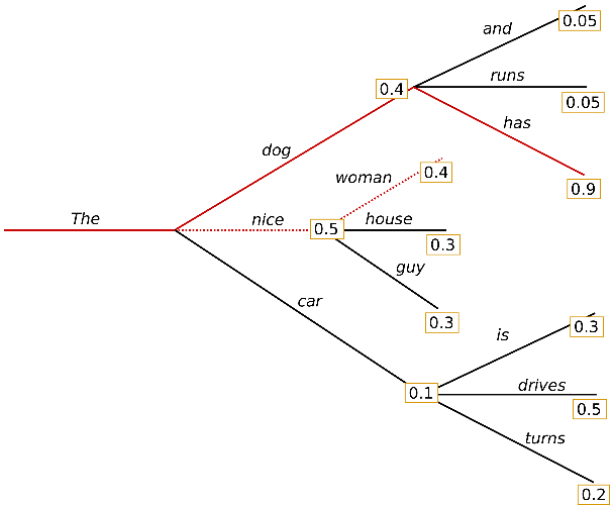

Figure 3.2: Beam Search. Source: How to generate text

To include some randomness in the generated text, Sampling was suggested. Sampling consists of choosing a word (w_t) according to its conditional probability distribution $w_t \sim P(w|w_{1:t-1})$. To make the distribution more accurate, the likelihood of high probability words can be increased and the likelihood of low probability words can be decreased. This can be handled by varying the value of a parameter called "temperature".

Later on, Top-K Sampling was introduced, it filters the K most likely next words and redistributes the probability mass among the K following words [71]. GPT-2 adopted this sampling scheme, which was one of the reasons for its success in story generation. In Figure 3.3, the illustration of the Top-K Sampling method, with K=6 can be observed. Top-K Sampling carries, however, some concerns, since it does not dynamically adapt the number of words that are filtered by the distribution $P(w|w_{1:t-1})$. Meaning that some words might be sampled from a flat distribution (left side of the Figure), whereas others might be from a sharp distribution (right side of the Figure).

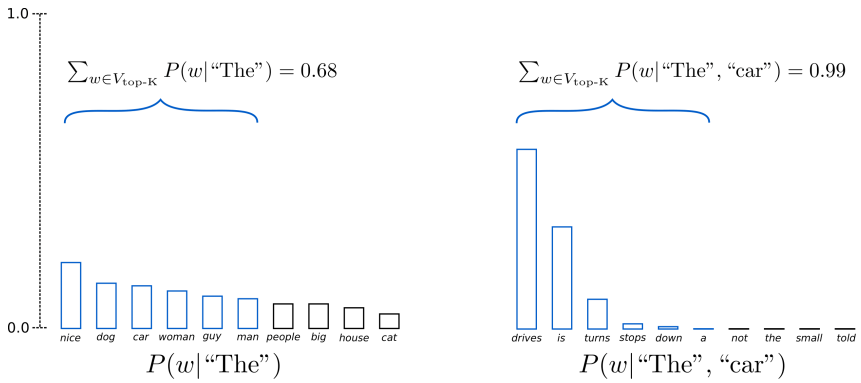

Figure 3.3: Top-K Sampling. Source: How to generate text

In the first step, Top-K removes the words "people", "big", "house" and "cat", which were not that

unlikely. On the other side, in the second step, Top-K includes ”down” and ”a” which don’t fit that well. Therefore, limiting the sample of words to a fixed size K could limit the models capability to work properly. To address this pain point, Top-P Sampling was created [70]. Instead of sampling the most likely K words, the method selects the smallest feasible combination of words whose cumulative probability exceeds the probability p. The probability mass is then redistributed among this collection of words. In this manner, the size of the set of words may be dynamically increased and decreased based on the probability distribution of the following word. This can be observed in Figure 3.4. For example, with p = 0.90, Top-P samples the minimum number of words that exceed this probability (90%). In the left side of the Figure, it includes the 9 most likely terms, but in the right side, it only needs to choose the top 3 words to reach 90%.

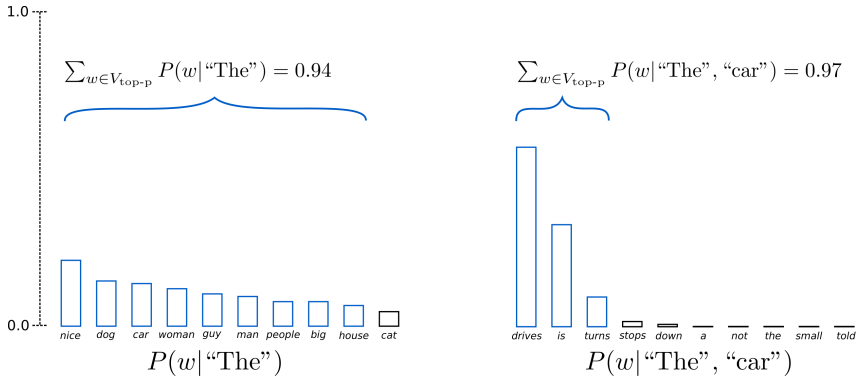

Figure 3.4: Top-P Sampling. Source: How to generate text

To achieve an even better performance, Top-P and Top-K can be used alongside. Hence, low ranked words can be avoided while having some dynamic selection. Therefore, this decoding method was the one chosen for this project.

Lastly, it is important to state that *HuggingFace Transformer* libraries enable the easy implementation of this different decoding and generation methods.

3.4 The Dataset

As previously mentioned, open-domain chatbots depend highly on the conversation databases used to train them. To accomplish this thesis goal, the models were fine-tuned with the two biggest Open-domain conversation databases, whose source language is English, that are available open source:

- **Daily Dialogue** [4]: Contains around 13,118 dialog utterances extracted from web pages for English learners. It includes a variety of topics (relationships, tourism, work, politics, among some others). This dataset is intended to resemble dialogues humans have in their ”daily life”.

- **Topical-Chat** [72]: Consists of more than 210,000 utterances, making it the largest social conversation and knowledge dataset available publicly. It underlies knowledge about 8 topics (Fashion, Politics, Books, Sports, General Entertainment, Music, Science & Technology and Movies).

4

The Implementation

Contents

In this chapter the system architecture will be described and the ability for a transformer to perform a NLP task, behaving as a chatbot, will be tested. Alongside the introduction of two different approaches, the model training process and the testing methods to evaluate it are going to be briefly explained.

4.1 System architecture

The systems described in Sections 4.1.1 and 4.1.2 are based on transfer learning, in which a pre-trained model is applied to a new problem. As previously seen, it is a popular method in DL, since it tries to exploit what was learned before, in order to improve generalization in the current task.

4.1.1 System 1: Fine-tune a pre-trained model

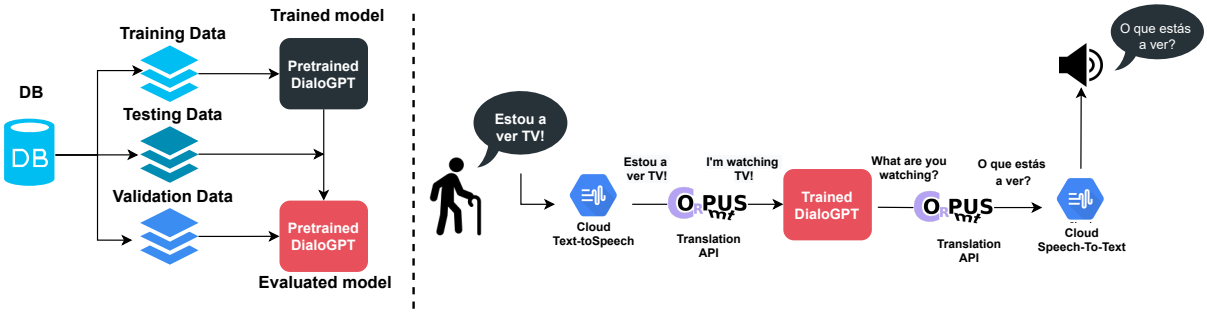

Figure 4.1: System 1 architecture with DialoGPT.

In figure 4.1, the main steps of the System 1 and the application scenario are shown. This project's purpose is to conduct a dialogue with a portuguese elder, to help with the loneliness problem mentioned in Chapter 1. The system was prepared with a speech-to-text API [73], followed by a translation API, which will translate the European Portuguese utterance to English. Afterwards, the translated utterance will be the input for the transformer. The latter will produce the output utterance, that will be translated to European Portuguese and passed through the text-to-speech module. The main idea is to make use of the already pre-trained model DialoGPT which was fine tuned with the two open-domain conversation databases mentioned in Chapter 3. The Transformers, the datasets adjustments and the modules to perform the training and evaluation will manipulated in Python, since it is the most widely used programming language to handle ML systems.

4.1.2 System 2: Fine-tune a pre-trained model on a new language

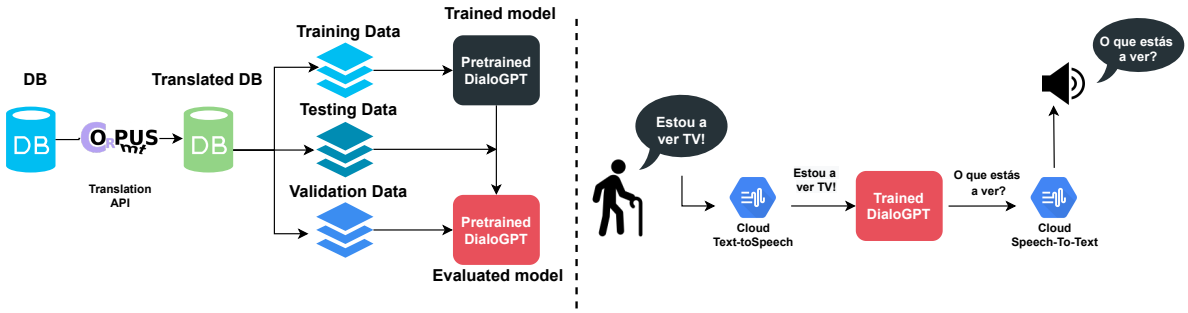

Figure 4.2: System 2 Architecture with DialoGPT.

In figure 4.2, the second system's architecture is shown The main idea is to fine-tune the models on a new dataset with a different language. Again, this project's purpose is to conduct a dialogue with a portuguese elder, so the system will also be prepared with a speech-to-text API and the model is going to be trained with only the small database, due to memory limitations, but translated to European Portuguese. The trained model will receive as input a portuguese sentence and will produce the output portuguese utterance, which will be passed through the text-to-speech module. The model, the dataset adjustments and module to perform the training will again be manipulated in Python.

4.2 Resources & Data

The experiment was going to be conducted in Google Colab, however its resources were not enough to train the small version of the model. As a result, the models were trained on a GEFORCE GTX 1080 Ti, with 11 Gbps. Even though this GPU permitted the training of the referred models, DL is still a subject with considerable computational demands. The chosen GPU will always highly influence the experience and the results.

The pre-trained models were tested before any fine-tuning was done, to see the experimental result baseline. As can be seen in the lists below, it is very repetitive and cannot keep a logical and engaging conversation.

DialoGPT-small

- **Human: Good morning.**
- Robot: Good morning everyone!
- **Human: How are you?**
- Robot: Good morning everyone!

- **Human: My name is Mariana.**
- Robot: I'm Mariana.
- **Human: What are you doing?**
- Robot: I'm Mariana.
- **Human: Bye.**
- Robot: I'm Mariana.

The model is repetitive and builds its predictions copying from the user input.

DialoGPT-medium

- **Human: Hello.**
- Robot: Hello.
- **Human: I'm going to my daughter's.**
- Robot: I'm going to my daughter's.
- **Human: I want to take a walk today.**
- Robot: I want to go to the beach.
- **Human: I like to have coffee.**
- Robot: I want to go to the beach.
- **Human: I want to take a walk this afternoon.**
- Robot: I want to go to the beach.

The medium model is again repetitive however it is able to build its predictions, that are logical in the dialogue context.

GPT2

- **Human: Good morning.**
- Robot: The first thing you need to know about the new iPhone 6 is that it's a very good phone. It's a very good phone. It's a very good phone. It's a very good phone. It's a very good phone. It's a very good phone. It's a very good phone. (...)

Lastly, the GPT2 is not even able to construct a logic "answer", which makes sense since it was not trained on dialogue data.

The fine-tune of the pre-trained models, was performed using the two datasets (in English) for System 1 and the smaller one (translated to Portuguese) for System 2. The datasets were converted in a way

that every sentence row contains 7 previous utterances for context. It was found that 7 was a good balance between having long enough context to train a conversational model and fit it within the memory constraints (longer contexts take more memory) [74].

Following that, the dataset was transformed into a format the model can understand. It was tokenized recurring to HuggingFace's tokenizers that were discussed in Chapter 3. With the data prepared, the model could finally be used, trained and tested.

4.3 Training

To train the model, a batch of examples is used, with both the inputs and the respective responses. This is due to GPT-2's auto-regressive property, meaning it uses some context to predict the next token. This prediction is then added to the original context and fed back in as the new context for generating the next token.

A brief overview will be done to some of the configuration variables and hyperparameters used for the training and evaluation of the model. Their purpose and the way they affect the model's behaviour will be discussed.

Listing 4.1: DialoGPT Training Parameters.

```
class Args():
    def __init__(self):
        self.output_dir = 'output-small'
        self.model_type = 'gpt2'
        self.model_name_or_path = 'microsoft/DialoGPT-small'
        self.config_name = 'microsoft/DialoGPT-small'
        self.tokenizer_name = 'microsoft/DialoGPT-small'
        self.cache_dir = 'cached'
        self.block_size = 512
        self.do_train = True
        self.do_eval = True
        self.evaluate_during_training = False
        self.per_gpu_train_batch_size = 4
        self.per_gpu_eval_batch_size = 4
        self.gradient_accumulation_steps = 1
        self.learning_rate = 1e-5
        self.lr_schedule = 'noam'
        self.weight_decay = 0.0
        self.adam_epsilon = 1e-8
        self.max_grad_norm = 1.0
        self.num_train_epochs = 3
        self.max_steps = -1
        self.warmup_steps = 0
        self.no_cuda = True
        self.seed = 42
```

```
        self.fp16 = True
        self.fp16_opt_level = 'O1'

args = Args()
```

- **Block Size:** Represents maximum number of tokens for each training instance. Up to a specific maximum sequence length, an encoder block made from original transformer paper can accept inputs up to 512 tokens. Sequences that are longer than the block size are truncated, while those that are shorter are padded with zeros.

- **Batch Size:** This hyperparameter has an influence on the accuracy of the DL models as well as their training process performance. The available GPU memory currently limits the range of batch size values. The maximum batch size that may be executed on a single GPU diminishes as the neural network grows in size. Today, as we run bigger models than ever before, the batch size options get smaller and may be far from ideal. Gradient accumulation is a simple approach to enable executing batch sizes that don't fit within the GPU memory. The batch size refers to the amount of samples (in this sentences) needed to train a model before changing its trainable model variables, such as weights and biases. It needs to be taken into consideration that very large batch sizes may cause bad generalization, implying that outside of the training set, the neural network will perform badly. However very small batch sizes may lead to slow convergence, since a single sample has a larger influence on the proposed variable updates. The primary idea is to experiment with different batch sizes until one is found, that would be optimal for the specific neural network and dataset in use.

- **Gradient Accumulation Steps:** The gradient accumulation is a method to prevail over the low GPU restriction and the need of utilizing lower batch sizes for training the model. It is a technique for splitting a batch of samples needed to train a neural network into multiple mini-batches that will be run consecutively. During back propagation, the parameters of the network are not changed in each step of the mini-batch, and the gradients results are accumulated. When all of the mini-batch steps are finalized, the models parameters are updated using the collected gradients. This process is as good as using higher batch size for training since gradients are updated the same number of times. For example, a Gradient Accumulation Steps of 8 and a batch-size of 4 is identical to using a batch-size of $8 \times 4 = 32$.

- **Learning Rate:** The learning rate is a hyperparameter that regulates how much the model changes each time the model weights are updated in response to the predicted error. A value too little may result in a long training process that becomes stuck, whereas a value too big may result in learning a sub-optimal set of weights too quickly or an unstable training process. The default learning rate

was selected based on validation loss. Each model is trained until there is no progress in validation loss [2].

- **Learning Rate Schedule:** The Noam learning rate scheduler is the default one. It is normally with used along the Adam optimizer [1]. For the first warmup steps, the Noam scheduler increases the learning rate linearly, then reduces it proportionally to the inverse square root of the step number.

- **Weight Decay:** Is a regularization whose goal is to prevent overfitting. It keeps the weights small and avoids exploding gradient.

- **Adam Epsilon:** Avoids the dividing by zero error, while updating the variable when the gradient is almost zero. Ideally epsilon should be a small value.

- **Number of training epochs:** It is a hyperparameter that sets the number of times the learning algorithm will run over the whole training dataset. Each sample in the training dataset has the chance to change the internal model parameters once each epoch. There must be one or more batches in an epoch. It can be described as a for-loop that runs for the number of epochs, with each loop running across the entire training dataset. Another nested for-loop is contained within this for-loop, which iterates over each batch of samples, each batch including the specified mini-batches.

- **Max number of steps:** This value is -1 by default. If it is set to a positive number, it will be the total number of training steps. It overrides the number of training epochs.

- **Warmup steps:** It is the number of steps used for a linear warmup from 0 to the value of the learning rate.

- **fp16:** This option changes from whether to use 16-bit (mixed) precision training instead of 32-bit training. When set to True, lowers the required memory and enables the training of larger models and training with larger batches. This option was set to true, due to the GPU limitation.

- **fp16 otimization level:** For fp16 training, Apex AMP optimization level selected in [‘O0’, ‘O1’, ‘O2’, and ‘O3’]. The ’O1’ is the mixed precision and it is the recommended one for typical use, when using 16-bit training.

After all these variables are set, the model is configured. Since the data is also prepared, it is possible to train the model. The complete developed software code (containing all classes, methods, and variables, as well as the scripts that were run) is documented in the following GitHub repository.

4.4 Metrics for NLG evaluation

4.4.1 Language modeling loss (for next-token prediction):

The HuggingFace trainer enables the computation of the loss. This metric on its own does not tell much. However, it is a good estimator for overfitting detection and to compare the various fine-tunings. When a model is called with the respective labels, the Cross Entropy Loss [75] between the predictions and the passed labels is given. After setting the optimizer (in this case the Adam optimizer), a backward pass is done and an update to the weights is performed.

4.4.2 Perplexity

One of the metrics frequently utilized to evaluate the model is perplexity, which is a basic yet effective metric that reflects relevancy. Perplexity measures how unsure the model is in its choice of the following token. The more unsure the model is, the higher its perplexity. Formally, perplexity is the number of choices the model is trying to choose from when it is producing the next token. One intriguing aspect of perplexity is that it closely resembles what people classify as coherent and specific natural conversations [74].

4.4.3 Dialog Ranking Pretrained Transformers (DialogRPT)

Conversational models are currently improving their capability to produce context-relevant turns, and to be more ”human-like”. In order to evaluate the models, the DialogRPT [76], was used. Since it was only trained in english it will only be used to evaluate the System 4.1.1.

DialogRPT is a set of GPT-2 models trained on 133M pairs of human feedback data (upvotes/replies of dialog systems). The ranker outperformed some baselines like perplexity. To rank the machine generated conversation, the feedback prediction models were combined with a human-like scoring model. Crowd-source evaluation showed that the DialogRPT ranker correlated better with real human preferences when compared to baseline models [76]. The following task was used to attend to this thesis project: *human_vs_machine* : ”How likely is the response human-written rather than machine-generated?”

The approach was to try to re-rank the fine-tuned DialoGPT outputs with DialogRPT using the tools of HuggingFace Transformers. After analysing the DialogRPT methods, a small Python script was written that was responsible for:

- Generating 5 different utterances to one input question;
- Outputting the fine-tuned DialoGPT generation probability, for each of those 5 sentences;

- Outputting the DialogRPT ranking probability for each of those 5 sentences;
- Saving one random utterance from the pool of 5, as the answer, for later use.
- Calculating the average generation and ranking probability for all answers.

Even though the DialogRPT paper [76] claimed very good results, some issues were found to this approach. For example, if the input is only ”Hi!”, the ranking probability of having ”Hi!”, as an answer, is considerably lower (3%) than ”Hi, Mr. Brown. I’m Steven. What seems to be the problem?” (55.4%). Which can make sense, since the answer is more complete, making it ”sound” more human. However the first choice would fit the purpose nicely and the second choice can deviate from the target person and the context.

To calculate these rankings, the Daily Dialogue validation dataset was split by questions and answers. Each part comprised 3870 sentences. The part with the questions is used as the model input.

4.4.4 BLEU, METEOR and ROUGE scores

To analyze the performance quantitatively BLEU [55], METEOR [77] and ROUGE [78] metrics were used. These metrics use statistical rules to measure the similarity between the output responses and reference responses. They were initially proposed for machine translation, however the same idea applies to evaluating generated text as it does to evaluating labels. If candidate text A matches one of the reference texts better than candidate text B, we want to give A a higher score than B.

Before diving into the evaluation metrics, there are two important measures to take into consideration. The first one is Precision that shows the percentage of your results which are relevant. The second one is Recall that refers to the percentage of total relevant results correctly classified by your algorithm.

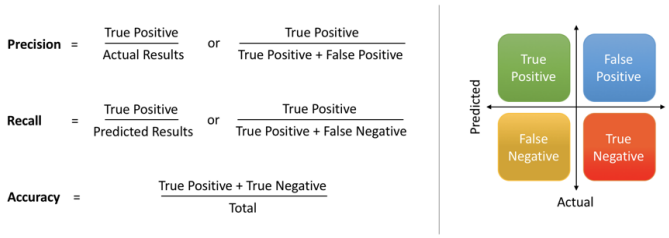

Figure 4.3: Source: Precision vs Recall

- **BLEU:**

  Or Bilingual Evaluation Understudy Score is an algorithm that was created with the goal of determining how accurate the machine translation was. Later it was found that it could be also used to evaluate the quality of text responses that a chatbot returns for an input text from a user. To esti-

mate precision the Modified n-gram precision is used, recurring to n-grams to compare an answer given by a chatbot with the reference text.

However, just using n-grams presents a problem. For example, consider the following two responses to a reference sentence and the unigram precision:

Reference: I can't go to school.

Response 1: He can't go to school.

Response 2: He can't go go to to school school.

Response 1 has a unigram precision of 60% while for 2 it is 75%. However, it is obvious that 2 is not a better candidate than 1. With the "modified" n-gram precision, it only matches the n-grams of the response as many times as they appear in the reference text. So, in the preceding example, the words will have only matched once, resulting in a 37.5% unigram precision. To include all the n-gram precision scores the following geometric mean is computed:

$$Precision = \exp\left(\sum_{n=1}^{N} \frac{1}{n} \log p_n\right) \tag{4.1}$$

To estimate recall, Best match length is used. It is natural to assume that a longer candidate text will have a higher percentage of some reference than a shorter candidate. Candidate texts are not very long, as this would result in a poor accuracy score. As a result, shortness can be penalized in candidate texts to introduce recall.

$$BP = \begin{cases} 1, & if \quad c > r \\ \exp(1 - \frac{r}{c}), & otherwise \end{cases} \tag{4.2}$$

In the equation above, c is the total length of the candidate, and r is the average length of all the references. As the candidate length decreases, the ratio r/c increases, and the BP decreases exponentially. Finally the BLEU score is given by:

$$BLEU = Precision \cdot BP \tag{4.3}$$

The evaluation was done with parameters from 1-gram to 4-grams to compute how good the chatbot's responses are. To understand how the BLEU scoring works, the following example can be visualized [79]:

Reference: I can't go to the school now.

Response: I can't come back from the school now.

1-gram	2-grams	3-grams	4-grams
66.67	57.73	46.03	0

Table 4.1: BLEU scores for the example above.

The Table 4.1 shows the BLEU scores for the sample above, from n-gram 1 to 4. The smaller the n-gram model, the higher the score, and in this case 0 for 4-grams since no sequence of four following words matches in both sentences.

- **METEOR:**

  Or Metric for Evaluation of Translation with Explicit ORdering is an automated metric for evaluating machine translation that is based on the idea of unigram comparison between machine and human translations [77] . It was introduced to tackle some of BLEUs weaknesses and also generate a strong connection with human judgment at the sentence. This differs from BLEU metric that seeks correlation at the text level [80]. The difficulty with BLEU is that since the BP value is based on averaged lengths throughout the whole text, individual sentence scores suffer. To tackle this, METEOR modifies precision and recall calculations with a weighted F-score based on unigrams and a penalty function for incorrect word order. The meteor score is given by:

$$METEOR = (1 - Penalty)F \tag{4.4}$$

  With:

$$Penalty = \gamma(\frac{c}{m})^{\beta}, 0 \leq \gamma \leq 1 \tag{4.5}$$

  Where c is the number of matching chunks and m is the total number of matches. The value of γ determines the maximum penalty and the value of β determines the functional relation between fragmentation and the penalty. And:

$$F = \frac{PR}{\alpha + (1 - \alpha)R} \tag{4.6}$$

  Suppose that m is the number of unigrams between the two texts. Precision and recall are given as m/c and m/r, where c and r are candidate and reference lengths, respectively. The free parameters in the metric, α, β and γ are tuned to achieve maximum correlation [81].

- **ROUGE:**

  Or Recall Oriented Understudy for Gisting Evaluation is based on recall, and is mostly used for summarizing evaluation. To evaluate this project's models, ROUGE-L were used. ROUGE-L is based on the longest common subsequence (LCS) between the model response and reference.

ROUGE-L uses F-score, which is the harmonic mean of accuracy and recall values, rather than just recall. Supposing that A and B are response and reference with lengths of m and n:

$$P = \frac{LCS(A,B))}{m} R = \frac{LCS(A,B)}{n} \tag{4.7}$$

Then F is then calculated as the weighted harmonic mean of P and R:

$$F = \frac{(1+b^2)RP}{R+b^2P} \tag{4.8}$$

The idea is that a longer common sequence indicates that the two sequences are more comparable. To better see how it works, the following example can be visualized:

Reference: I like school.

Response: I hello my name like school.

$$R(recall) = \frac{LCS(gram-n)}{count(gram-n)} = \frac{2}{3} = 66\% \tag{4.9}$$

$$P(precision) = \frac{LCS(gram-n)}{count(gram-n)} = \frac{2}{7} = 29\% \tag{4.10}$$

$$ROUGE-L(F1score) = 2 \cdot \frac{0.29 \cdot 0.66}{0.29+0.66} = 0.4 \tag{4.11}$$

Therefore we have a 40% F1 score.

Fortunately to simplify the calculation of these scores, the NLG-Eval [82] API was created and it was used to evaluate the trained models. As uttered above, the Daily Dialogue testing dataset was divided among questions and answers. The part with the questions is again used as the model input. The part with the answers is used as references, which are compared to the prediction of the model for each input.

4.5 Machine Translation Evaluation

When it comes to evaluating the performance, the ideal would be to have a human-based evaluation, specially in a system where paraphrasing can lead to inferior metrics, despite still having a good performance. Consequently, to add a score to the translated dataset [4], a small interface was developed.

Figure 4.4: Translation Quality Evaluation interface.

In this User Interface (UI), from a dataset storing almost 8000 source and target segments pairs, 14 random are displayed to the user. The user has then the option to classify the translation from 1-10. From the pool of translations, 784 were evaluated.

4.6 Chatbot Evaluation

Once more, as mentioned, human evaluation is preferable and this was done done by recurring to two methods.

For the past iteration of this thesis project [32], a form was built that followed a base conversation, keeping the robot speech while hiding all of the human utterances in the discourse. This was conducted with Google Forms and it was shared with a small sample of people (35 people). The resulting dataset can be visualized in appendix A. For the first method, to evaluate our model, we tested the different *Training Utterances* and check if its answer is equivalent to the *Corresponding Robot Utterance*. Two concepts were evaluated: the possibility of the prediction given by the model to be the same as the *Corresponding Robot Utterance*, or the possibility of the prediction to be a logical and a feasible answer to the input. If the answer does not fit in either of these categories, the response was marked as a fail.

The second method took a more direct human interaction. To add a score to the performance of the Chatbot, a small interface was created. In this User Interface (UI), the user has a chance to engage

in a conversation with the system. After a six utterance interaction with the system, the user has then the option to classify the conversation from 1-10. This experiment was conducted with approximately 40 dialogues.

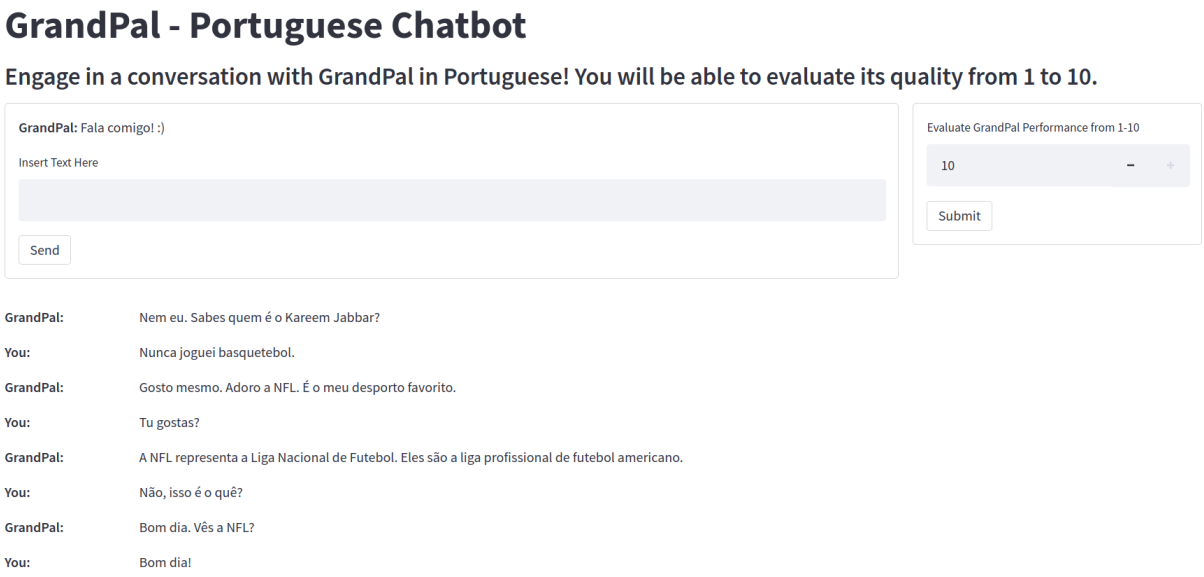

Figure 4.5: Translation Quality Evaluation interface.

One more advantage of this evaluation is allowing to analyze the ability of the model to ”remember” previous utterances during the dialogue. The model was trained with 7 sentences for context, therefore the input only stores the last 6 uttered sentences.

The drawback of this evaluation is that it is time consuming and expensive. One aspect to have in consideration is that the focus group should be the elderly, however with the ongoing Covid-19 pandemic, and with the fact that most people belonging to this group may not be accustomed with computers this possibility is set on standby.

The model was trained with different hyperparameters, to evaluate which were the best for the problem at hands.

4.7 Physical System

To have a practical view of the project, the system was built recurring to a Raspberry Pi Zero W, a speaker and a microphone. In order to handle the communication between the Raspberry Pi and the Deep Learning model, a simple Flask application was built.

With the aim of enhancing the performance and engagement of the Chatbot, a weather API and a radio API where also added. So taking into consideration the Figures 4.1 and 4.2 with the model architecture, the entire system architecture can be observed in the Figure below.

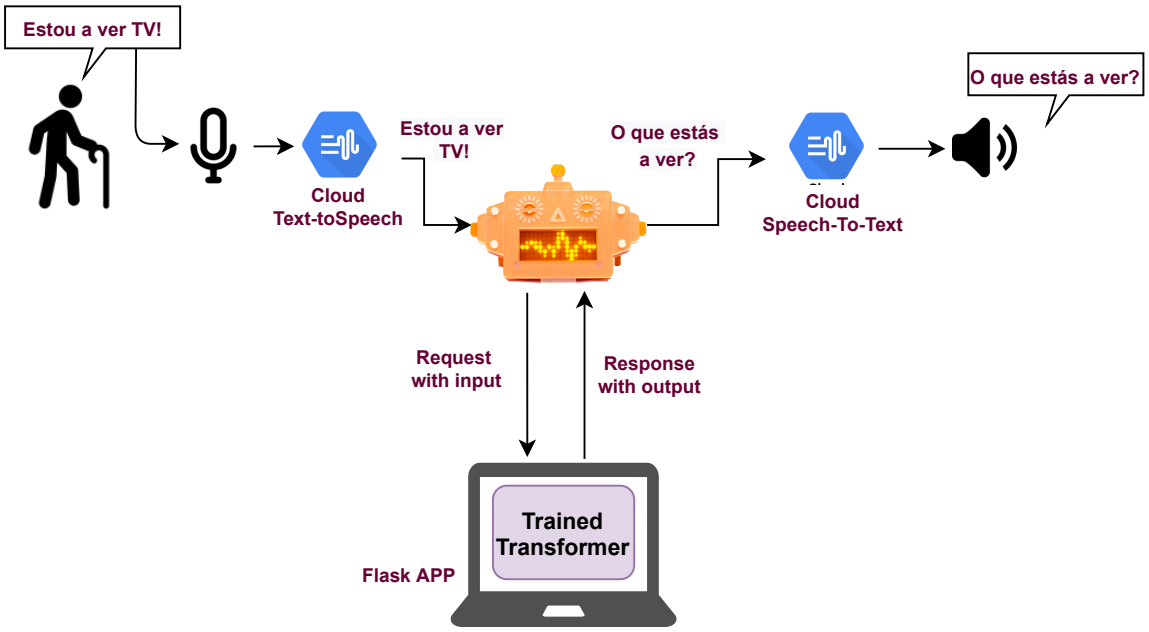

Figure 4.6: System architecture.

5

System Evaluation

Contents

In this Chapter the training results will be shown and discussed, for each dataset and model. The Machine Translation Evaluation and the system overall evaluation will be discussed too.

5.1 System 1 - Fine-tuning an English pre-trained model

5.1.1 Daily Dialogue

The first fine-tuned model was the **DialoGPT-small**. Some model ablation was conducted, to analyze the way the configuration variables and hyper parameters affected the model. In a first stage, to avoid having such a lengthy training, the model was only fine-tuned with the small English dataset, Daily Dialogue [4]. The memory limitation was inspected, along with the performance of the model, for each different training. The transformer was trained with 3 epochs, 5 epochs, and 7 epochs. With the increase of the number of epochs, the duration of the training rose too. Being, respectively, near to 4, 6 and 8 hours. The 8 trained models' results are shown in Table 5.1.

The goal in mind was to try to decrease the metrics *loss* and *perplexity*. These metrics were calculated using the Daily Dialogue testing dataset, that contained 8069 utterances. For each experiment, the *DialoRPT ranking* was also calculated, with the generation and ranking probabilities as mentioned in Section 4.4.3. The Daily Dialogue validation dataset was split into two. Here, we assume that a normal dialogue has a format of *'question-answer-question-answer...'*. Therefore, by splitting it, two new sets were obtained: Questions and Answers. The first containing the questions and the other with the answers, each with 3870 utterances. Finally, as referred in Section 4.4.4, the *Bleu-(1,2,3,4)*, *METEOR* and *ROUGE-L* scores were also computed. To estimate them, the Answers dataset was used as reference and the model's predictions to the Questions dataset was used as hypotheses. Even though this may not be the ideally suited method, since more than one reference to the scores computation should be used (paraphrasing), it still enabled a performance contrast among the several experiments.

The datasets used for training, testing and validation (questions and answers sets) can be found in the following GitHub repository.

DialoGPT small - Daily Dialogue													
Nº	**BatchSize/ GPU**	**Grad. Acc.**	**Epochs**	**Perplexity**	**Loss**	**Gen**	**Rank**	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**	**METEOR**	**ROUGE-L**
RAW	-	-	-	-	-	57.54%	36.22%	4.00%	1.31%	0.52%	0.23%	4.64%	5.31%
1	2	4	3	11.82	1.36	58.33%	57.31%	4.81%	1.97%	1.07%	0.67%	6.13%	6.73%
2	2	32	3	10.95	1.90	52.89%	54.93%	4.03%	1.54%	0.72%	0.34%	5.71%	6.30%
3	2	32	5	10.99	1.67	55.65%	55.60%	4.52%	1.89%	1.01%	0.60%	6.16%	6.78%
4	4	16	5	9.68	1.44	55.26%	55.83%	4.20%	1.65%	0.83%	0.45%	5.75%	6.25%
5	4	8	5	10.17	1.26	57.57%	56.99%	4.66%	1.83%	0.94%	0.55%	6.06%	6.58%
6	4	12	5	9.81	1.40	56.11%	56.40%	4.59%	1.87%	0.98%	0.55%	5.96%	6.66%
7	2	32	7	10.15	1.45	57.49%	56.96%	4.36%	1.67%	0.85%	0.47%	5.90%	6.25%
8	4	16	7	11.56	1.24	57.06%	56.84%	4.56%	1.81%	0.95%	0.56%	5.92%	6.35%

Table 5.1: DialoGPT-small training results with Daily Dialogue dataset

The first row in Table 5.1, shows the performance of the raw model with no fine-tuning. This way, it allows a better understanding of the fine-tuning's impact.

Making an analysis to Table 5.1, it is clear that the model that showed the worst performance was the raw model. Therefore, it can be deduced that fine-tuning a model to a narrower domain can lead to an improvement. The red highlighted values point the poorest score, whereas the values highlighted in green show the best value, for each metric. In addition, we see that with the increase of the number of epochs, the system tends to overfit. For example in the experiment Nº 8, the loss value decreased, however the perplexity increased significantly.

Finally, the model with the overall best scores was experiment Nº 1, since it is the one with more green highlighted values. Nonetheless, experiment Nº 4 was the one with the lowest perplexity (being the most confident model) and experiment Nº 3 was the one with the best scores for METEOR and ROUGE-L. As noted in Section 4.4.4, these latter metrics are better evaluating the Chatbot task.

To conclude which experiment was the most successful, the models experiments Nº 1, Nº 3 and Nº 4 will be compared. The *Utterances* from Appendix B, Table B.2, will be inputed into the model and the predictions were juxtaposed with the *Corresponding Robot Utterance*. As mentioned in Section 4.6, two concepts were surveyed: the possibility of the prediction given by the model to be the same as the *Corresponding Robot Utterance*, or the possibility of the prediction to be a logical answer. If the answer does not fit in these categories, the response falls out of context. One setback of this evaluation is that it does not have the conversation history in consideration, however it enables us to see if the model replies adequately to questions.

Experiment Nº 1

For the first model, from the 45 given utterances, 7 (11, 20, 21, 33, 36, 42 and 44) fell out of context. Nevertheless, 5 utterances (1, 7, 17, 25, 28) matched the *Corresponding Robot Utterance* and the remaining utterances were feasible answers. Some examples of these feasible answers are:

- **Human: I'm fine, too.**
- Robot: I'm glad to hear that.
- **Human: What's the weather for this morning?**
- Robot: The weatherman says it's going to snow.
- **Human: I'm going to the supermarket today.**
- Robot: You mustn't go shopping alone.

To show an example of a failed answer:

- **Human: I'm having lunch with my brother today.**

- Robot: What do you think of the film?

One other curious fact to note is that, even though some of the answers fell out of context, all sentences were written correctly and were coherent.

Experiment Nº 3

For this experiment, from the 45 given utterances, only 4 (10, 11, 12 and 27) fell out of context. Nonetheless, just 2 utterances (29 and 37) matched the Corresponding Robot Utterance. The remaining utterances were feasible answers. Below are again some examples of feasible answers:

- **Human: I'm going to my daughter's.**
- Robot: Oh, that's so nice of you.
- **Human: What's the weather for this morning?**
- Robot: It's supposed to snow in the morning.

Once again, all the uttered sentences were logical and coherent.

Experiment Nº 4

For this model, from the 45 given utterances, 6 (7, 11, 19, 38, 39, 40) fell out of context. Nevertheless, 4 utterances (1, 24, 25, 26) matched the *Corresponding Robot Utterance* and the remaining utterances were feasible answers. Again, a feasible answers is:

- **Human: Is everything all right with you?**
- Robot: Not at all.

Showing a failed answer:

- **Human: With me this is all magnificent.**
- Robot: What's the matter with you, Mr. Smith? You look down in the dumps.

All sentences, once more, were written correctly and were coherent.

Comparing the three experiments, although Experiment Nº 1 had more answers that matched the *Corresponding Robot Utterances*, it also had more failed answers. The model with the best execution was the one from Experiment Nº 3, since it was the one with less failed predictions. It can be concluded that, despite not having the best scores for each single metric, it is the one with the best average and also the best performance.

The second fine-tuned model was **DialoGPT-medium**. The memory limitations became substantial when performing the fine-tuning with this medium model. First off, the training took 17 hours to complete.

Secondly, all the training with a higher Batch Size/GPU or Gradient Accumulation than the one shown in the Table below, failed. The same occurred when the number of epochs was increased. Lastly, this model is really heavy when compared with the DialoGPT-small. The comparison between the raw model and the fine-tuning experiment can be observed in Table 5.2.

DialoGPT medium - Daily Dialogue													
Nº	**BatchSize/ GPU**	**Grad. Acc.**	**Epochs**	**Perplexity**	**Loss**	**Gen**	**Rank**	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**	**METEOR**	**ROUGE-L**
RAW	-	-	-	-	-	56.29%	36.88%	4.40%	1.55%	0.67%	0.29%	5.07%	5.46%
1	1	8	3	1.63	0.49	80.29%	60.16%	8.16%	4.93%	3.65%	2.92%	8.52%	10.25%

Table 5.2: DialoGPT-medium training results with Daily Dialogue dataset

Even without being able to do an extensive number of experiments, as deducted before, the fine-tuning of a model improves its performance. The generation and ranking scores rose above the ones from the raw model, and even the ones from the previous experiments. All the remaining metrics almost doubled their value, when put against the raw model.

To test if the experiment successful, and the model is able to conduct a conversation, the utterances from Appendix B, Table B.2, were again inputed into the model and the prediction was compared with the *Corresponding Robot Utterance*.

Experiment Nº 1

The model, from the 45 given utterances, fell out of context 9 times (2, 11, 12, 18, 22, 30, 32, 36 and 38). However, 5 utterances (1, 24, 26, 27, 28) matched the *Corresponding Robot Utterance* and the remaining sentences were logical answers. A feasible answers could be:

- **Human: I can't get through without a cup of coffee a day.**
- Robot: That's true. Coffee helps me wake up, keep alert, and improve my short term memory.

To show an example of a failed answer:

- **Human: Hello.**
- Robot: Hello. Is this Ann?

In this case, even though the answer is well written and logical it doesn't fit the project's purpose or domain.

Once more, even though some of the answers fell out of context, all sentences were again written correctly and were coherent. One other interesting fact to note is that this model was more eloquent and was better to constructs its predictions. This can be understandable since the medium model has almost the triple of the number of parameters of the small model (345M vs 117M). An overview to the results with this dataset is shown in Figure 5.1.

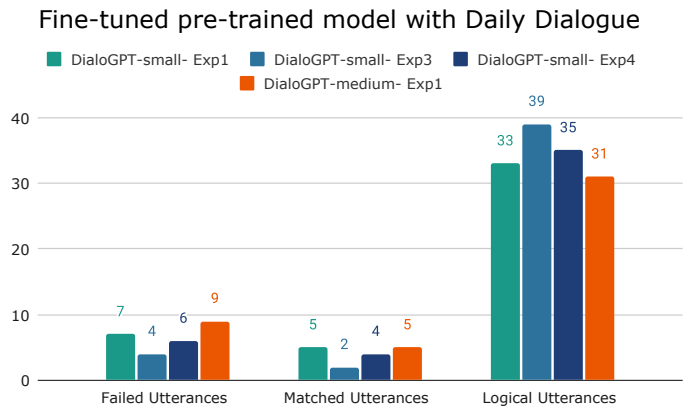

Figure 5.1: Fine-tuned pre-trained model with Daily Dialogue .

It can be concluded that the best performance was the one with DialGPT-small, Experiment Nº3, event though it was the one with less matched utterances, it was almost always able to answer coherently.

5.1.2 Topical Chat

DialoGPT-small was again fine-tuned. Now with the biggest English dataset, Topical Chat [31], available open-source. The memory limitation was once again inspected, along with the performance of the model, for each fine-tuning trial. These experiments were longer than the ones conducted with the smaller dataset. As before, with the increase of the number of epochs, the duration of the training rose too. Being, respectively for 3, 5 and 7 epochs, near to 11, 14 and 18 hours. Once again, the red highlighted values point the poorest scores, whereas the values highlighted in green show the most satisfactory value, for each metric. The 4 trained models' scores are shown in the Table below.

DialoGPT small - Topical Chat													
Nº	**BatchSize/ GPU**	**Grad. Acc.**	**Epochs**	**Perplexity**	**Loss**	**Gen**	**Rank**	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**	**METEOR**	**ROUGE-L**
RAW	-	-	-	-	-	57.54%	36.22%	4.00%	1.31%	0.52%	0.23%	4.64%	5.31%
1	2	4	3	16.25	2.87	61.72%	64.89%	7.58%	2.18%	0.74%	0.25%	6.32%	6.65%
2	2	16	5	15.00	2.77	61.41%	64.38%	7.59%	2.07%	0.72%	0.27%	6.27%	6.60%
3	2	32	5	14.27	2.66	60.41%	63.50%	7.60%	2.10%	0.73%	0.30%	6.33%	6.66%
4	2	32	7	17.63	1.82	60.41%	65.21%	7.56%	2.04%	0.64%	0.20%	6.30%	6.67%

Table 5.3: DialoGPT-small training results with Topical Chat dataset

The memory limitations were prominent when training the model with a larger dataset. GPU limitations did not allow to train the model with more than 2 Batch Size/GPU, or with a greater gradient accumulation than the one shown. The same occurred when the number of epochs was increased. Once more, the raw model had a significantly worse performance when compared to the conducted experiments. The training with the finest metrics was Nº 3. This experiment shows the most confident

model (lower perplexity), shows good ranking scores, and the best Bleu-1, Bleu-4 and METEOR. Similarly to before, with 7 epochs the model starts to overfit. This can be concluded since it shows the higher perplexity, and also the lowest loss. The evaluation, for the *perplexity* and *loss*, was done using the Topical Chat testing dataset, that contained 11736 utterances.

To estimate the rankings and *BLEU*, *METEOR* and *ROUGE-L* scores, the splitted Daily Dialogue validation dataset was used. Once more, this may not be the most precise method. However, it allows the contrast for the several experiments.

To test which experiment was the most successful and see if the model is able to dialogue, the experiments Nº 2 and Nº 3 will be compared. The utterances from Appendix A, Table B.2, were again inputed into the model and the prediction was compared with the *Corresponding Robot Utterance*.

Experiment Nº 2

The model, from the 45 given utterances, fell out of context 9 times (8, 10, 12, 18, 20, 21, 26, 27 and 43). Only 1 utterance (1) matched the *Corresponding Robot Utterance* and the remaining sentences were logical answers. Showing a feasible answer:

- **Human: What's the weather for today?**
- Robot: It's going to be really cold, I think. I'm not sure what the forecast is though.

To show an example of a failed answer:

- **Human: I love tea.**
- Robot: I love the tea too. I think it is important to have a healthy relationship with the tea party.

In this last example, even though the answer is again well written it misses the purpose and it lacks logic.

Experiment Nº 3

For this model, from the 45 given utterances, 8 (3, 7, 11, 21, 24, 25 and 36) fell out of context. None of the utterances matched the *Corresponding Robot Utterance*, so the remaining sentences were logical answers. One example of a feasible answers is:

- **Human: Today I'll be with my son.**
- Robot: You are a great dad! Do you like to read?

To show an example of a failed answer:

- **Human: Everything's great with me.**
- Robot: Did you know that the president of Indonesia has released 3 pop albums while in office?

In this last example, even though the answer is well organize, once again misses the purpose and it is to some extent, random.

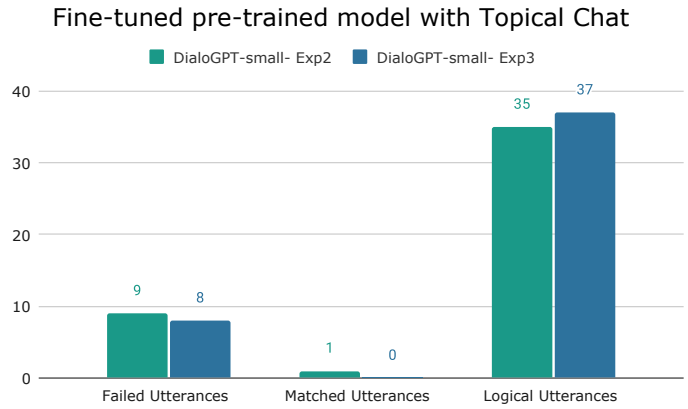

Figure 5.2: Fine-tuned pre-trained model with Topical Chat.

Analyzing both experiments results results and Figure 5.2, it can be concluded that Experiment Nº 3, had a better performance, since it had one less failed predictions and more logical outputs. Comparing these experiments with the ones from the dataset before, despite the fact that the metrics had better scores, the model failed more to answer accordingly. It was identified that the times the model fell out of context, its answer was to introduce a new topic or to present a random fact. This led to the thought that this dataset may not be as appropriate for the task that wants to be achieved. Nevertheless, it is engaging to enrol in a conversation with this model, since it teaches a various amount of trivias.

This dataset, Topical Chat, could not be used to train the DialoGPT-medium model, due to the GPU constraint.

5.1.3 Daily Dialogue + Topical Chat

Comparing the performance of the DialoGPT-small trained on the Daily Dialogue and trained on the Topical Chat, it can be inferred that with more data in the training, the model becomes more eloquent and smarter (since it is trained with more information). And even though the model's perplexity increased in the latter, overall the ranking and the metrics scores were better.

Therefore, an experiment was conducted where both datasets, Daily Dialogue and Topical Chat, were merged. Resulting in a bigger dataset with a total of 284153 utterances. In order to compute the perplexity and the loss, the model evaluation was done with both testing sets from Daily Dialogue and from Topical Chat. With the results from each dataset, the mean was calculated. The training, with 3 epochs, took around 15h and the one with 5 epochs took 24h. The results are shown below. The attempts to increase the BatchSize/GPU and gradient accumulation failed, again due to the limited GPU.

The same occurred when the number of epochs was increased.

DialoGPT small - Daily Dialogue + Topical Chat														
Nº	**BatchSize/ GPU**	**Warmup Steps**	**Grad. Acc.**	**Epochs**	**Perplexity**	**Loss**	**Gen**	**Rank**	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**	**METEOR**	**ROUGE-L**
RAW	-	-	-	-	-	-	57.54%	36.22%	4.00%	1.31%	0.52%	0.23%	4.64%	5.31%
1	2	200	16	3	13.05	2.56	65.47%	53.21%	3.90%	1.78%	1.00%	0.60%	6.19%	6.99%
2	2	200	32	3	13.23	2.57	64.13%	52.70%	3.63%	1.64%	0.93%	0.56%	5.94%	6.72%
3	2	200	16	5	13.54	2.59	68.11%	55.50%	4.75%	2.39%	1.52%	1.04%	6.88%	8.07%
4	2	200	32	5	13.13	2.50	66.13%	54.10%	4.05%	1.87%	1.10%	0.71%	6.30%	7.14%

Table 5.4: DialoGPT-small training results with Daily Dialogue + Topical Chat dataset

Comparing the results from Tables 5.1 and 5.3 with Table 5.4, it can be noted that the best generation probabilities and ROUGE-L scores were obtained with this dataset.

To test the success of the experiments, and see if the models are able to dialogue, the utterances from Appendix B, Table B.2, were inputed into the models and the predictions were compared with the *Corresponding Robot Utterance*.

Experiment Nº 1

The model, from the 45 given utterances, fell out of context only 2 times (3 and 11). 3 utterances (1, 25, 30) matched the *Corresponding Robot Utterance* and the remaining sentences were logical answers.

Experiment Nº 3

The model, from the 45 given utterances, fell out of context only 1 time (11). 1 utterance (1) matched the *Corresponding Robot Utterance* and the remaining sentences were logical answers. Some examples of the feasible answers are:

- **Human: Today I'll be with my son.**
- Robot: Oh, that's good. I'll talk to you later.
- **Human: I'm going shopping today, I need food.**
- Robot: Are you shopping online?

To show an example of one failed answer:

- **Human: With me this is all magnificent.**
- Robot: I think it's a beautiful piece of work. I am not a fan of the style, but I appreciate the work that went into it.

In this last prediction, even though the answer is well written, it misses the context of the user input. We should note, observing the original source utterance and the target segment, that the translation is not perfect. From "Comigo está tudo magnífico." the correct translation should be something near "With

me everything is magnificent.", instead of "With me this is all magnificent.".

Experiment Nº 4

For this model, from the 45 given utterances, only 2 (7 and 18) fell out of context. 4 utterances (1, 2, 30, 32) matched the Corresponding Robot Utterance and the remaining sentences were logical answers. An example of a feasible answer is:

- **Human: How's the afternoon weather?**
- Robot: Not too bad today. It's supposed to get cold tomorrow. I didn't know that the sun is actually white.

To show an example of two failed answers:

- **Human: Everything's great with me.**
- Robot: Hi. Are you interested in politics?
- **Human: I must go to my son's house.**
- Robot: Hello. Did you know that the first president of Zimbabwe was called President Banana.

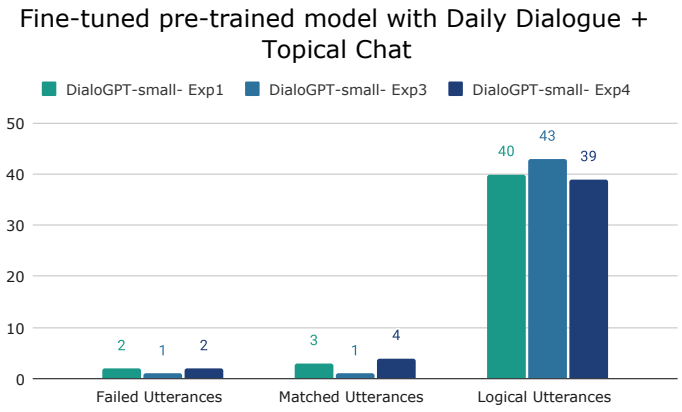

Figure 5.3: Fine-tuned pre-trained model with Daily Dialogue + Topical Chat.

These experiments are the proof that with a bigger dataset, the model is able to generalize more, "learning" how to dialogue in a wider domain. Despite the fact that the scores are not the best in comparison to the previous experiments, the model from experiment Nº 3 was the one that had the best performance when conducting a dialogue, only having failed 1 utterance.

Equally to the previous section, it was not possible to use this big dataset to fine-tune the DialoGPT-medium model, again due to the GPU constraint.

5.2 System 2 - Fine-tuning model on a Portuguese Dataset

5.2.1 Translated Daily Dialogue

Since this thesis project's goal is to have a Portuguese speaking Chatbot, we also tried to fine-tune the model on a new language, as stated in 4.1.2.

To obtain the portuguese dataset, the OPUS-MT model, mentioned in Section 2.7, was used to translate the Daily Dialogue training, testing and validation datasets.

For this model and dataset, we strove to put the same configuration variables and hyper parameters as the previous best models. All the training with a higher Batch Size/GPU or Gradient Accumulation than the one shown in the Table below, failed. The same occurred when the number of epochs was increased. This can be justified by the length of sentences, in words and characters. Normally portuguese words are bigger than english words, and the same happens regarding the length of the sentences [83]. Hence, the memory used is greater.

The metrics were calculated using the translated Daily Dialogue testing dataset, containing 8069 utterances. For each experiment, the *Bleu-(1,2,3,4)*, *METEOR* and *ROUGE-L* scores were computed.

The first two rows in Table 5.5, shows the performance of the raw model with no fine-tuning with the english Daily Dialogue validation dataset and with the translated Daily Dialogue validation dataset.

The datasets used for training, testing and validation (Questions and Answers sets) can be found in the following repository.

DialoGPT small - Translated Daily Dialogue												
Nº	**BatchSize/ GPU**	**Warmup Steps**	**Grad. Acc.**	**Epochs**	**Perplexity**	**Loss**	**Bleu-1**	**Bleu-2**	**Bleu-3**	**Bleu-4**	**METEOR**	**ROUGE-L**
RAW	-	-	-	-	-	-	4.00%	1.31%	0.52%	0.23%	4.64%	5.31%
RAW-PT	-	-	-	-	-	-	3.37%	0.63%	0.12%	0.00%	2.66%	3.33%
1	2	200	8	3	6.87	1.93	3.02%	1.33%	0.72%	0.41%	3.64%	5.18%
2	2	200	16	3	7.22	2.05	2.36%	0.87%	0.40%	0.21%	3.16%	3.97%

Table 5.5: DialoGPT-small training results with the translated Daily Dialogue dataset

It is curious to see that the Bleu-1 score, from the RAW-PT model is higher than the score from the fine-tuned model. This can be justified since the raw models, as already seen, are very repetitive. For that reason, if the model knows how to utter some portuguese words and they happen to be in the reference, the score increases as many times as they appear.

To the same input question "Sim, tenho um entendimento geral.", the reference answer and the RAW-PT model prediction were compared.

- **Answer reference:** Eu acho que você já tem bom conhecimento sobre a nossa empresa.
- **Model Hypothesis:** Eu n o fazia eu acho que acho que eu n o fazia eu acho que eu acho que eu acho que eu acho que acho que acho que eu acho que acho que eu acho que acho que acho que eu acho que.

Looking into the Model Hypothesis above, it is interesting to note that the model, is able to create portuguese words without being trained on a portuguese dataset. Leaving us to guess that the data extracted from Reddit comment chains probably had some portuguese sentences. And it is amazing to observe that the model was able to ”identify the language” and answer in accordance to the language.

Analysing the rest of the Table, we conclude that from the two experiments, the model that performed best was Nº 1, however the metrics are still poorer when compared to the RAW model.

To test the success of the experiment, and see if the model is able to dialogue, the utterances from Appendix B, Table B.1, were inputed into the model and the prediction was compared with the *Corresponding Robot Utterance*.

Experiment Nº 1

The model, from the 45 given utterances, 15 (1, 6, 7, 8, 12, 15, 20, 25, 27, 31, 33, 34, 36, 40 and 45) were logical answers. None of the utterances matched the Corresponding Robot Utterance, and the remaining sentences fell out of context. One example of a feasible answers is:

- **Human: Vou jantar hoje com a minha filha.**
- Robot: Porque é que não vais jantar amanhã?

To show an example of a failed answer:

- **Human: Olá.**
- Robot: Olá, Mary. O que se passa contigo?

In this last prediction, even though the answer is well written and logical it doesn't fit the project's purpose or domain.

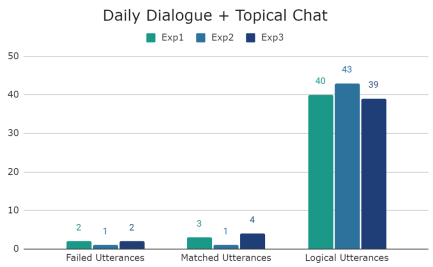

Figure 5.4: Fine-tuned pre-trained model with translated Daily Dialogue

It is a fact that this model was the one that perform the worse, in comparison to the previous models. By training the model with a small, translated dataset, one layer of error is already being added to the system. This is due to the fact that the translation is not flawless.

As seen, the DialoGPT-small model trained on the Daily Dialogue has some issues. By translating the dataset, the performance only decreased more. However, since this thesis purpose is to have a portuguese speaking robot, another approach will be evaluated, which consists in translating the input and output of the model that performed best: DialoGPT-small model, fine-tuned on the merged dataset (Daily Dialogue + Topical Chat), experiment Nº 3.

Similarly to the previous sections, it was not possible to use this dataset to fine-tune the DialoGPT-medium model, or even train the DialoGPT-small model with the Topical Chat translated dataset, again due to the GPU constraint.

5.3 Machine Translation Evalutation

To evaluate the Machine translation model, the human classifications from the interface mentioned before were analyzed. From the 8000 utterances, 784 were evaluated. Taking into consideration that 14 sentences are displayed per page, there were 56 evaluated pages. Counting with, approximately, 2 pages per person, we estimate a participation of 30 people. The results can be analyzed in Figure 5.5. This grading goes in accordance to the belief mentioned that for every translated sentence, its message is communicated correctly in almost all cases, since most of the translations obtained the maximum grade.

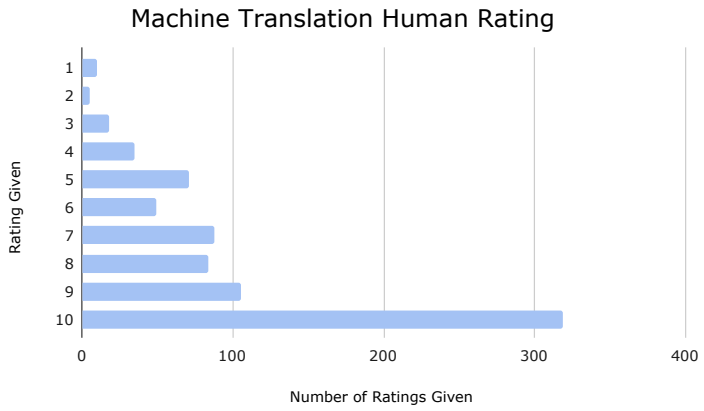

Figure 5.5: Human Evaluation to the Machine Translation model.

5.4 Pipeline evaluation

To complete this result analysis, an experiment was conducted were the whole system was put into action.

Firstly only part of the system was tested. Following the Diagram from Figure 4.6, the Trained transformer used was the one from Experiment Nº 3, trained on the merged dataset (Daily Dialogue + Topical Chat) and the Translation API was OPUS-MT.

The first test, similarly to before was to use the utterances from Appendix B, Table B.1. The portuguese utterances were inputed into the system, with the translation API for the input and output. The prediction was compared with the *Corresponding Robot Utterance*. It should be noted that we are introducing again some error layers. The first error layer is the translation. Even though the model is able to answer logically, the translation might miss the context.

For this system, from the 45 given utterances, 2 were wrongly translated but only 1 fell out of context. None of the utterances matched the *Corresponding Robot Utterance*, and the remaining sentences were logical answers. One example of a feasible answers is:

- **Human: Hoje vou passar o dia com os meus netos.**
- Robot: Oh bom, bem espero que gostes do resto do teu dia, foi um prazer falar contigo, espero que o teu dia tenha sido muito melhor.

To show some examples of failed translations:

- Source: What's up? Do you like rap music?
- Translation: O que se passa? Gostas de música rap?
- Correct: Tudo bem? Gostas de música rap?

- Source: It's supposed to snow in the afternoon but it's going to be drizzling all afternoon.
- Translation: É suposto nevar à tarde, mas vai estar a passar a tarde toda.
- Correct: É suposto nevar à tarde, mas vai estar a chuviscar a tarde toda.

To show an example of a failed answer:

- **Human: Tudo bem?**
- Robot: Sim, não sou um grande fã desse tipo de coisas. Sabia que é ilegal aquecer o teu carro em Ohio. Isso é muito estranho.

Secondly, the entire system was tested. The conversion from speech-to-text [73] and vice-versa was added. With this addition one second layer of error was added too, because if the API fails to do the conversion, the whole execution will fail.

Finally, for the last test, the human classifications taken from the interface interactions were analyzed. We counted with the participation of around 30 people resulting in 40 conducted dialogues. The results can be analyzed in Figure 5.6. This grading means that the Chatbot is able to conduct a fair dialogue, being 7 the rating with more occurrence.

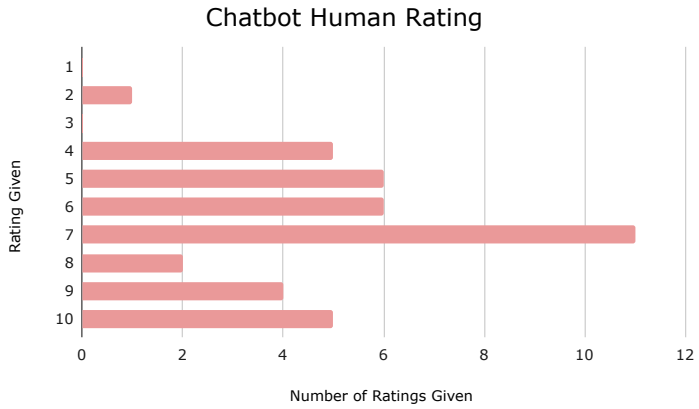

Figure 5.6: Human Evaluation to the System.

5.4.1 Summary

In summary, some of the conclusions that could be taken from these experiments are:

- Fine-tuning with more data, improves the model performance, for every conducted experiment;
- Increasing the number of epochs, for the DialoGPT-small model, to more than 7 leads to an overfit, for the Daily Dialogue and the Topical Chat Datasets. For the merged dataset the training with more than 5 epochs was not possible due to the limit of resources;
- For DialoGPT-small, having a BatchSize of 2 and a Gradient Accumulation of 32 led to the best execution;
- Fine-tuning a pre-trained model on another language with a translated dataset does not led to a good result;
- The MT model, for each translated utterance is able to correctly transmit the intended message.
- The system, was able to conduct a logical conversation with a user, almost every time.

6

Conclusion & Future Work

Contents

"Our imagination is the only limit to what we can hope to have in the future." - Charles F. Kettering.

In this Chapter the conclusions of this work will be presented, along with the possible future Work.

6.1 Conclusion

This thesis created a feasible solution to the problem defined in chapter 1: given that a system that generates a robot utterance for every human utterance, allowing a coherent conversation between a person and a robot is obtained. Nevertheless, before concluding that the obtained system is the best solution, an overview of the system and the evaluation conducted to the model should be inspected.

First of all, one asset of this solution is that the system architecture is based on State-of-the-Art transformer. Nevertheless, deciding on a Deep Learning path brings some challenges such as:

- The need of having huge amount of data for the model training;
- The need of a good graphics card (GPU) with a considerable memory size;
- The lack of data specialized in elderly dialogue.

Notwithstanding, despite those drawbacks, the possibility of using Transfer Learning makes up for them. In this thesis, it was presented, a moderate open-domain chatbot that was trained with limited GPU resources.

Second of all, with the use of Machine Translation the obstacle of the lack of Portuguese dialogue data was practically surpassed. It cannot be presumed that it was entirely surpassed since adding the layers of translation also adds a new layer of error. Nevertheless, as acknowledged several times, a perfect translation might not be possible, but the intent is passed along almost every time, as can be concluded by the Machine Translation Evaluation grading that was given by human testers, by testing the model in the User Interface.

Third of all, the system could benefit if the conversational data was not only comprised of general conversation. The best possible solution could be to have conversations recorded in nursing homes and transcribed to text. However, gathering enough data from the nursing home context would require human resources beyond the available ones for this thesis. Therefore, extracting tons of conversational data from open-source websites is considerably more efficient.

Coming down to the evaluation conducted to the model, all the trained models obtained better results than their corresponding baselines. As can be inferred by the Chatbot Evaluation grading seen in 4.6 the model has the ability to conduct a coherent dialogue. Regardless of the success of the developed system, it is not without flaws. Inspecting some of the utterances that were wrongly predicted by the system, it was observed that the predicted phrases introduces one random topic leading to the belief

that it has not been trained with sentences similar to the input. The most adequate model for human-robot conversation is the model DialoGPT, fine-tuned in the English-based (formed by Daily Dialogue with Topical Chat) dataset, with a translation layer to the input and output.

The robot responds to the last human phrase uttered. It also considers the course of the conversation, due to the introduction of the last human and robot utterances, as well as the history of human inputs to the model and the robot predictions.

6.2 Future Work

Future work could focus on a more in-depth study of certain transformers, new ideas to test various systems, to turn the training more efficient, or to repeat the experiments with better resources.

The following ideas could be tested and implemented:

- Each passing day new and better transformers models, for a huge variety of tasks, are released. It could be interesting to try other conversational transformers and compare the performance;
- Recur to better GPUs, that could allow the carrying on of experiments with higher Batch Sizes and with more training epochs. This suggestion might really improve the performance of the model and a better understanding of the impact of the model ablation;
- Evaluate the MT model responsible for the translation with English language as source and Portuguese language as target.
- Implement the system with a better translation model. Even though the translation results were quite impressive with the open-source MT model, there are better models now (like Unbabel, DeepL) that know better how to deal better with paraphrasing and double meaning words. (i.e. Avoid the translation of ”Tudo bem?” to ”All right?”, but to ”How are you?”).
- Like mentioned, another good improvement could be to have conversations recorded in nursing homes and transcribe them to text, to better fine-tune the model. Understanding the important conversations topics for the elders would make it more adapt to the problem in cause.
- One more improvement, again to improve the data, could be to store the conversations conducted with the Chatbot APP and use them to inspect from where the bad classifications came and use the good ones to to enlarge our dataset.
- Lastly, some improvements to the evaluation could be done too. Instead of only using one random sentence to compute the metrics (BLEU, ROUGE, METEOR), the 5 predicted utterances (used to calculate the rankings) could be used as hypotheses to be compared with the reference.

- To evaluate the predictions of the model, the dataset, 6k multi-reference dataset created from Reddit data, used to evaluate the baseline of DialoGPT could have been used too. Instead of having only a dataset, split into question and answers and using the answers as reference.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[2] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "Dialogpt: Large-scale generative pre-training for conversational response generation," *arXiv preprint arXiv:1911.00536*, 2019.

[3] S. Bao, H. He, F. Wang, H. Wu, and H. Wang, "Plato: Pre-trained dialogue generation model with discrete latent variable," *arXiv preprint arXiv:1910.07931*, 2019.

[4] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "Dailydialog: A manually labelled multi-turn dialogue dataset," *arXiv preprint arXiv:1710.03957*, 2017.

[5] A. Richard, S. Rohrmann, C. L. Vandeleur, M. Schmid, J. Barth, and M. Eichholzer, "Loneliness is adversely associated with physical and mental health and lifestyle factors: Results from a swiss national survey," *PloS one*, vol. 12, no. 7, p. e0181442, 2017.

[6] M.-H. Su, C.-H. Wu, K.-Y. Huang, Q.-B. Hong, and H.-M. Wang, "A chatbot using lstm-based multi-layer embedding for elderly care," in *2017 International Conference on Orange Technologies (ICOT)*. IEEE, 2017, pp. 70–74.

[7] A. K. Bekhet and J. A. Zauszniewski, "Mental health of elders in retirement communities: Is loneliness a key factor?" *Archives of psychiatric nursing*, vol. 26, no. 3, pp. 214–224, 2012.

[8] B. Hanratty, D. Stow, D. Collingridge Moore, N. K. Valtorta, and F. Matthews, "Loneliness as a risk factor for care home admission in the english longitudinal study of ageing," *Age and ageing*, vol. 47, no. 6, pp. 896–900, 2018.

[9] M. Scheutz, R. Cantrell, and P. Schermerhorn, "Toward humanlike task-based dialogue processing for human robot interaction," *Ai Magazine*, vol. 32, no. 4, pp. 77–84, 2011.

[10] M. A. Goodrich and A. C. Schultz, *Human-robot interaction: a survey*. Now Publishers Inc, 2008.

[11] T. B. Sheridan, "Human–robot interaction: status and challenges," *Human factors*, vol. 58, no. 4, pp. 525–532, 2016.

[12] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, 2018.

[13] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[14] M. Henderson, I. Casanueva, N. Mrkšić, P.-H. Su, T.-H. Wen, and I. Vulić, "Convert: Efficient and accurate conversational representations from transformers," *arXiv preprint arXiv:1911.03688*, 2019.

[15] S. Ruder, "Why You Should Do NLP Beyond English," http://ruder.io/nlp-beyond-english, 2020.

[16] O. Firat, K. Cho, B. Sankaran, F. T. Y. Vural, and Y. Bengio, "Multi-way, multilingual neural machine translation," *Computer Speech & Language*, vol. 45, pp. 236–252, 2017.

[17] N. MCGUIRE, “How Accurate is Google Translate in 2019?” https://www.argotrans.com/blog/accurate-google-translate-2019/, 2019.

[18] J. Tiedemann and S. Thottingal, “Opus-mt–building open translation services for the world,” in *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, 2020, pp. 479–480.

[19] A. M. Turing, “Computing machinery and intelligence,” in *Parsing the turing test*. Springer, 2009, pp. 23–65.

[20] T. Winograd, “Procedures as a representation for data in a computer program for understanding natural language,” MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, Tech. Rep., 1971.

[21] A. Chopra, A. Prashar, and C. Sain, “Natural language processing,” *International journal of technology enhancements and emerging engineering research*, vol. 1, no. 4, pp. 131–134, 2013.

[22] K. Hao, “We analyzed 16.625 papers to figure out where AI is headed next,” https://www.technologyreview.com/2019/01/25/1436/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next, 2019.

[23] L. Lei, X. Jing, G. Yuan, Z. Yan-quan, W. Cong, and Z. Yi-xin, “Ci-nlu based text processing after speech recognition,” *Journal of Beijing University of Posts and Telecommunications*, vol. 29, no. s2, p. 192, 2006.

[24] P. Semaan, “Natural language generation: an overview,” *J Comput Sci Res*, vol. 1, no. 3, pp. 50–57, 2012.

[25] E. D. Liddy, “Natural language processing,” 2001.

[26] M. Dahiya, “A tool of conversation: Chatbot,” *International Journal of Computer Sciences and Engineering*, vol. 5, no. 5, pp. 158–161, 2017.

[27] F. Colace, M. De Santo, M. Lombardi, F. Pascale, A. Pietrosanto, and S. Lemma, “Chatbot for e-learning: A case of study,” *International Journal of Mechanical Engineering and Robotics Research*, vol. 7, no. 5, pp. 528–533, 2018.

[28] M. Neururer, S. Schlögl, L. Brinkschulte, and A. Groth, “Perceptions on authenticity in chat bots,” *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 60, 2018.

[29] A. P. Chaves and M. A. Gerosa, “How should my chatbot interact? a survey on social characteristics in human–chatbot interaction design,” *International Journal of Human–Computer Interaction*, pp. 1–30, 2020.

[30] S. Ruan, L. Jiang, J. Xu, B. J.-K. Tham, Z. Qiu, Y. Zhu, E. L. Murnane, E. Brunskill, and J. A. Landay, “Quizbot: A dialogue-based adaptive learning system for factual knowledge,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.

[31] K. Gopalakrishnan, B. Hedayatnia, Q. Chen, A. Gottardi, S. Kwatra, A. Venkatesh, R. Gabriel, D. Hakkani-Tür, and A. A. AI, “Topical-chat: Towards knowledge-grounded open-domain conversations.” in *INTERSPEECH*, 2019, pp. 1891–1895.

[32] C. Henrique and O. Silva, “Natural Language Processing : My ” grandchild-Bot ” Electrotechnical and Computer Engineering,” no. June, 2020.

[33] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[34] A. Kulkarni and A. Shivananda, *Natural language processing recipes: Unlocking text data with machine learning and deep learning using python*. Apress, 2019.

[35] Z. S. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[37] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” *arXiv preprint arXiv:1802.05365*, 2018.

[38] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[40] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[41] T. K. Kumar and A. LeenaRani, "Generate and ranking the multiple sentences based on context using natural language generation."

[42] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.

[43] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[46] S. Hussain, O. A. Sianaki, and N. Ababneh, "A survey on conversational agents/chatbots classification and design techniques," in *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer, 2019, pp. 946–956.

[47] A. Sojasingarayar, "Seq2seq ai chatbot with attention mechanism," *arXiv*, pp. arXiv–2006, 2020.

[48] S. Minaee, E. Azimi, and A. Abdolrashidi, "Deep-sentiment: Sentiment analysis using ensemble of cnn and bi-lstm models," *arXiv preprint arXiv:1904.04206*, 2019.

[49] K. Ramesh, S. Ravishankaran, A. Joshi, and K. Chandrasekaran, "A survey of design techniques for conversational agents," in *International Conference on Information, Communication and Computing Technology*. Springer, 2017, pp. 336–350.

[50] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, pp. 3104–3112, 2014.

[51] R. Csaky, "Deep learning based chatbot models," *arXiv*, pp. arXiv–1908, 2019.

[52] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[53] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[54] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[55] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[56] S. Ranathunga, E.-S. A. Lee, M. P. Skenduli, R. Shekhar, M. Alam, and R. Kaur, "Neural machine translation for low-resource languages: A survey," *arXiv preprint arXiv:2106.15115*, 2021.

[57] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev *et al.*, "Marian: Fast neural machine translation in c++," *arXiv preprint arXiv:1804.00344*, 2018.

[58] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," *arXiv preprint arXiv:1511.06709*, 2015.

[59] I. Caswell and B. Liang, “Recent advances in google translate.”

[60] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[61] A. Malte and P. Ratadiya, “Evolution of transfer learning in natural language processing,” *arXiv preprint arXiv:1910.07370*, 2019.

[62] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, “Spottune: transfer learning through adaptive fine-tuning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4805–4814.

[63] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *arXiv preprint arXiv:1411.1792*, 2014.

[64] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, “Convolutional neural networks for medical image analysis: Full training or fine tuning?” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[65] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.

[66] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” *arXiv preprint arXiv:1511.06732*, 2015.

[67] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra, “Diverse beam search: Decoding diverse solutions from neural sequence models,” *arXiv preprint arXiv:1610.02424*, 2016.

[68] R. Paulus, C. Xiong, and R. Socher, “A deep reinforced model for abstractive summarization,” *arXiv preprint arXiv:1705.04304*, 2017.

[69] Y. Yang, L. Huang, and M. Ma, “Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation,” *arXiv preprint arXiv:1808.09582*, 2018.

[70] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” *arXiv preprint arXiv:1904.09751*, 2019.

[71] A. Fan, M. Lewis, and Y. Dauphin, “Hierarchical neural story generation,” *arXiv preprint arXiv:1805.04833*, 2018.

[72] K. Gopalakrishnan, B. Hedayatnia, Q. Chen, A. Gottardi, S. Kwatra, A. Venkatesh, R. Gabriel, D. Hakkani-Tür, and A. A. AI, “Topical-chat: Towards knowledge-grounded open-domain conversations.” in *INTERSPEECH*, 2019, pp. 1891–1895.

[73] Google, “Speech-to-Text,” https://cloud.google.com/speech-to-text/, 2020.

[74] D. Adiwardana and T. Luong, “Towards a conversational agent that can chat about... anything,” *Google AI Blog*, 2020.

[75] N. Ketkar, “Introduction to pytorch,” in *Deep learning with python*. Springer, 2017, pp. 195–208.

[76] X. Gao, Y. Zhang, M. Galley, C. Brockett, and B. Dolan, “Dialogue response ranking training with large-scale human feedback data,” *arXiv preprint arXiv:2009.06978*, 2020.

[77] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[78] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.

[79] S. Dutta and D. Klakow, “Evaluating a neural multi-turn chatbot using bleu score,” *Universität des Saarlandes*, p. 10, 2019.

[80] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau, “Towards an automatic turing test: Learning to evaluate dialogue responses,” *arXiv preprint arXiv:1708.07149*, 2017.

[81] A. Agarwal and A. Lavie, “Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output,” in *Proceedings of the Third Workshop on Statistical Machine Translation*, 2008, pp. 115–118.

[82] S. Sharma, L. El Asri, H. Schulz, and J. Zumer, “Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation,” *CoRR*, vol. abs/1706.09799, 2017. [Online]. Available: http://arxiv.org/abs/1706.09799

[83] R. Ishida, “Text size in translation,” *W3C Internationalization. N. p*, vol. 3, 2007.

DialoGPT-small Usage

Listing A.1: DialoGPT-small example

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-small")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-small")

# Let's chat for 5 lines
for step in range(5):
    # encode the new user input, add the eos_token and return a tensor in Pytorch
    new_user_input_ids = tokenizer.encode(input(">> User:") + tokenizer.eos_token, return_tensors='pt')
    # append the new user input tokens to the chat history
    bot_input_ids = torch.cat([chat_history_ids, new_user_input_ids], dim=-1) if step > 0 else
        new_user_input_ids
    # generated a response while limiting the total chat history to 1000 tokens,
    chat_history_ids = model.generate(bot_input_ids, max_length=1000, pad_token_id=tokenizer.
        eos_token_id)
    # pretty print last ouput tokens from bot
    print("DialoGPT: {}".format(tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0],
        skip_special_tokens=True)))
```

B

Testing Utterances

Nº	Utterances	Corresponding Robot Utterance
1	Bom dia.	Bom dia.
2	Olá.	Bom dia.
3	Tudo bem?	Está tudo bem e consigo?
4	Tudo bem consigo?	Está tudo bem e consigo?
5	Tudo bem contigo?	Está tudo bem e consigo?
6	Está tudo a andar?	Está tudo bem e consigo?
7	Comigo está tudo.	Ainda bem.
8	Tambem estou bem.	Ainda bem.
9	Eu estou ótimo.	Ainda bem.
10	Eu estou excelente.	Ainda bem.
11	Comigo está tudo magnífico	Ainda bem.
12	Eu estou excelente.	Ainda bem.
13	Como está o tempo hoje?	Hoje está um tempo fantástico.
14	Como está o tempo à tarde?	Hoje está um tempo fantástico.
15	Como está a meteorologia?	Hoje está um tempo fantástico.
16	Qual é a meteorologia para esta manhã?	Hoje está um tempo fantástico.
17	Qual é a meteorologia para hoje?	Hoje está um tempo fantástico.
18	Devo ir até a casa do meu filho.	Aproveite o tempo com a sua família.
19	Hoje vou estar com o meu filho.	Aproveite o tempo com a sua família.
20	Vou jantar hoje com a minha filha.	Aproveite o tempo com a sua família.
21	Hoje vou almoçar com o meu irmão.	Aproveite o tempo com a sua família.
22	Vou a casa da minha filha.	Aproveite o tempo com a sua família.
23	Hoje vou passar o dia com os meus netos.	Aproveite o tempo com a sua família.
24	Vou hoje ao supermercado.	O que é que vai comprar?
25	Hoje vou as compras, preciso de comida.	O que é que vai comprar?
26	Vou as lojas de roupa hoje.	O que é que vai comprar?
27	Vou as lojas à tarde.	O que é que vai comprar?
28	Hoje vou ao centro comercial.	O que é que vai comprar?
29	Vou ao centro comercial de manhã.	O que é que vai comprar?
30	Hoje quero passear.	Então divirta-se.
31	Vou dar um passeio.	Então divirta-se.
32	Quero dar uma volta esta tarde.	Então divirta-se.
33	Vou sair de casa hoje.	Então divirta-se.
34	Hoje vou dar uma volta.	Então divirta-se.
35	Hoje vou fazer uma excursão.	Então divirta-se.
36	Gosto de tomar um café.	Eu também, o sabor é ótimo.
37	Adoro um café todas as manhãs.	Eu também, o sabor é ótimo
38	Amo café.	Eu também, o sabor é ótimo
39	Nao consigo passar sem um café por dia.	Eu também, o sabor é ótimo
40	A minha bebida favorita é café.	Eu também, o sabor é ótimo
41	Gosto de tomar um chá	Eu prefiro café em relaçao ao chá.
42	Adoro um chá todas as manhãs.	Eu prefiro café em relaçao ao chá.
43	Amo chá.	Eu prefiro café em relaçao ao chá.
44	Nao consigo passar sem um chá por dia.	Eu prefiro café em relaçao ao chá.
45	A minha bebida favorita é chá.	Eu prefiro café em relaçao ao chá.

Table B.1: Utterances used in the testing phase

Nº	Translated Utterances	Translated Corresponding Robot Utterance
1	Good morning.	Good morning.
2	Hello.	Good morning.
3	All right?	Is everything okay with you?
4	Everything okay with you?	Is everything okay with you?
5	Is everything all right with you?	Is everything okay with you?
6	Is everything moving?	Is everything okay with you?
7	Everything's great with me.	Good.
8	I'm fine, too.	Good.
9	I'm fine.	Good.
10	I'm excellent.	Good.
11	With me this is all magnificent.	Good.
12	I'm excellent.	Good.
13	How's the weather today?	Today the weather is great.
14	How's the afternoon weather?	Today the weather is great.
15	How's the weather?	Today the weather is great.
16	What's the weather for this morning?	Today the weather is great.
17	What's the weather for today?	Today the weather is great.
18	I must go to my son's house.	Enjoy your time with your family.
19	Today I'll be with my son.	Enjoy your time with your family.
20	I'm having dinner tonight with my daughter.	Enjoy your time with your family.
21	I'm having lunch with my brother today.	Enjoy your time with your family.
22	I'm going to my daughter's.	Enjoy your time with your family.
23	Today I'm spending the day with my grandchildren.	Enjoy your time with your family.
24	I'm going to the supermarket today.	What are you going to buy?
25	I'm going shopping today, I need food.	What are you going to buy?
26	I'm going to the clothes shops today.	What are you going to buy?
27	I'm going to the shops in the afternoon.	What are you going to buy?
28	I'm going to the mall today.	What are you going to buy?
29	I'm going to the mall in the morning.	What are you going to buy?
30	I want to take a walk today.	Then have fun.
31	I'm going for a walk.	Then have fun.
32	I want to take a walk this afternoon.	Then have fun.
33	I'm leaving the house today.	Then have fun.
34	I'm going for a walk today.	Then have fun.
35	I'm going on a tour today.	Then have fun.
36	I like to have coffee.	Me too, the taste is great.
37	I love coffee every morning.	Me too, the taste is great.
38	I love coffee.	Me too, the taste is great.
39	I can't get through without a cup of coffee a day.	Me too, the taste is great.
40	My favorite drink is coffee.	Me too, the taste is great.
41	I like to have tea.	I prefer coffee to tea.
42	I love tea every morning.	I prefer coffee to tea.
43	I love tea.	I prefer coffee to tea.
44	I can't get through without a tea a day.	I prefer coffee to tea.
45	My favorite drink is tea.	I prefer coffee to tea.

Table B.2: Translated utterances used in the testing phase